

USING NLP TO RESOLVE MISMATCHES BETWEEN JOBSEEKERS AND
POSITIONS IN RECRUITMENT

THOMAS AF GREEN



PhD Thesis

PhD Candidate

Department of Computer Science

Faculty of Engineering

The University of Sheffield

September 2023

Thomas AF Green: *Using NLP to Resolve Mismatches Between Jobseekers and Positions in Recruitment*, PhD Thesis, © September 2023

ABSTRACT

Recruiting through online portals has seen a dramatic increase in recent decades and it is challenging for job seekers to evaluate the overwhelming amount of data to efficiently identify positions that align with their skills and qualifications. This research addresses this issue by investigating automatic approaches that leverage recent developments in Natural Language Processing (NLP) that search, parse, and evaluate the often unstructured data in order to find appropriate matches. We present the development of a benchmark suite consisting of an annotation schema, training corpus and baseline model for Entity Recognition (ER) in job descriptions, published under a Creative Commons licence. The dataset contains 18.6k entities comprising five types: *Skill*; *Qualification*; *Experience*; *Occupation*; and *Domain*. We develop a benchmark Conditional Random Fields (CRF) ER model which achieves an F_1 score of 0.59, and our best performing model utilises Bidirectional Encoder Representations from Transformers (BERT) and achieves an F_1 score of 0.73. We consider different ways of framing the matching problem and develop Machine Learning (ML) models to address each. We propose that the Natural Language Inference (NLI) paradigm most closely aligns with the matching problem. Our best performing model utilises decomposable attention and achieves an F_1 score of 0.73 on a job application success prediction task. Finally, we integrate the ER and success prediction models into a cohesive pipeline that predicts whether a given job application made by a user will be successful, which can be extended into a system that recommends suitable jobs to a user. Although we observe poorer results on this pipeline relative to a more simple input truncation approach, we suggest this may be limited by the ER component for feature selection and the entity encoding process.

PUBLICATIONS

Green, Thomas AF, Diana Maynard, and Chenghua Lin (2022). “Development of a Benchmark Corpus to Support Entity Recognition in Job Descriptions.” In: pp. 1201–1208. URL: <http://www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.128.pdf>.

ACKNOWLEDGMENTS

This work was supported by the Centre for Doctoral Training in Speech and Language Technologies and their Applications funded by UK Research and Innovation [grant number EP/S023062/1].

I would like to express my heartfelt gratitude to my first supervisor, Dr Diana Maynard, for her unwavering support, guidance, and encouragement throughout my research. Our weekly meetings, insightful discussions, and her dedication to my academic growth played a pivotal role in shaping this thesis.

I would also like to acknowledge my second supervisor, Dr Chenghua Lin, and my Industry Advisor, Dan Kirkland, for their valuable feedback and guidance. I thank Tribepad for their support and for providing the job application data that made this research possible.

I extend my sincere thanks to the directors of the CDT, Professors Rob Gaizauskas and Thomas Hain, as well as the Research Supervisors and associated academics at the University of Sheffield who fostered and maintained a conducive research environment to support this academic work. In particular, I would like to thank Dr Stuart Wrigley for managing the delivery of the CDT programme and for being a jolly participant in our Friday ‘coffee and catch-ups’.

I would like to thank my CDT cohort for their support and companionship over the last four years. Their camaraderie, encouragement, and shared experiences have made this academic journey enjoyable and memorable.

I am deeply grateful to my family for all their support and encouragement over the last four years: Sara, Stuart, Alice, Joey, Sam, and Jack. My father, Stuart, proofread this thesis diligently and I thank him for his insightful comments and presentation suggestions.

Lastly, I want to offer special thanks to my wife and constant supporter, Tasmine. Her love, patience, and understanding have been my anchor through the most challenging period of my academic life. Her sacrifices and endless encouragement made this work possible.

CONTENTS

0.1	Acronyms	xix
1	Introduction	1
1.1	Research Motivation	1
1.2	Research Questions	4
1.3	Research Contributions	6
1.4	Thesis Outline	6
2	Preliminaries	9
2.1	Overview	9
2.2	Domain-Specific Data Structures	9
2.2.1	Structure of Job Descriptions	9
2.2.2	Structure of User Profiles	11
2.3	Workflow of Online Recruitment	11
2.4	Introduction to Natural Language Processing	13
2.4.1	TF-IDF	13
2.4.2	Word Embeddings	14
2.4.3	Entity Recognition	22
2.4.4	Text Classification	23
2.4.5	Natural Language Inference	23
2.5	Introduction to Machine Learning	24
2.5.1	Model Structures	24
2.5.2	Model Training	35
2.5.3	Model Validation	36
2.5.4	Model Evaluation	38
2.6	Introduction to Recommender Systems	40
2.6.1	Content-Based Filtering	41
2.6.2	Collaborative Filtering	43
2.6.3	Knowledge-Based Filtering	46
2.6.4	Hybrid Filtering	47
2.7	Evaluating Human Annotations	48

2.7.1	Accuracy versus Gold Standard	49
2.7.2	Inter-Annotator Agreement	49
2.8	Equality, Diversity, and Inclusion in Recruitment Algorithms	51
3	Related Work	53
3.1	Overview	53
3.2	Job Recommendation Systems	53
3.2.1	RecSys 2016	58
3.2.2	Job Recommendation Challenge	61
3.2.3	Content-Based Filtering	62
3.2.4	Collaborative Filtering	63
3.2.5	Knowledge-Based Filtering	64
3.2.6	Hybrid Filtering	65
3.3	Large Scale Analysis of Job Descriptions	66
3.4	Natural Language Processing in Job Descriptions	68
3.4.1	Word Embeddings	68
3.4.2	Entity Recognition	69
3.4.3	Natural Language Inference	74
4	Extracting Salient Entities from Job Descriptions	77
4.1	Chapter Overview	77
4.2	Introduction	77
4.3	Unlabelled Job Description Data Acquisition	79
4.3.1	Discrepancies of Job Description Style and Purpose	79
4.4	Schema Development	82
4.5	Accuracy Threshold Identification	98
4.6	Corpus Development	101
4.7	Corpus Statistics	101
4.8	Entity Recognition Method Development	107
4.8.1	Data Preprocessing	107
4.8.2	Baseline CRF Model	109
4.8.3	BiLSTM-CRF Model	110
4.8.4	Convolutional Neural Network	112
4.8.5	Transformer-Based Models	113
4.9	Entity Recognition Evaluation	116

4.9.1	Baseline CRF Model	116
4.9.2	Competitive ER Models	118
4.10	Application of Entity Recognition Models	121
4.11	Ethical Considerations	122
4.12	Publication of Materials	123
4.13	Conclusion	124
5	Matching Candidate Profiles and Job Descriptions	125
5.1	Chapter Overview	125
5.2	Introduction	125
5.3	Framing the Matching Problem	126
5.3.1	Recommendation Problem	127
5.3.2	Text Classification Problem	128
5.3.3	Natural Language Inference Problem	128
5.4	Kaggle Job Recommendation Challenge	130
5.4.1	Corpus Analysis	131
5.4.2	Application Prediction Task	134
5.5	Tribepad Application Corpus	141
5.5.1	Corpus Statistics	142
5.5.2	Application Prediction Task	145
5.5.3	Status Prediction Task	147
5.5.4	Using Short-Form Models on Long-Form Input Sequences	161
5.6	Matching Pipeline	167
5.6.1	Matching Pipeline Overview	167
5.6.2	Data Selection	168
5.6.3	Feature Extraction	170
5.6.4	Embedding Method	171
5.6.5	Model Architecture	173
5.6.6	Output Prediction	174
5.6.7	Evaluation	174
5.7	Conclusion	176
6	Concluding Remarks	179
6.1	Assessment of Contributions	181
6.1.1	Extracting Salient Entities from Job Descriptions	181

6.1.2	Matching Candidate Profiles and Job Descriptions	182
6.2	Limitations	182
6.3	Future Work	183
6.3.1	Extending the Salient Entity Extraction Systems	183
6.3.2	Extending the Matching Pipeline	184
6.3.3	The Skills Delta	185
6.4	Impact of Thesis Contributions	189
A	Appendix	191
A.1	Entity Recognition Corpus	191
A.1.1	Annotation Materials	191
A.2	ER Model Results	205
A.3	TribePad Matched User Profile - Job Description Corpus	205
A.3.1	Status Codes	208
A.3.2	career data	209
A.3.3	education data	210
A.3.4	job data	211
A.3.5	skills data	213
A.3.6	user data	214
A.4	Ethical Approval	215
	Bibliography	227

LIST OF FIGURES

Figure 1.1	Sample text from CV A.	3
Figure 1.2	Sample text from CV B.	3
Figure 2.2	Example of a Job Description.	10
Figure 2.3	Flowchart of Pre-Application Online Recruitment.	12
Figure 2.4	Flowchart of Post-Application Online Recruitment.	12
Figure 2.5	Visualisation of Word Embeddings	15
Figure 2.6	Visualisation of Euclidean Distance.	18
Figure 2.7	Visualisation of Manhattan Distance.	19
Figure 2.8	Visualisation of Cosine Similarity.	21
Figure 2.9	Example Named Entity Recognition (NER)-tagged sentence. . .	22
Figure 2.10	Traditional Machine Learning (TML) and Deep Learning (DL) Model Architecture.	25
Figure 2.11	Diagram of a Perceptron.	28
Figure 2.12	Visualisation of Activation Functions.	30
Figure 2.13	Diagram of a Multi-Layer Perceptron.	32
Figure 2.14	An example of a Recurrent Neural Network (RNN).	32
Figure 2.15	An example of a Long Short-Term Memory (LSTM).	33
Figure 2.16	An example of a Gated Recurrent Unit (GRU).	34
Figure 2.17	An example of a Convolutional Neural Network (CNN).	35
Figure 2.18	Visualisation of Model Overfitting.	37
Figure 2.19	Visualisation of Content-based Filtering job recommendation system.	42
Figure 2.20	Visualisation of Collaborative Filtering recommendation system.	43
Figure 2.21	Visualisation of Knowledge-based Filtering recommendation system.	46
Figure 2.22	Visualisation of Hybrid recommendation system.	48
Figure 3.1	Architecture of a generic job matching system.	54
Figure 3.2	Taxonomy of Job Recommendation Systems.	57
Figure 4.1	Visualisation of RQ1 Project Workflow.	78

Figure 4.2	Example unlabelled Human Intelligence Task (HIT) on the DataTurks platform.	86
Figure 4.3	Example labelled HIT on the DataTurks platform.	87
Figure 4.4	Example unlabelled HIT on the Amazon SageMaker platform.	90
Figure 4.5	Example labelled HIT on the Amazon SageMaker platform.	91
Figure 4.6	Random Noise Induction Experiment Results.	99
Figure 4.7	Systematic Noise Induction Experiment Results.	100
Figure 4.8	Qualification Task Results.	102
Figure 4.9	Span token count distribution in the live corpus.	105
Figure 4.10	Visualisation of the CRF model.	110
Figure 4.11	Visualisation of the Bidirectional Long Short-Term Memory (BiLSTM)-CRF model.	111
Figure 4.12	Visualisation of the CNN model.	112
Figure 4.13	Visualisation of the BERT model.	114
Figure 5.1	Tribepad Corpus Status Transition Graph.	148
Figure 5.2	Application Prediction Model Architecture.	155
Figure 5.3	Token Count Distribution of Tribepad Data.	162
Figure 5.4	Visualisation of the Job Matching Pipeline.	168
Figure 5.5	Visualisation of Feature Extraction Component.	172
Figure 6.1	Visualisation of Job Transition Graph.	188

LIST OF TABLES

Table 2.1	Example Confusion Matrix.	38
Table 2.2	Interpretation of Fleiss' Kappa.	50
Table 3.1	Inter-Annotator Agreement (IAA) for Stanford Natural Language Inference Corpus (SNLI) Dataset.	75
Table 4.1	Unlabelled Job Description Corpus Statistics.	80
Table 4.2	Round 1 of corpus annotation development.	85
Table 4.3	Round 2 of corpus annotation development.	89

Table 4.4	Round 3 of corpus annotation development.	93
Table 4.5	Example of disagreement in annotation task.	93
Table 4.6	Problems and solutions for the annotation task.	94
Table 4.7	Round 4 of corpus annotation development.	95
Table 4.8	Round 5 of corpus annotation development.	96
Table 4.9	Round 6 of corpus annotation development.	96
Table 4.10	Annotated corpus statistics.	102
Table 4.11	Class distribution for the live, aggregated corpus.	103
Table 4.12	Class distribution for the test set.	103
Table 4.13	IAA on the live corpus.	104
Table 4.14	Frequent entities in the live corpus.	106
Table 4.15	Class distribution for the preprocessed data.	109
Table 4.16	Results for CRF ER model.	116
Table 4.17	Results for the ER task.	119
Table 5.1	Summary of Job Recommendation Challenge data.	132
Table 5.2	Common Majors in the Kaggle Job Recommendation Challenge (KJRC) Corpus.	133
Table 5.3	Common Job Titles in the KJRC Corpus.	134
Table 5.4	Job Description Challenge Model Results.	137
Table 5.5	Summary of Tribepad data.	141
Table 5.6	Summary of apps data.	142
Table 5.7	Baseline Application Prediction Model Performance on Tribepad Corpus.	146
Table 5.8	Tribepad Status Prediction, {Hired; Rejected}.	149
Table 5.9	Tribepad Status Prediction, {Interview; No Interview}.	150
Table 5.10	Status Prediction Results on the Tribepad & SNLI corpora.	158
Table 5.11	Truncation method comparison on Tribepad corpus using <i>Hired</i> dichotomy.	165
Table 5.12	Truncation method comparison on Tribepad corpus using <i>Inter-</i> <i>viewed</i> dichotomy.	165
Table 5.13	Data Selection for the Job Matching Pipeline.	169
Table 5.14	Encoder Model Attributes for the Job Matching Pipeline.	173
Table 5.15	Results for the Matching Pipeline on the Tribepad corpus.	175

Table 5.16	Results for UMAP Experiments on the Matching Pipeline. . . .	177
Table A.1	Results for BERT base uncased ER model.	205
Table A.2	Results for DistilBERT base uncased ER model.	206
Table A.3	Results for BERT base cased ER model.	206
Table A.4	Results for BERT base multilingual cased ER model.	207
Table A.5	Summary of apps data.	208
Table A.6	Summary of career data.	209
Table A.7	Summary of education data.	210
Table A.8	Summary of job data.	212
Table A.9	Summary of skills data.	213
Table A.10	Summary of user data.	214

GLOSSARY

job description the information pertaining to the job. This may be entirely without coherent structure, and may or may not contain aspects of the job such as the job title, role description, requirements for applicants in terms of skills or experience, location, or salary. Unless specified otherwise, this term includes ‘job adverts’, which are descriptions of the job intended to entice the user. Referred to in related literature as a ‘job listing’, ‘job post’, or simply ‘job’. [xvii](#), [9](#)

skill the attributes of a user in terms of attitudes, knowledge, and competencies that enable them to complete tasks. Note that this definition is not consistent across existing literature, where the same term may only refer to a subset of terms encompassed by the definition given here. [xvii](#), [xix](#)

skillset the set of skills. If used in the context of a user then it refers to the skills that user has. If used in the context of a job then it refers to the skills that job requires. See [skill](#). [xvii](#)

user an individual who submits job applications with the intention of seeking a job. Referred to in related literature as an ‘applicant’, ‘job seeker’, or ‘candidate’. [xvii](#)

user profile the information pertaining to the user. May include any combination of the information the user submits (e.g. uploaded CVs) or information obtained through the user’s engagement on an online platform (e.g. viewed job descriptions). Referred to in related literature as an ‘applicant profile’, ‘candidate profile’, and often used as a synonym of ‘CV’ or ‘resume’. [xvii](#), [11](#)

0.1 ACRONYMS

[AI](#) Artificial Intelligence

AMT Amazon Mechanical Turk

AUC Area Under the ROC Curve

BERT Bidirectional Encoder Representations from Transformers

BIGRU Bidirectional Gated Recurrent Unit

BILSTM Bidirectional Long Short-Term Memory

CONLL Computational Natural Language Learning

CNN Convolutional Neural Network

CRF Conditional Random Fields

DL Deep Learning

ESCO European Skills/Competences, Qualifications and Occupations Commission

EDI Equality, Diversity, and Inclusion

EQF European Qualifications Framework

ER Entity Recognition

GBRT Gradient Boosting Regression Tree

GLOVE Global Vectors for Word Representation

GRU Gated Recurrent Unit

GPT Generative Pretrained Transformer

HIT Human Intelligence Task

IAA Inter-Annotator Agreement

IDF Inverse Document Frequency

KJRC Kaggle Job Recommendation Challenge

KNN *k*-Nearest Neighbours

LDA Latent Dirichlet Allocation

LM Language Model

LSTM Long Short-Term Memory

MAP Mean Average Precision

MF Matrix Factorisation

ML Machine Learning

MLP Multi-Layer Perceptron

NER Named Entity Recognition

NLI Natural Language Inference

NLP Natural Language Processing

NLTK Natural Language Toolkit

NN Neural Network

OOV Out of Vocabulary

POS Part-of-Speech

RELU Rectified Linear Unit

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

SGD Stochastic Gradient Descent

SOTA state-of-the-art

SNLI Stanford Natural Language Inference Corpus

SVD Singular Value Decomposition

TF Term Frequency

TF-IDF Term Frequency - Inverse Document Frequency

TI Temporal Intensity

TML Traditional Machine Learning

UMAP Uniform Manifold Approximation and Projection

XGBOOST eXtreme Gradient Boosting

INTRODUCTION

1.1 RESEARCH MOTIVATION

Recruiting through online portals has seen a dramatic increase in recent decades, both in terms of the number of recruiters advertising job roles on online portals and the number of job seekers applying to them (Petre, Osoian, and Zaharie, 2016). One of the issues that job seekers and recruiters face is that there is an overwhelmingly large amount of data that needs to be reviewed in order to find ideal jobs or candidates, causing an *information overload* problem (Dhameliya and Desai, 2019). Alternatively, this problem can be seen as a *filter failure* problem (Shirky, 2008), in that the issue facing job seekers and recruiters is not solely due to the large amount of data, but rather the difficulty associated with effectively filtering the data to find appropriate jobs or candidates. The lack of efficient data parsing and filtering methods make the task of identifying appropriate jobs and candidates a costly and time-consuming process. For reference, the median cost per hire for Senior Managers is approximately £5,000 for senior level positions, and £2,000 for other positions¹.

To facilitate the job/candidate seeking process, automatic approaches have been developed that search, parse, and evaluate the available data in order to find appropriate matches (Balog et al., 2012). Although simplistic methods such as *keyword matching* are still used, these approaches will exclude suitable candidates if their profiles use different keywords or phrases to express the same concepts as those selected for matching. Contemporary solutions to matching problems tend to leverage recent advancements in the field of *NLP* and *ML* that are able to utilise the *semantic* properties of concepts (for example, the semantic similarity between the terms *building* and *constructing*) as opposed to purely morphological properties (for example, the sub-word *java* in the term *javascript*), and involve the evaluation of applicant profile - job description matches

¹ From a 2020 Chartered Institute of Personnel and Development survey. Figures include in-house resourcing time, advertising costs, agency or search fees. <https://www.cipd.org/uk/knowledge/reports>

to produce a ranked list of suitable recommendations (Malherbe, Cataldi, and Ballatore, 2015).

The dynamic nature of the job market necessitates continuous adaptation to evolving skill requirements by jobseekers and recruiting agents alike; new job vacancies emerge which require new skills, and candidate skill sets change over time (Petre, Osoian, and Zaharie, 2016). In this way, the job matching task itself is constantly changing, and automated processes that facilitate the matching task need to account for the evolution of the domain.

An NLP-driven matching solution would need to replicate the decision-making process of a human agent in order to function effectively. In order to accomplish this it is important that candidate and job selection preferences are understood. Attributes shown to be important for job seekers to varying extents include *remuneration*, *job responsibilities*, *work style*, *geographical region*, *work culture*, *company values*, and the *business model* (Iacovou and Thompson (2002); Poll (2020)). Some of these attributes can be represented numerically (for example, *remuneration*) and some can be extracted or inferred from text contained in job descriptions or associated metadata (for example, *job responsibilities* or *geographical region*), which make them suitable fields of data for consideration in an automatic matching solution. However, other important attributes are abstract and inherently subjective and cannot be represented in a way that can be considered by automatic matching solutions (for example, the *work culture* of a company or the *personability* of a candidate). For this reason, the hiring process cannot be entirely automatic in that it requires a human intelligence component beyond the initial stages. Although Although these components are not suitable for automation, a system using NLP and ML can significantly augment the initial filtering process, presenting ranked recommendations that align with the preferences of job seekers and recruiting companies. This research project focuses on the experimentation and development of the processes and algorithms that contribute to the efficacy of such a system. Beyond the application of this research to online job portals, the investigation of skill demands and the alignment of skills with job adverts can offer valuable insights into the dynamic nature of the job market.

There are three main challenges associated with developing an applicant profile-job description matching solution that have been investigated to varying extents by existing research in this field. Firstly, the format in which recruiting agents construct their job

Figure 1.1: Sample text from CV A.

Skills and Experience:

- Market Analysis
- Marketing Plans
- Stock Management

Figure 1.2: Sample text from CV B.

Solely responsible for identifying new product development opportunities by analysing market requirement information, creating specific marketing plans, and stock management of around 50 product ranges.

descriptions and the format in which job seekers provide their personal information (for example, their CV) is non-uniform and often entirely without coherent structure. Key information the author seeks to convey may be noted explicitly in bullet points or embedded in free text. Figures 1.1 and 1.2 show the way in which the same information may be represented in different surface forms across two different CVs. An effective matching solution would need to be able to extract the relevant information from non-uniform and potentially unstructured text.

The second challenge associated with developing an applicant profile-job description matching solution is that it is not clear which aspects of the job description or applicant profile are important when a candidate makes the decision to apply to a particular role or when a recruiter decides to invite a particular applicant to interview. Furthermore, it is not clear to what extent some aspects are more important than others. Although the information required for human agents to make these decisions is contained within the items to review, the specific details, and the relative importance of each, can only be found through empirical research. An effective matching solution would need to calculate the overall suitability of a given job description or candidate profile by considering each of the important aspects with its corresponding importance weighting.

Finally, the evaluation of matching solutions is difficult for a number of reasons. It is difficult to know how good a system is intrinsically, given the lack of data that can be used to evaluate an individual applicant profile-job description match. Although an application to a job listing made by a candidate may be seen as a possible indicator of a good fit for the applicant to the job, and the acceptance of an interview or job offer may be better indicators still, subsequent employment longevity may be limited for a

number of reasons, and it is infeasible to obtain an applicant's rating of the job after their engagement with the recruitment portal has ended. Given this lack of data, it is difficult to qualify any match as truly *successful*.

Given that intrinsic evaluation is difficult, *artificial* evaluation is generally used via matched applicant profile-job description datasets. However, the issue of subjectivity regarding the *correctness* of a given match persists. Furthermore, there is a lack of public data that can be used to compare different systems. Privacy concerns and protection of the data used to develop matching solutions (in particular, candidate profile data in the form of CVs) precludes the sharing of data between researchers and the academic community, which makes comparing the effectiveness of different matching solutions a difficult task.

Ultimately, a matching solution could be used to develop an online portal to assist jobseekers and recruiters. Jobseekers would be able to upload their CV and immediately view a ranked list of appropriate job matches, and similarly recruiting agents would be able to list their job opening and view a ranked list of suitable candidates. Since the matching problem is bilateral in that a good recommendation for a jobseeker is a job that is not only suitable for them, but a job for which they are also suitable, an effective matching solution would address both job recommendations for users and candidate recommendations for recruiters and therefore a single online portal that encompasses both forms of recommendation is preferable to separate platforms for each.

1.2 RESEARCH QUESTIONS

There are two main research questions this thesis aims to address. The first concerns the parsing of applicant profile and job description data in order to extract relevant information. The second involves utilising deep insight from available data combined with existing matching methodologies in order to develop a state-of-the-art applicant profile-job description matching solution.

RQ1 *How can salient entities in applicant profiles and job descriptions be identified and extracted for use in an applicant profile-job description matching solution?*

At the time of writing, there is no consensus regarding which entities within applicant profiles and job descriptions are important for developing matching solutions, and

there are discrepancies in term definitions (for example, the inconsistent definitions of *skills*, discussed in §3.4.2). Additionally, prior to the work described by this thesis, there were no public, labelled datasets that could be used in the development of a system for identifying and extracting salient elements. Formal definitions for these entities were required, as well as a public, labelled dataset for the purpose of developing entity extraction systems.

Using this dataset, various methods of feature extraction are developed and evaluated to establish benchmark performance. We compare the performance of *CRF*, *BiLSTM*, *CNN*, and transformer-based models on this task.

The scientific contributions of this research question are:

- A list of entity classifications and their definitions in the form of an annotation schema
- A public, labelled dataset for the development and evaluation of entity extraction systems
- A state-of-the-art system for extracting salient entities from applicant profiles and job descriptions

RQ2 *How can deeper understanding of the candidate and job be used to influence a candidate profile-job description matching solution?*

The second research question seeks to develop a novel matching solution by utilising the rich data contained within applicant profiles, job descriptions, and associated metadata. The investigation centres on job recommendation techniques, employing a publicly available corpus of job applications and a proprietary corpus provided by sponsor company Tribepad. Initial research involves framing the task as an application prediction problem, whereby models are trained to predict which job listings a candidate would engage with. This problem is then framed as a status prediction challenge, whereby models are trained to predict the outcome of job applications made by users for specific job positions. This bilateral approach allows for the investigation of job recommendation from distinct angles, and provides insights and methodologies that will shape future research and practical implementations within the recruitment domain.

The scientific contributions of this research question are:

- A novel approach to applicant profile-job description matching that leverages deep insight of applicant profiles and job descriptions

1.3 RESEARCH CONTRIBUTIONS

The main scientific contributions of this thesis, aligned with their corresponding research questions, are:

- A list of entity classifications and their definitions in the form of an annotation scheme for salient entities within applicant profiles and job descriptions, made publicly available (**RQ1**)
- A public, labelled dataset for the development and evaluation of entity extraction systems (**RQ1**)
- A state-of-the-art system for extracting salient entities from applicant profiles and job descriptions (**RQ1**)
- A novel approach to applicant profile-job description matching that leverages deep insight of applicant profiles and job descriptions (**RQ2**)

These scientific contributions necessitate research, adaptation, and development of various feature extraction methods, matching and ranking components, modelling techniques, and the use of appropriate traditional machine learning and deep learning approaches.

1.4 THESIS OUTLINE

Chapter 1 serves as the **Introduction** chapter, complete with the motivation of this thesis, the research questions this thesis aims to address, and the contributions of this research, and an outline of the thesis chapters.

Chapter 2 covers the **Preliminaries** required to understand the methods and contributions discussed in this thesis. This includes a basic description of the structure of the main sources of data: the Job Description and the Candidate Profiles, a brief introduction to **NLP**, **ML** algorithms discussed in this thesis, and Recommender Systems.

Chapter 3 contains the **Related Work** section, providing in-depth analysis of the literature used during the development of this thesis. This is divided into sections according to content, covering papers on Job Recommendation Systems, ER in Job Descriptions, and the KJRC.

Chapter 4 is the first original contribution chapter, and contains the experiments and contributions for **Extracting Salient Entities from Job Descriptions**. We describe the development of the annotation task and associated schema, and details of the live corpus. We present statistics describing the corpus, and describe methods for developing an Entity Recognition (ER) model trained on this data. We evaluate various ER models using this data, and detail the publication of these resources for use by the academic community.

Chapter 5 is the second original contribution chapter, and contains the experiments and contributions for **Matching Candidate Profiles and Job Descriptions**. In this chapter we define the matching problem and offer three lenses through which it can be viewed. We describe the public corpus used for job recommendation experiments, followed by more in-depth corpus analysis and initial application prediction experiments. We then describe the private corpus of job applications with in-depth corpus analysis, and present experiments in application prediction. We then describe experiments in status prediction and summarise our findings and contributions in job recommendation.

Chapter 6 contains the **Concluding Remarks**. We draw on the conclusions made in the original contribution chapters, assessing the contributions and discussing the limitations of this work. We end the thesis by proposing novel areas of future work that would expand on the work described by this thesis and discussing the impact of its contributions.

PRELIMINARIES

2.1 OVERVIEW

This chapter provides a foundational framework for understanding the experimental research described in this thesis, and encompasses essential concepts, methodologies, and terminologies. First, we describe the structure and form of the data specific to the recruitment domain in §2.2, and show the workflow of online recruitment in §2.3 to situate our work in context. We provide a brief overview of Natural Language Processing (NLP) in §2.4 and Machine Learning (ML) in §2.5 which includes a description of both Traditional Machine Learning (TML) in §2.5.1.1 and Deep Learning (DL) in §2.5.1.2. We describe common recommender systems in §2.6, and provide overview of human annotation evaluation in §2.7 and a brief description of Equality, Diversity, and Inclusion (EDI) considerations for research in this domain in §2.8.

2.2 DOMAIN-SPECIFIC DATA STRUCTURES

2.2.1 *Structure of Job Descriptions*

A **job description** is defined in this thesis as the *collection of details pertaining to a single job*. Therefore, the term may include the following attributes:

- Job Title
- Job Summary
- List of responsibilities and duties
- Skill requirements
- Qualification requirements

Licensed Real Estate Assistant - Computer savvy, Boca, Delray Area.
Job Title
Skill Requirement
Location

Figure 2.2: An example job description with multiple elements concatenated in one text field.

- Experience requirements
- Description of the ideal candidate
- Advertisement for candidates
- Salary and benefits
- Location
- Start date
- Contract type

Generally, there are three main components to a job description, some of which may be absent due to the context of publication:

- A technical document detailing the job, for internal use
- A marketing document detailing the job, for external review
- Additional details, such as location, salary

There is no uniform structure to a job description, and discrepancies in format will exist even within a given repository of job descriptions (for example, on job search engines such as Indeed¹).

Often, several of these elements are embedded in a single text field as opposed to appearing in delimited and separate fields. Consider the text extracted from a job description in Figure 2.2, in which the job title *Licensed Real Estate Assistant* is shown in a single text field along with a single skill requirement *computer savvy* and the location of the role *Boca, Delray Area* (an area of Boca Raton, Florida, USA).

Since these fields are concatenated and not explicitly delimited or labelled, methods are required for the identification and classification of these elements in text fields.

¹ <https://indeed.com>

2.2.2 Structure of User Profiles

A **user profile** is defined in this thesis as the *collection of details pertaining to a single candidate*. Therefore, the term may include the following attributes:

- Candidate name
- Candidate summary
- Contact information
- Employment history
- Education history
- List of acquired qualifications
- List of skills
- List of hobbies and interests
- References
- Location

Some online portals will require candidates to upload a CV when signing up to the platform², and others will require candidates to fill out multiple designated fields to appropriately segment the information contained in their CV³. Therefore, the elements that compose a candidate's profile may be in pre-defined and labelled fields, or otherwise contained in the text of a CV document.

2.3 WORKFLOW OF ONLINE RECRUITMENT

In order to gain an understanding of the application of the automatic matching solutions presented in this thesis, it is important to contextualise them within the workflow of the online recruitment process.

² for example, <https://www.cv-library.co.uk/>

³ for example, <https://www.totaljobs.com>

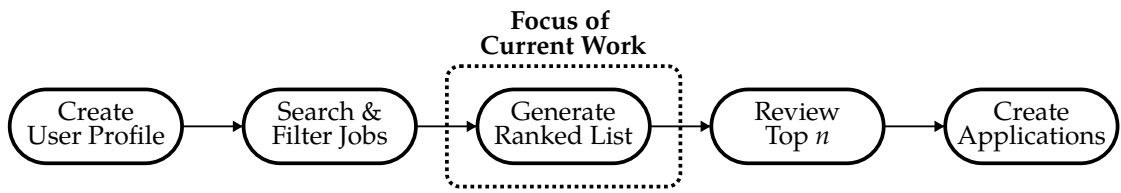


Figure 2.3: Flowchart of Pre-Application Online Recruitment.

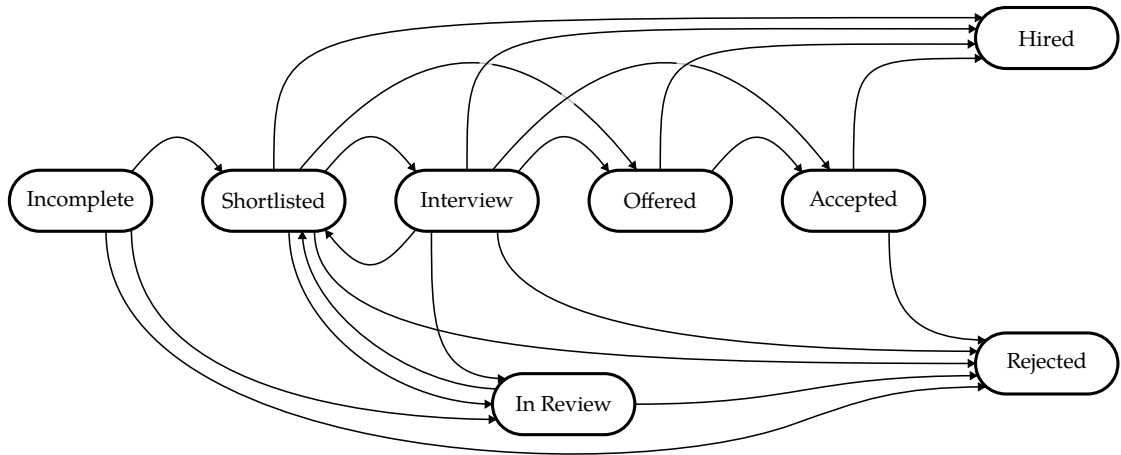


Figure 2.4: Flowchart of Post-Application Online Recruitment inferred from Tribepad hiring data.

Figure 2.3 visualises the typical initial stages of the online recruitment process from the creation of the user profile through to the point at which the job application is made. The highlighted section *generate ranked list* indicates the point at which our proposed matching solutions are implemented, whereby the remaining universe of job descriptions is evaluated to suggest the most appropriate items after applying user-defined filters.

Although the post-application stages will likely vary between online portals and applicant tracking systems employed by recruiting agents, Figure 2.4 visualises one example of the post-application stages used by Tribepad. Job application data that includes these application statuses or status transition sequences can be used to better inform matching solutions in that a successful application (for example, where the applicant was eventually hired) can be distinguished from an unsuccessful application (for example, where the applicant was rejected).

Transition arrows in Figure 2.4 are drawn from an analysis of status transition sequences from historic application data. The use of specific statuses varies between organisations using the Tribepad platform, which leads to some conceptually unusual transitions such as *Accepted* \rightarrow *Rejected*. In this instance, it is possible that the *Accepted* status is equivalent to *Hired, pending background checks*, and the subsequent transition to the *Rejected* status indicates that these checks failed.

2.4 INTRODUCTION TO NATURAL LANGUAGE PROCESSING

NLP is a sub-field of computer science and artificial intelligence and refers to the automatic analysis, understanding, and generation of human language.

Using NLP, we are able to create systems that replicate human decision-making for language-specific tasks. Typically, the advantages for doing so are described in terms of their speed and their reduced rate of random errors. Additionally, by delegating tasks to automated computer systems we are able to alleviate human workload.

Although the field of NLP encompasses several research topics, this thesis focuses on Term Frequency - Inverse Document Frequency (TF-IDF) (§2.4.1), a common method of evaluating word relevance in a document; word embeddings (§2.4.2), which are methods that generate mathematical representations of word semantics; Entity Recognition (§2.4.3), which refers to the identification, extraction, and classification of salient entities from text; Text Classification (§2.4.4), which refers to the prediction of class labels for text documents; and Natural Language Inference (§2.4.5), which refers to methods that infer entailment or contradiction between a premise and hypothesis.

2.4.1 TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) (Sparck Jones, 1988) is a measure that evaluates the relevance of a word relative to a document in a corpus. It is the product of the Term Frequency (TF) (Equation 2.1), which is the number of times the word appears in the document compared to the total number of words in the document, and the Inverse Document Frequency (IDF) (Equation 2.2), which is the proportion of documents in the corpus that contain the word. The addition of 1 to the

denominator, while not strictly necessary, avoids division by zero errors when the term does not appear in the corpus.

$$\text{TF} = \frac{\text{frequency of term in document}}{\text{total number of terms in document}} \quad (2.1)$$

$$\text{IDF} = \log \left(\frac{\text{total number of documents in corpus}}{1 + \text{total number of documents that contain the term}} \right) \quad (2.2)$$

Boolean TF is a variant of *TF* which sets the *TF* to 1 if the term is present in the document. This approach tends to be useful in contexts where it is important to capture the presence or absence of terms across a collection of documents, and where the frequency of those terms within documents is of less importance.

2.4.2 Word Embeddings

Word embeddings are methods for generating mathematical representations of text (Turney and Pantel, 2010). These *ML* methods learn vector representations for a fixed-length vocabulary from a corpus of text where vector values are based on how each word in the vocabulary appears in context. The resultant vector representations capture semantic relationships between words and can be used to calculate word similarities. Figure 2.5 shows a simplified example of vector represented words, in which the words *prince* and *woman* are shown in 3-dimensional space represented by vectors $(1, 2, 4)$ and $(5, 4, 1)$ respectively. In this example, the three axes represent *royalty*, *gender*, and *age*, although typically word embeddings are of considerably higher dimensionality and dimensions are not as easily human-interpretable.

Methods that create vector representations of words (for example, *word2vec* (Mikolov et al., 2013)) can be extended to create vector representations of variable-length pieces of text, such as sentences, paragraphs, or documents (for example, *doc2vec* (Le and Mikolov, 2014)).

These vector word representations can theoretically have any number of dimensions, but typically feature 50, 100, 200, or 300 dimensions (Pennington, Socher, and Christopher D Manning, 2014). In n -dimensional vector space, words that appear in similar contexts will be close together.

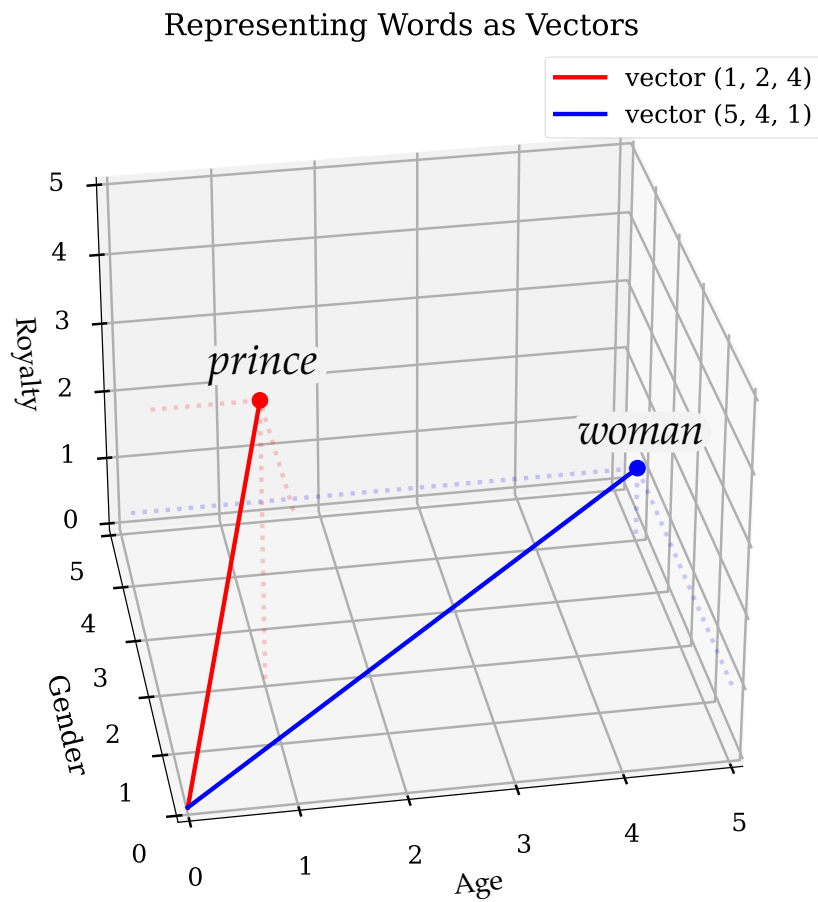


Figure 2.5: Visualisation of Word Embeddings, in which the words *prince* and *woman* are represented as vectors $(1, 2, 4)$ and $(5, 4, 1)$ respectively.

Since word embeddings aim to capture word semantics in vector form, they are useful in NLP tasks where the understanding of word meaning is important.

Although there are several established methods that can be used to train word embeddings, a popular option in academic research involves using *pretrained* word embeddings; large, publicly accessible files containing vector representations of a vocabulary of words, typically trained on a large amount of data using powerful computational resources. These pretrained word embeddings can be applied directly, or fine-tuned on a smaller dataset for a particular task. There are three notable advantages of using pretrained word embeddings in contemporary NLP research and applications. Firstly, using pretrained embeddings reduces the total computational resource cost of the task, given that training word embeddings is no longer required and that fine-tuning is considerably less computationally expensive than training directly. Secondly, there is an assurance that the trained embeddings are capturing semantic content given that they are ubiquitous in contemporary NLP research, and thirdly, they facilitate comparison between different models where the same embedding method has been applied; differences in performance across models can be reliably attributed to the models themselves rather than to variances in how the embeddings were created. The disadvantages of using pretrained model embeddings include the limited vocabulary; since pretrained word embeddings are essentially a dictionary of terms and their associated vector representations, words which were absent or infrequent in the documents used for pre-training will not be included, which necessitates an extra method for handling these Out of Vocabulary (OoV) words⁴.

Pretrained word embeddings available for academic use can be downloaded freely from online repositories^{5,6,7}, which detail the training process, training corpora, vocabulary size, and the dimensionality of the resultant vectors.

2.4.2.1 *Similarity Metrics*

Words that have similar meanings will correspond to word embeddings that are close together in vector space. This principle extends to sentences and documents that are

⁴ Typically, OoV words are ignored, although some embedding methods account for OoV words by using substrings (for example, <https://fasttext.cc>).

⁵ <http://vectors.nlp.lpl.eu/repository/>

⁶ <https://nlp.stanford.edu/projects/glove/>

⁷ <https://fasttext.cc/docs/en/english-vectors.html>

represented in vector space, in that similarity of semantic content will result in clustered embeddings. Similarity metrics are used to calculate the extent to which two words (or sentences etc.) are similar. These include: the length of the most direct path between the two points in vector space (Euclidean Distance); the length of the path from one point to another travelling perpendicularly (Manhattan Distance); a generalisation of the Euclidean and Manhattan Distance metrics (Minkowski Distance); or the size of the angle measured between the two vectors (Cosine Similarity).

EUCLIDEAN DISTANCE

The Euclidean distance metric measures the distance between two points represented in vector-space. A small Euclidean distance between two points is an indicator of similarity between two word embeddings. The formula for calculating Euclidean distance between two points $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$ is shown in equation 2.3.

$$d(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.3)$$

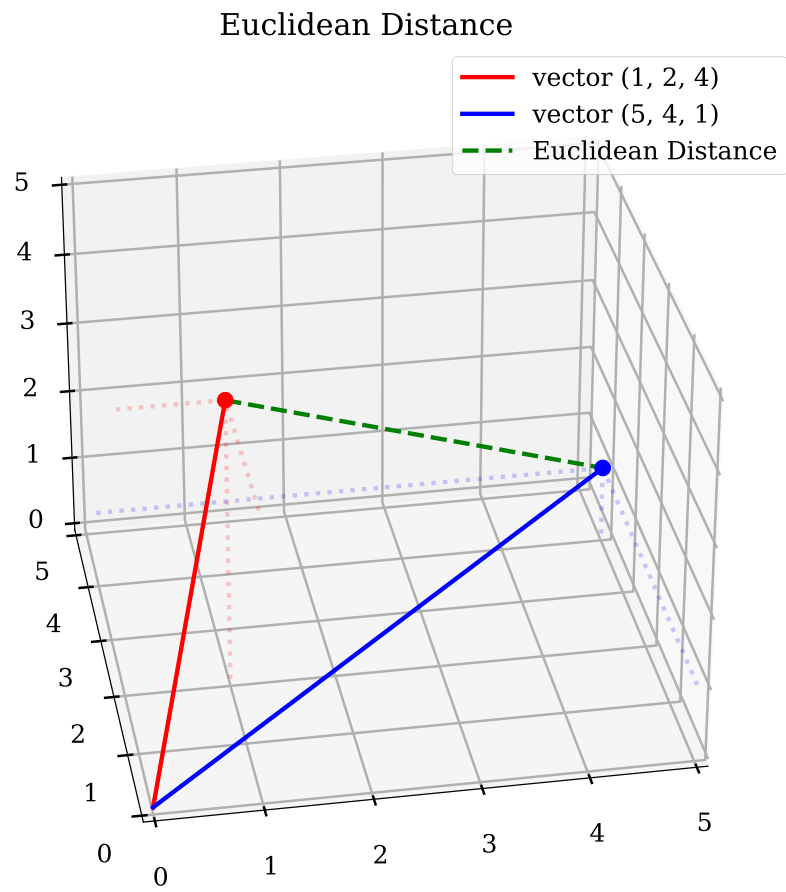
To illustrate the Euclidean distance metric, Figure 2.6 shows this distance between two vectors.

MANHATTAN DISTANCE

A similar metric for evaluating vector similarity is the Manhattan distance metric, which is the distance between two points in vector-space measured along axes at right angles. The formula for calculating Manhattan distance is shown in equation 2.4.

$$d_1(X, Y) = \sum_{i=1}^n |x_i - y_i| \quad (2.4)$$

To illustrate the Manhattan distance metric, Figure 2.7 shows the distance between two vectors.

Figure 2.6: Visualising the Euclidean Distance between vectors $(1, 2, 4)$ and $(5, 4, 1)$.

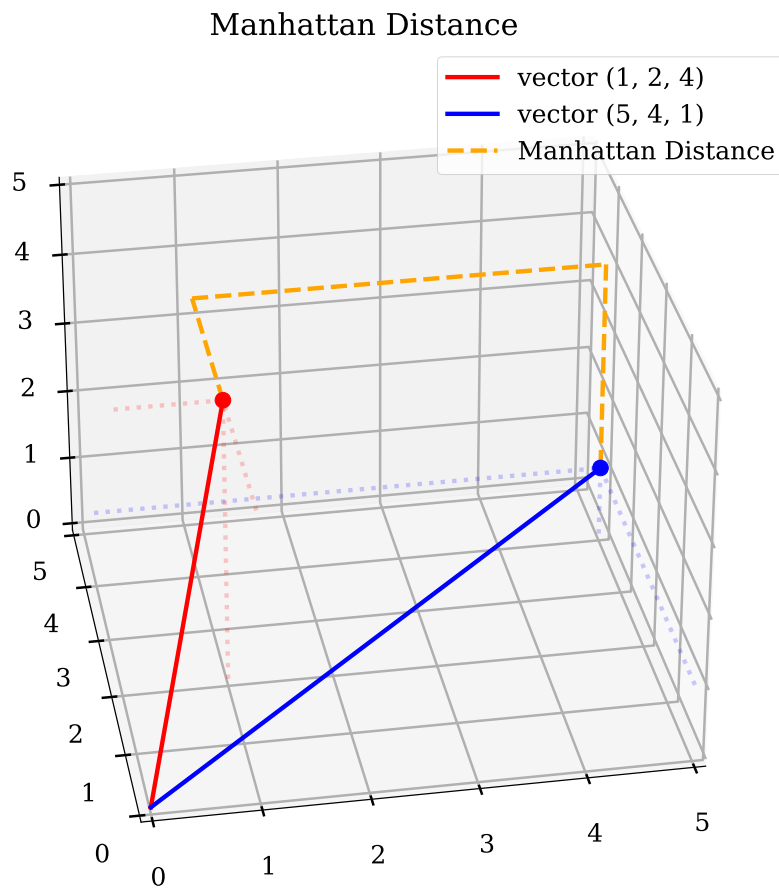


Figure 2.7: Visualising the Manhattan Distance between vectors (1,2,4) and (5,4,1).

MINKOWSKI DISTANCE

The Minkowski distance is a generalisation of both the Euclidean distance and Manhattan distance metrics, and varies depending on the chosen order of p ; when $p = 1$, this is equivalent to the Manhattan distance, and when $p = 2$ it is equivalent to the Euclidean distance. The formula for calculating Euclidean distance is shown in equation 2.5. When applying these metrics to machine learning tasks, C. Aggarwal, Hinneburg, and Keim (2001) suggest it may be preferable to use lower values of p for problems with high dimensionality, and so the Manhattan distance ($p = 1$) is preferred over the Euclidean distance ($p = 2$) as dimensionality of the data increases.

$$D(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (2.5)$$

COSINE SIMILARITY

A limitation of distance metrics (Euclidean, Manhattan, and Minkowski) for calculating similarity between two vectors is that they are not scale invariant. In the context of word or document embeddings, a short document with few words will be distant from a long document with many words even if the content of the documents is very similar. This can be addressed by using similarity metrics such as cosine similarity which remove the effect of document length.

Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between the two vectors. A high cosine (and therefore a small angle) between two vectors is an indicator of high similarity. The formula for calculating cosine similarity is shown in equation 2.6. Cosine similarity is ubiquitous in NLP research topics, particularly in cases where there is high variation in text length (Singhal, 2001).

$$\cos(\theta) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}} \quad (2.6)$$

To illustrate the cosine similarity metric, Figure 2.8 shows the angle between two vectors.

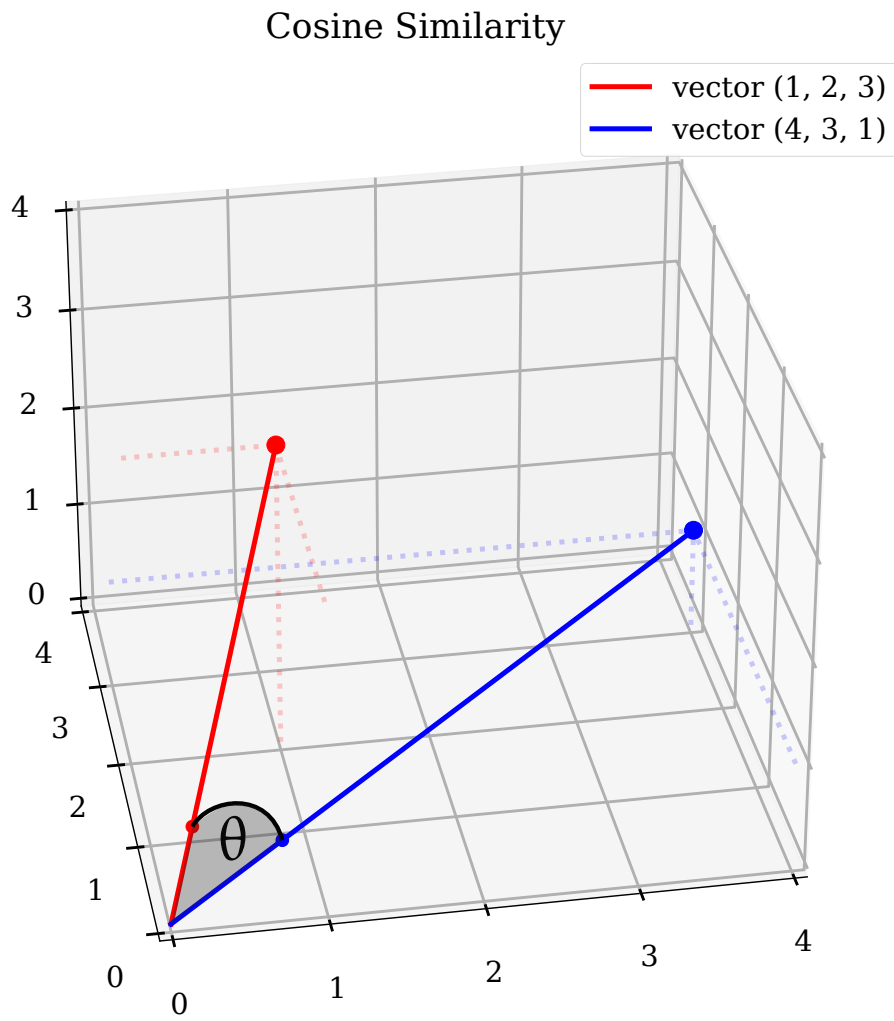


Figure 2.8: Visualising the cosine similarity between vectors (1,2,3) and (4,3,1).

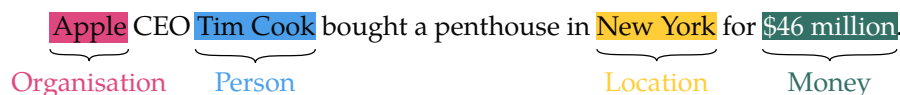


Figure 2.9: An example sentence with Named Entities identified and classified.

PEARSON CORRELATION COEFFICIENT

The Pearson correlation coefficient (Freedman, Pisani, and Purves, 2007), given by Equation 2.7, is also used as a similarity metric. It is essentially equivalent to the cosine similarity between word embedding vectors, but there are cases where it is inappropriate and cosine similarity is preferable (Zhelezniak et al., 2019).

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (2.7)$$

2.4.3 Entity Recognition

ER is a subtask of NLP that concerns the automatic identification and classification of entities in text. In this thesis we make the distinction between NER and ER by defining the former a specific application of the latter; *Named Entities* are standardised categories such as *People*, *Locations*, and *Organisations*, and NER methods identify and classify these Named Entities, whereas ER methods are not limited to standard entity classifications and may vary by task and schema.

Figure 2.9 shows an example sentence with Named Entities identified and classified.

Typically, the best-performing ER systems are developed using a supervised machine learning approach which requires a large dataset of human-annotated examples (Yadav and Bethard, 2019).

ER is used in several applications of NLP such as retrieving documents relevant to a search query (document retrieval), reducing a large article into its most salient points (automatic summarisation), and even for identifying the key elements that should be used as input into another NLP model (feature extraction). In this thesis, ER is used for

the latter purpose; for the extraction of salient entities in job descriptions and candidate profiles to be used as input to a model that is able to recommend appropriate jobs for users and vice versa.

2.4.4 Text Classification

Text classification is a subtask of NLP that involves the automatic categorisation of text documents into one or more predefined classes. This is distinct from ER since a text document may contain several entities, but a text classification model is typically used to assign a single class to the entire document.

Typically, the best-performing text classification models are developed using a supervised machine learning approach which requires a large dataset of labelled examples (Q. Li et al., 2022).

Text classification models may be configured to accept any form of input, such as the documents themselves, or the features extracted from the text documents using a feature extraction method such as ER.

2.4.5 Natural Language Inference

NLI is a subtask of NLP that concerns the task of identifying the relationship between a given premise and an associated conclusion (MacCartney and Christopher D. Manning, 2008). Consider the following premise: *a man wearing blue jeans is painting his garage*. A conclusion that could logically be drawn from this premise, such as *the man is painting*, is said to *entail* this hypothesis. Conversely, a conclusion that can be shown to be incorrect, such as *the man is building a shed*, is said to *contradict* the hypothesis. A conclusion that neither entails nor contradicts the premise, such as *the paint is blue*, is considered *neutral*. The goal of an NLI model is to automatically predict the relationship between premise and conclusion and to select an appropriate classification from a set of possible labels, typically one of {entail; contradict; neutral}.

Contemporary NLI models utilise pretrained word embeddings (§2.4.2) and feature some form of *fusion* layer in their model architecture, whereby the encoded representations of the premise and conclusion are compared, and it is the comparison of premise

and conclusion that drives model prediction as opposed to the encoded inputs themselves (Storks, Gao, and Chai, 2020).

2.5 INTRODUCTION TO MACHINE LEARNING

ML refers to the computer models and systems that are able to improve their performance on a given task for which they have not been explicitly programmed. This is achieved by applying statistical methods that are able to identify patterns and infer information from training data. **ML** can be applied to tasks that are conceptually simple, such as drawing a line of best fit through a data set (regression analysis), and tasks that are conceptually extremely complex, such as object detection in self-driving vehicles.

Although **ML** is a distinct sub-field of Artificial Intelligence (**AI**) from **NLP**, **ML** methods have revolutionised the field of **NLP** and are ubiquitous in modern approaches to language-oriented tasks (Nagarhalli, Vaze, and Rana, 2021).

2.5.1 Model Structures

There are two general approaches to **ML**. The first is referred to in this thesis as Traditional Machine Learning, and pertains to the use of standardised statistical models applied to structured data. The second is referred to in this thesis as Deep Learning, which pertains to the use of Neural Networks (**NNs**) applied to large volumes of unstructured data. A further distinction between the two approaches is that, when a **TML** algorithm generates a series of incorrect predictions, intervention is necessary and manual model adjustments are required, but conversely, when **DL** models generate incorrect predictions, the model itself can determine inaccuracies and self-adjust accordingly, learning features and model structure from unstructured data and self-improving without manual intervention.

Figure 2.10 shows a high-level summary of the architecture of **TML** and **DL** methods. While a **TML** model typically requires a *data preprocessing* step to parse and form input data into a usable format, and a *feature extraction* step to further identify the aspects of the data that contain the salient information the model should draw inference from in order to make predictions, a **DL** model does not require these explicit steps since they

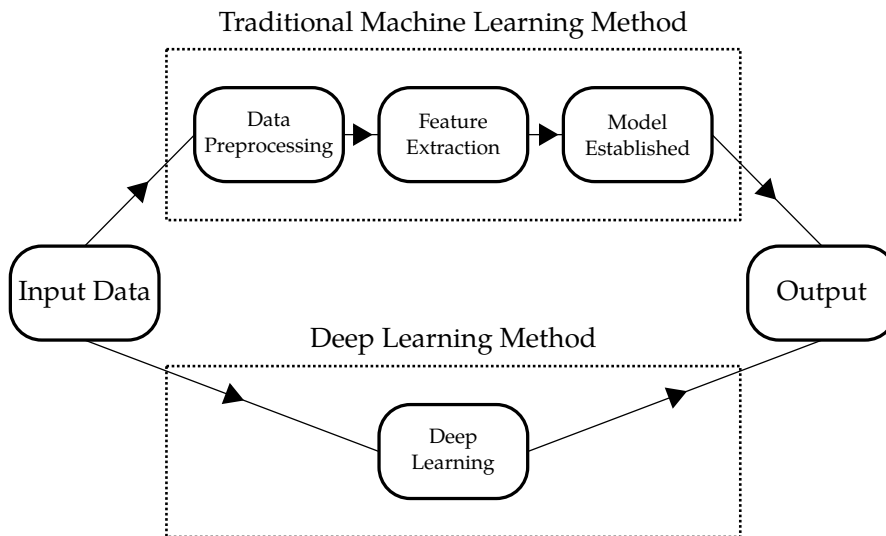


Figure 2.10: An illustration of the difference in model architecture between TML and DL.

are contained in the model training process; the DL model learns the features directly from the unstructured data.

While DL offers several advantages over TML in terms of its higher performance in cases of large volumes of training data and the lack of an explicit *feature extraction* step (which makes DL models desirable when the salient features of the input data are unknown), they are unsuitable in many cases, and in these cases, TML methods are more appropriate.

DL is unsuitable in cases where the volume of available training data is limited, and in cases where it is not acceptable to employ an automatic solution in the absence of clear reasoning as to how a given prediction or recommendation has been made; since DL models are typically complex, featuring large numbers of nodes and connections between them, it is exceptionally difficult to understand exactly which aspect of the data each part of the model is extracting and abstracting, and the reasoning behind a prediction made by a DL model is often inextricable. This is true to a lesser extent in TML models, which may be more transparent in comparison. Research into the explainability of DL models is a popular area of AI research (Saranya and Subhashini, 2023) as it addresses the critical need for transparency and interpretability in AI systems, making them more accessible and their predictions more trustworthy.

In this thesis, we investigate methods of evaluating candidate-job fit and making data-driven recommendations, and to this end, a method that is not entirely understood does not constitute an acceptable solution since it could not be implemented without risking violation of the principles of EDI. This is discussed in more detail in §2.8.

2.5.1.1 Traditional Machine Learning

TML refers to the use of standardised statistical models applied to structured data. It is distinct from DL in that TML algorithms do not use hidden layer architectures. The two main advantages of TML over DL are:

- improved performance in cases where data is limited; DL methods outperform TML methods when vast amounts of data are available, but perform poorly when data is limited.
- comparatively greater model transparency; since TML methods do not involve hidden layers, it is relatively simpler to interpret why a model has made a particular prediction or recommendation. However, interpreting the processes of a TML model may still be difficult for a non-expert, and so there are still inherent risks for the users of these systems.

CONDITIONAL RANDOM FIELDS

CRF (Lafferty, McCallum, and Fernando Pereira, 1999) is a type of TML model and is commonly applied to structured prediction tasks, such as ER, to model structural dependencies. Due to its simplicity and adequate performance, CRF presents an appropriate benchmark setting for sequence prediction tasks such as ER. In CRF, the *feature function* captures the compatibility between a particular output sequence and the input observations. Its formula is shown in Equation 2.8, where X is the set of input observations, y_i is the label at data point i , $Z(X)$ is the normalisation, and λ is the learned feature function weights. CRF is a probabilistic graph model that takes neighboring sample context into account when making predictions.

$$p(y|X, \lambda) = \frac{1}{Z(X)} \exp \sum_{i=1}^n \sum_j \lambda_j f_i(X, i, y_{i-1}, y_i) \quad (2.8)$$

The Natural Language Toolkit (NLTK)⁸, a popular toolkit for NLP tasks, selects word features for CRF including word identity, suffix, shape, and Part-of-Speech (POS) tags, as well as information from adjacent words in the sequence.

LOGISTIC REGRESSION

Logistic Regression (McCullagh and Nelder, 1989) is a TML model applied to binary classification tasks. The model calculates the *hyperplane*, which is the linear boundary in the feature space that maximises the separation between the data associated with each of the binary classes. The sigmoid function, given by equation 2.9, is used to transform the combination of input features and model parameters to a score between 0 and 1. This score indicates the probability that the item belongs to one class or the other.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.9)$$

2.5.1.2 Deep Learning

DL algorithms learn features and model structure from unstructured data and self-improve without manual intervention. NNs are frameworks for DL and allow for the processing of complex data inputs. Typically, an NN consists of connected networks of *perceptrons* which simulate human neurons. The anatomy of a perceptron is shown in Figure 2.11.

The perceptron combines inputs x_i with weights w_i which modifies the effect of each input feature according to its predictive importance. The products are then summed and an *activation function* (§2.5.1.2) is applied which determines the extent to which the signal should progress through the network and its effect on the overall output, yielding an output of 1 or 0. A *layer* in a neural network is a series of perceptrons. Multiple layers can be stacked together, such that the output of one layer becomes the input of the next. The ultimate goal of a NN is to minimise the *loss function*, which is the calculable error rate of the task to which the NN has been applied. The weights (w_i) of the network are updated each iteration over the training data (epoch) in order

⁸ <https://www.nltk.org/>

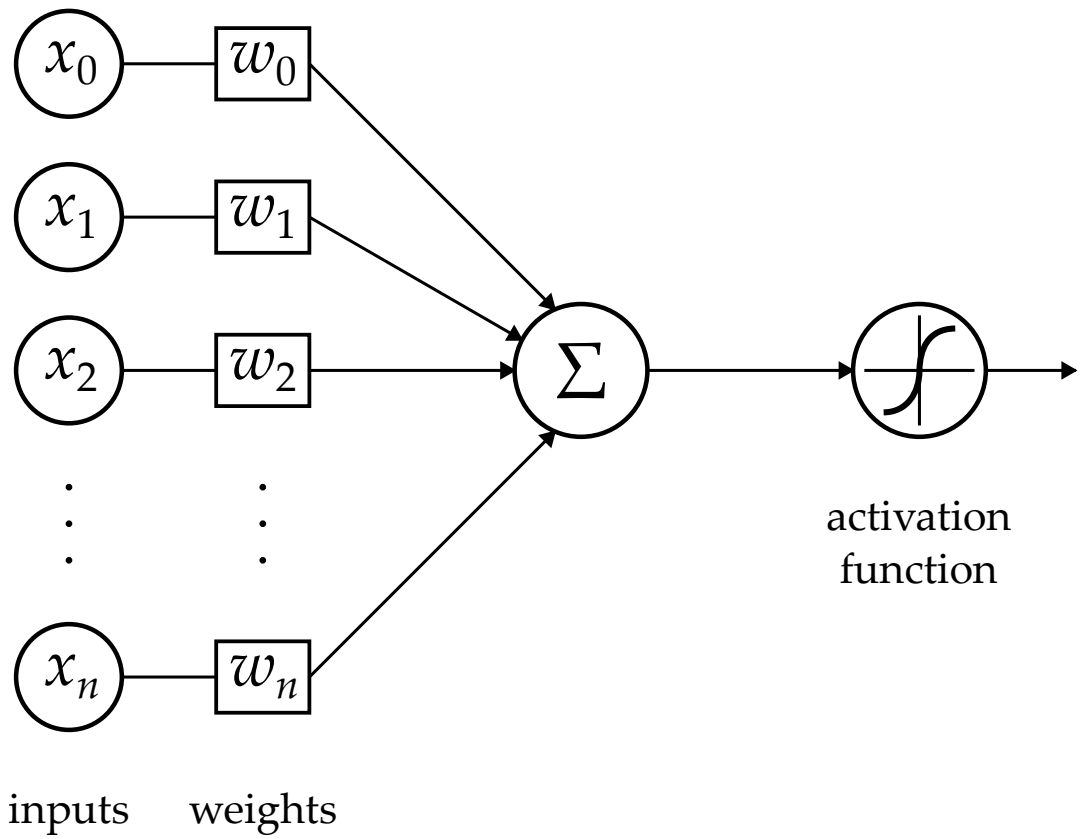


Figure 2.11: Diagram of a perceptron.

to minimise the loss function, and in this way the performance of the **NN** on the task improves over subsequent iterations.

ACTIVATION FUNCTIONS

Activation functions define the output of a perceptron given a set of inputs, and the choice of activation function in the hidden layers has an impact on the ability of the model to learn patterns from the input data. A variety of activation functions are used in **DL** (Nwankpa et al., 2018), four of which are visualised in Figure 2.12.

Each activation function is associated with specific strengths and weaknesses. For example, the output of *tanh* is centered about zero, which allows for output values to be mapped as strongly negative, neutral, or strongly positive. However, *tanh* suffers from the *vanishing gradient problem* (Kolen and Kremer, 2001); compared to inputs, the derivatives are small (between 0 and 1), which leads to updated weight values that are very similar to the previous values, hindering model convergence. Rectified Linear Unit (**ReLU**), on the other hand, counters the vanishing gradient problem with constant derivative for positive or negative inputs. However, **ReLU** suffers from the *dying ReLU problem* (Agarap, 2019); if most of the inputs are negative, outputs may continually output zeroes, resulting in poor model performance.

Typically, one activation function is chosen and used throughout the model as opposed to varying the function by layer. In contemporary **NLP** research, **ReLU** is usually chosen as the activation function (Goodfellow, Bengio, and Courville, 2016), although some models (for example, **RNNs** and **LSTMs**) use sigmoid for recurrent connections and *tanh* for output.

OUTPUT ACTIVATION FUNCTIONS

The final layer of an **NN** is the output layer which outputs a prediction. This thesis focuses on classification tasks (in which the task of the model is to predict one of a series of possible class labels for each input), where the output layer forms a probability distribution across the class labels, and the most likely class label is predicted.

For classification tasks, the choice of activation function is dependent on the number of classes: for binary classification tasks, the sigmoid activation function (Figure 2.12) is used, whereas the softmax activation function is used for multiclass classification tasks.

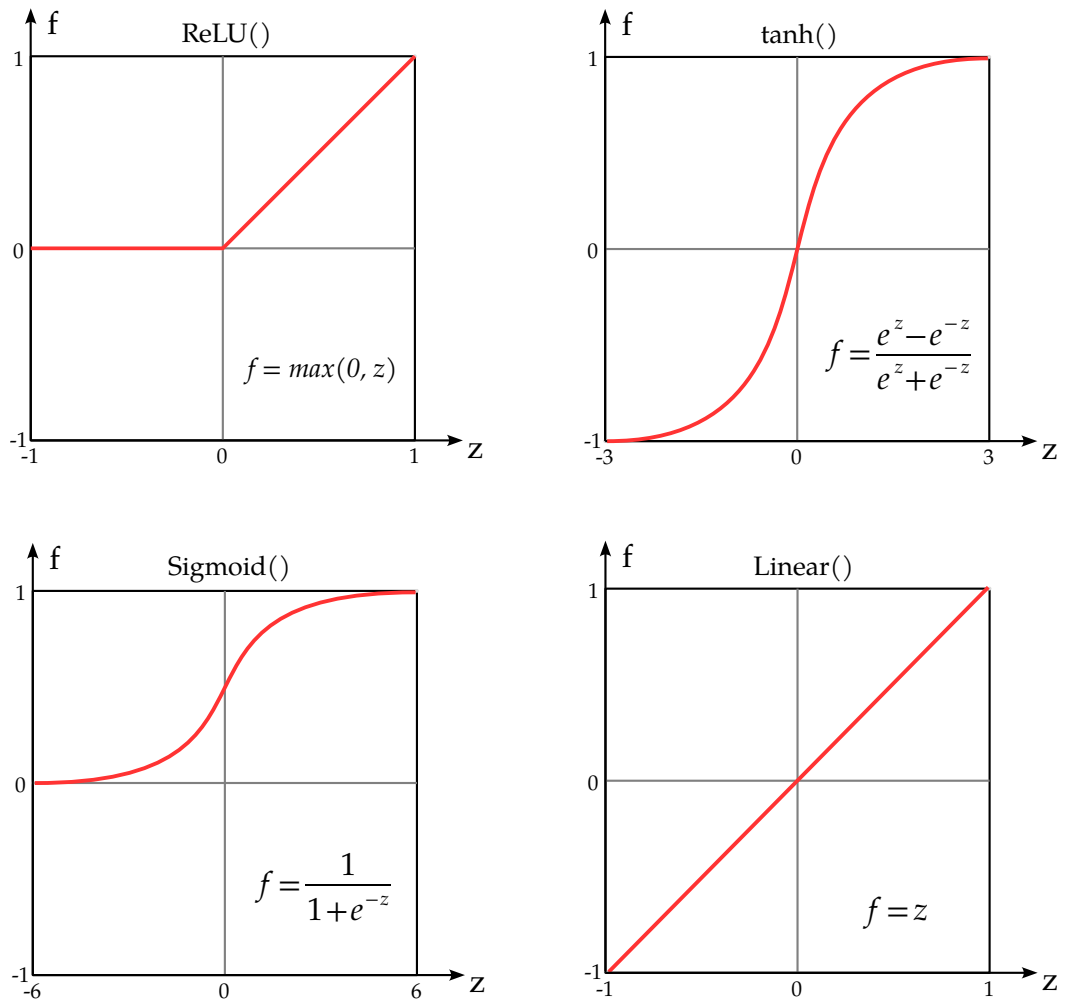


Figure 2.12: Visualisation of various activation functions. Adapted from <https://www.herongyang.com>.

Softmax is a generalisation of the sigmoid function for multiple classes, and converts an input vector of numbers into an output vector of probabilities proportional to the relative scale of each input vector value. An example of the softmax function applied to an input vector is shown in Equation 2.10.

$$\begin{array}{ccc}
 \begin{bmatrix} 1.2 \\ 0.9 \\ 0.6 \\ -0.3 \end{bmatrix} & \longrightarrow \sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \longrightarrow & \begin{bmatrix} 0.398 \\ 0.295 \\ 0.218 \\ 0.089 \end{bmatrix} \\
 \text{Input Vector} & \text{Softmax Function} & \text{Output Vector}
 \end{array} \tag{2.10}$$

MULTI-LAYER PERCEPTRONS

A Multi-Layer Perceptron (MLP) model is an example of an NN. Typically, an MLP is composed of at least three layers: a single input layer, one or more hidden layers, and an output layer, as shown in Figure 2.13.

The advantage of using an MLP over a single perceptron is that, when applied to classification tasks, they are able to distinguish classes that are non-linearly separable. There is no standard architecture for MLP models, and larger models (with larger and more numerous hidden layers) can typically interpret and classify more complex data, but are more computationally expensive to train.

RECURRENT NEURAL NETWORKS

RNNs are types of neural networks that are particularly suited to sequence-based tasks. Information from previous segments of the input sequence is used to influence later states.

A simple RNN is visualised in Figure 2.14, where the activation function \tanh is applied to the weighted input state x at time t combined with the weighted output state at time $t - 1$ to produce the output state at time t .

The formula for the activation function \tanh is shown in Equation 2.11, where W_x and W_h are the weights for the input neuron and recurrent neuron respectively.

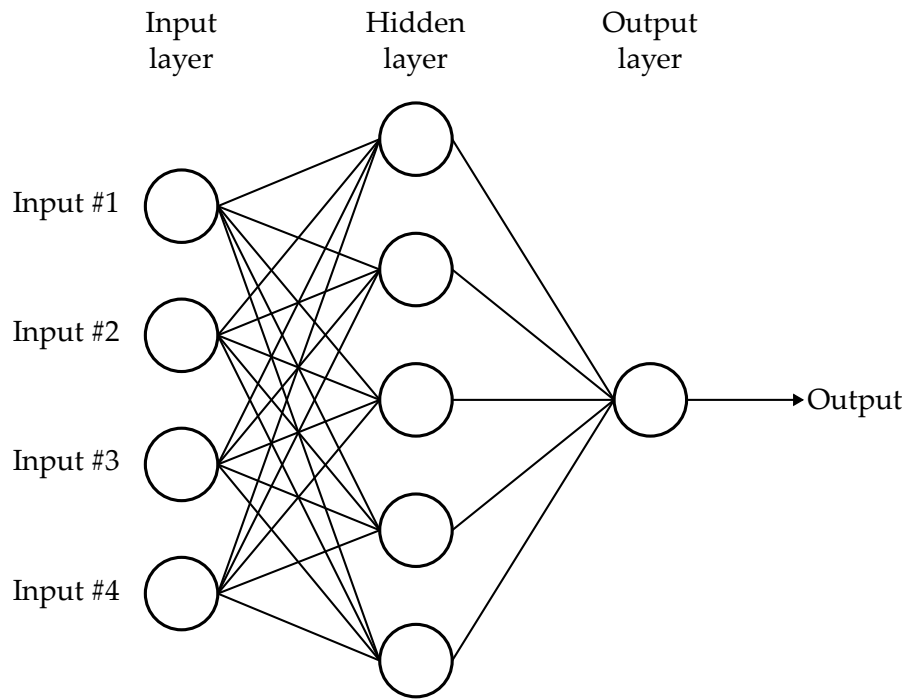


Figure 2.13: Diagram of a Multi-Layer Perceptron with one hidden layer.

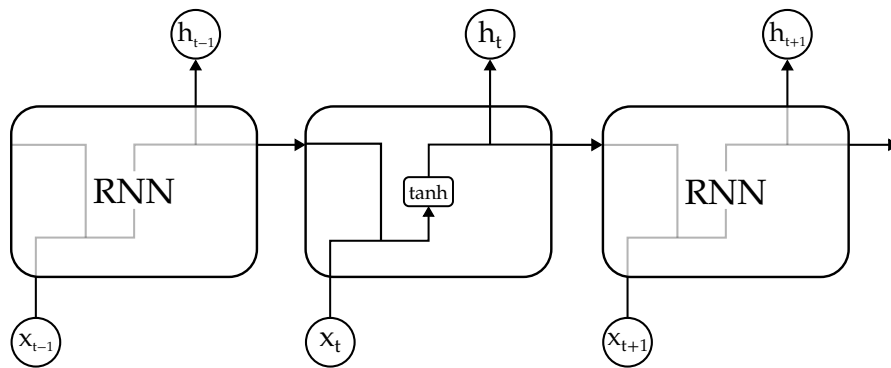


Figure 2.14: An example of a RNN. Diagram inspired by <http://colah.github.io>.

$$h_t = \tanh(W_x x_t + W_h h_{t-1}) \tag{2.11}$$

LONG-SHORT TERM MEMORY MODELS

Long Short-Term Memory (**LSTM**) models are a form of **RNN** that are better suited for capturing long-range dependencies. They contain *memory* that is able to store and output information, and *gates* that control incoming and outgoing information. These *gates* use the sigmoid (σ) activation function, which allows the network to selectively retain information.

The structure of an **LSTM** cell is visualised in Figure 2.15.

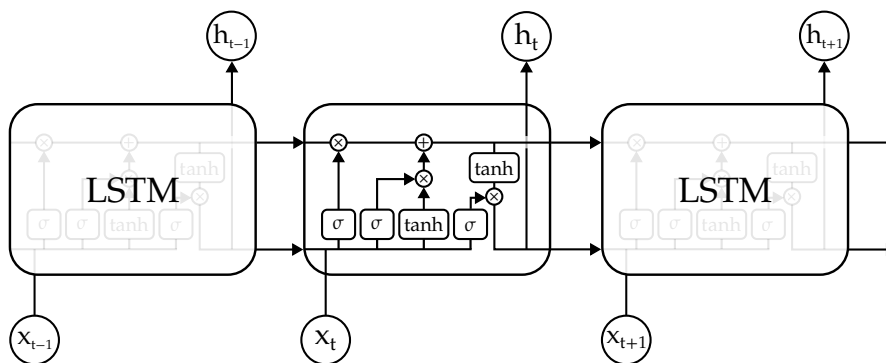


Figure 2.15: An example of a **LSTM**. Diagram inspired by <http://colah.github.io>.

Bidirectional Long Short-Term Memory (**BiLSTM**) models feature two layers of **LSTM** cells where the sequence direction of the first is mirrored in the second. This allows the model to capture sequential dependencies in both directions, and typically **BiLSTM** models outperform **LSTM** models at structured prediction tasks (Z. Huang, Xu, and K. Yu, 2015).

GATED RECURRENT UNITS

Gated Recurrent Unit models are functionally simplified versions of **LSTMs**. Compared to the three gates in **LSTMs**, **GRUs** contain a single *update gate* which determines how much data should be passed on to subsequent cells, and a *reset gate* which determines how much data should be forgotten.

Generally, **GRU** models are simpler to implement and train faster than **LSTM** models, but are less able to capture long-term dependencies and tend to yield inferior results.

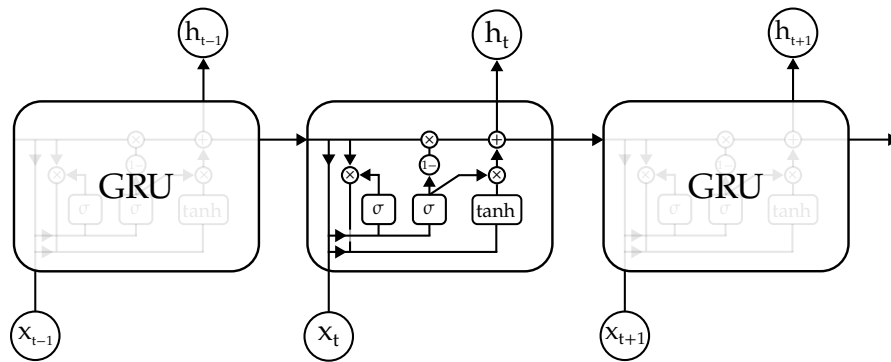


Figure 2.16: An example of a GRU. Diagram inspired by <http://colah.github.io>.

Identical in principle to the augmentation from LSTM to BiLSTM, Bidirectional Gated Recurrent Units (BiGRUs) are models with two layers of GRU cells where the sequence direction of the first is mirrored in the second, which capture structural dependencies in both directions and typically outperform their unidirectional counterparts.

CONVOLUTIONAL NEURAL NETWORKS

A CNN is an example of an MLP that contains *convolution* and *pooling* layers. *Convolution* is the process whereby two functions are combined to form one function which is an expression of how the shape of one is modified by the other, and a *convolution layer* uses this process rather than general matrix multiplication. Since convolution layers take advantage of the spatial coherence of the input, they are able to reduce the number of required parameters by sharing weights. ‘Pooling’ layers essentially compress the output of the previous layer while still preserving enough salient information, and are often placed directly after convolution layers. Figure 2.17 shows an example architecture of a simple CNN.

When applied to ML tasks, effective CNN models are often very large and computationally more expensive to train than MLP models.

TRANSFORMERS

The Transformer model, proposed by Vaswani et al. (2017), is an NN that uses *attention* to increase both model performance and the speed at which the model trains. Essentially, a

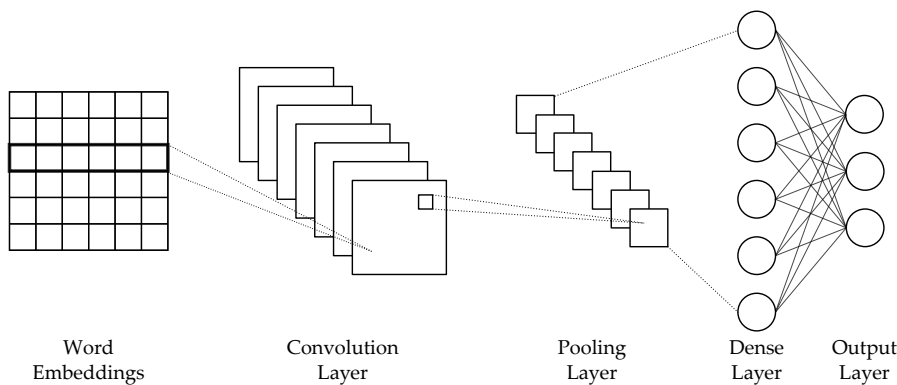


Figure 2.17: An example of a CNN. In this model, the convolution layer is followed by a pooling layer.

transformer model contains a stack of encoder components with connections to a stack of an equal number of decoder components. Each encoder and decoder component contains an *attention* layer, which allows the model to consider other positions in the input sequence when generating an encoding of the current item. In this way, the transformer model incorporates information about the most relevant other words in an input sequence (for example, which entity the pronoun *it* refers to in a sentence).

Due to their improved training times and relatively superior performance versus RNN and CNN models on NLP tasks, transformers are prevalent in the field and have been used to great effect in many tasks.

BERT (Devlin et al., 2019) and Generative Pretrained Transformer (GPT) (Radford and Narasimhan, 2018) are two examples of transformer models. Open-source pretrained models are available for academic use⁹ and can be downloaded and fine-tuned to a particular language task.

2.5.2 Model Training

In order to train a machine learning model, a **training set** is required, consisting of:

- **Training Data**; containing the data to be analysed and used in decision-making; for example, the text in an email

⁹ <https://huggingface.co/>

- **Training Labels;** containing the correct identity of each item in the training data; for example, whether the email is *spam* or *not spam*

In addition to the model architecture and training set, there are a number of parameters associated with the model that can be altered to modify the way in which the model trains, referred to as *hyperparameters*:

- **Layer Configuration;** in a neural model, the number of hidden layers and the number of neurons in each layer can be increased which may improve model performance at the cost of computational complexity.
- **Learning Rate;** the value which modifies the magnitude of each update in the learning process. If the learning rate is small, the model will converge slowly, but if set too large, the model may converge quickly to a suboptimal solution.
- **Number of Epochs;** the number of passes over the training data made by the model. If the number of epochs is too low, the model may not be optimised, but if too large, the model may *overfit* and suffer reduced performance on unseen data.
- **Batch Size;** the amount of training data passed to the network at one time. If the batch size is too small, the model may be slow to converge, but if too large, the model may suffer from poor generalisation (Wilson and Martinez, 2003).

There is little insight in the literature as to how the values for these hyperparameters should be selected. A common approach is to test different sets of hyperparameter values at a time, and values associated with better-performing values are used in subsequent tests. This process can be performed automatically, or an automatic parameter selection method can be employed (for example, random search).

2.5.3 Model Validation

During each epoch of training, the neural network internal weights and biases are updated to improve the performance of the model with respect to the training data. After each epoch, the network *validates* by reviewing the extent to which the model is learning. This validation step is performed using the validation set, which is a separate set of data from the training set, consisting of validation data and validation labels. It

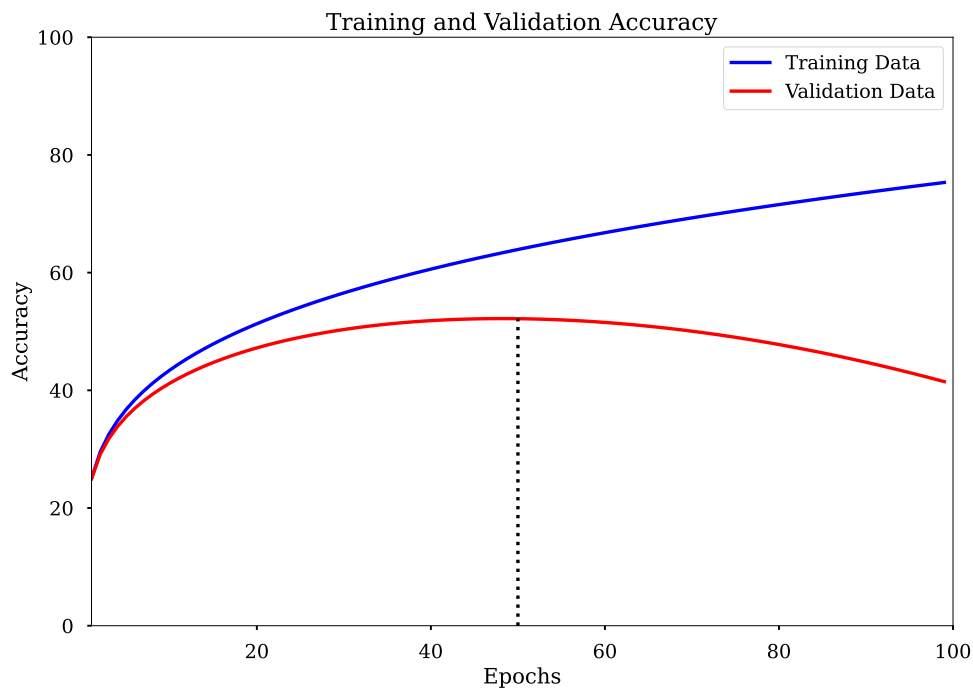


Figure 2.18: Example plot of accuracy over several epochs. Overfitting occurs after 50 epochs.

is important that this data is only accessed by the model during the validation step (in that it is not seen during training), since the purpose of model validation is to simulate model performance on novel data. The model makes predictions using this validation data and compares the predicted labels to the validation labels using the loss function, which produces a metric of error that can be used to evaluate the network. In addition to the loss, the accuracy of the model is often computed as a measure of model performance. Ideally, the model is able to learn salient features of the training data over multiple epochs, and validation accuracy will increase as a result. *Overfitting* occurs when validation accuracy *decreases* after reaching a local maximum because the model maximises performance on the training data at the cost of generalisability. A visual example of overfitting is shown in Figure 2.18.

Two main ways of preventing overfitting are *early stopping* and *dropout*. Early stopping refers to the automatic termination of model training when validation accuracy starts to decrease, even when the maximum specified epoch limit has not yet been reached.

Dropout refers to ignoring a specified proportion of neurons in a **NN** during training. By doing so, the **NN** is forced to utilise all available neurons, as opposed to just a few, and prevents the **NN** from becoming too dependent on any particular set of features in the input data.

2.5.4 Model Evaluation

The process of evaluating a model is similar to the validation step; unseen test data is given as input to the model, and predicted labels are compared against the true labels. However, unlike model validation, testing occurs once at the end of model training.

There are a number of metrics that can be used to evaluate model performance, including accuracy, precision, recall, F_1 , Receiver Operating Characteristic (**ROC**) and Area Under the ROC Curve (**AUC**). When evaluating classification tasks (i.e. where the task is to predict pre-defined labels for a set of items), it is common to generate a *confusion matrix* of predicted labels versus correct labels, as shown in Table 2.1.

		Predicted Label	
		Positive	Negative
True Label	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Confusion Matrix to Visualise Prediction Success and Error Types

2.5.4.1 Accuracy

Accuracy, shown in Equation 2.12, represents the proportion of correctly classified items.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (2.12)$$

Although accuracy is a commonly reported metric, it is not suitable in cases where true class labels are unbalanced; a model that predicts class 1 for every item will yield

an accuracy of .9 if the evaluation set contains 90% items with class 1. Accuracy is often reported alongside metrics such as precision, recall, and F_1 score.

2.5.4.2 Precision

Precision, shown in Equation 2.13, represents the proportion of correct predictions out of all instances where a given class was predicted.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.13)$$

A model with a high precision makes few False Positive errors. It is often reported alongside recall, which shows the proportion of False Negative errors, and F_1 , a type of average between precision and recall.

Mean Average Precision (MAP) $@k$ is an extension of precision typically used for evaluating recommendation tasks. ‘Precision $@k$ ’ is the fraction of relevant items in the top k recommended results, and ‘Average Precision $@k$ ’ is the sum of precision $@k$ divided by the total number of relevant items in the top k results. MAP $@k$ is the mean of the Average Precision $@k$ over the entire dataset. The advantage of MAP $@k$ for recommendation tasks is that it not only indicates whether predicted recommendations are *relevant*, but also whether the most relevant recommendations are recommended *preferentially*.

2.5.4.3 Recall

Recall, shown in Equation 2.14, represents the proportion of correct predictions out of all instances of a given class.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.14)$$

A model with high recall makes few False Negative errors. It is often reported alongside precision, which shows the proportion of False Positive errors, and F_1 , a type of average between precision and recall.

2.5.4.4 F_1

F_1 score, shown in Equation 2.15, is the harmonic mean of precision and recall. It is a type of average between the two metrics that penalises low values of either.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.15)$$

F_1 is a commonly used metric for evaluating classification models since it is sensitive to low values of either precision or recall. Models with a high F_1 score tend to have both high precision and high recall.

In multi-class classification tasks (i.e. where there are more than two possible classes in the data), F_1 across all classes can be averaged using *macro*, *micro*, or *weighted* methods. *Macro* averaging is the unweighted mean of F_1 scores; after F_1 is calculated for each class, the average of F_1 scores is the *macro* average. In this way, it is insensitive to class imbalance, and treats all classes equally regardless of the number of cases in the evaluation data. *Micro* averaging is performed by computing F_1 on the global level (as opposed to the class level). In multi-class classification tasks, micro F_1 is equivalent to the accuracy. *Weighted* averaging is calculated by multiplying the per-class F_1 score by the relevant proportion of data in the evaluation set. In this way, it is sensitive to class imbalance, and gives higher weighting to F_1 scores for classes that were more prevalent in the evaluation data.

2.6 INTRODUCTION TO RECOMMENDER SYSTEMS

As mentioned in Chapter 1, a common approach to addressing the *information overload* (Dhameliya and Desai, 2019) or *filter failure* (Shirky, 2008) problems is through the use of recommender systems, which are automatic methods of generating personalised item recommendations to a user, often utilising a combination of user engagement history and item popularity statistics. Generally, approaches to developing recommender systems fall into one of the following distinct categories based on the aspect of data that inference is drawn from: content-based filtering (§2.6.1), which generates recommendations based on the similarity of new items to the items that the user has previously interacted with; collaborative filtering (§2.6.2), which generates

recommendations based on the interactions of similar users; knowledge-based filtering (§2.6.3), which generates recommendations based on inferred connection between the user and recommended items; and hybrid systems (§2.6.4), which combine one or more of the aforementioned approaches in an ensemble system.

2.6.1 Content-Based Filtering

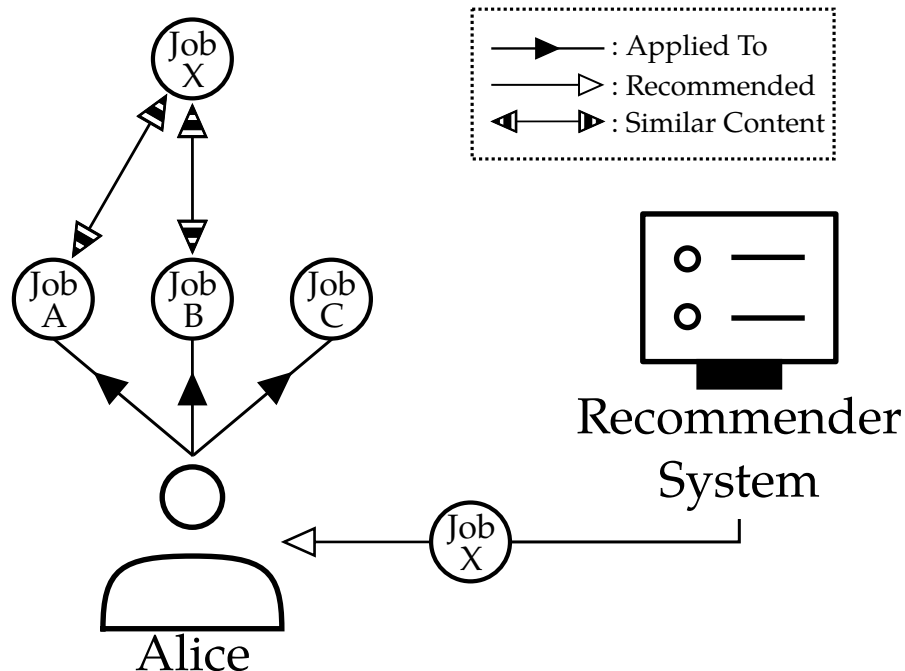
Content-based filtering approaches, visualised in Figure 2.19, recommend items with similar content to the content the user has previously engaged with (Mooney and Roy, 2000).

It is worth noting that related work into the specific application of content-based filtering for job recommendation often conflates the content-based approach with knowledge-based filtering (Tripathi, Agarwal, and Vashishtha, 2016), which involves recommending *items with similar content to the user*, which is extracted either from user data or queried through a series of prompts. In this thesis, we treat these as distinct approaches, since each is associated with a unique set of advantages and disadvantages. To clarify this distinction, content-based filtering can be summarised as *item-item* filtering, in that similarities between items are inferred, but we do not infer direct connection between user and item; conversely, knowledge-based filtering can be summarised as *user-item* filtering, in that connections between users and items are inferred, but we do not infer connection between different items.

The two key components of a content-based filtering system are the *user profile*, which represents the features of the user after extracting information about their engagement with the recommender system, and the *item representation*, which contains the information about each item that can be used for calculating similarity to the user's preference.

One advantage of the content-based filtering approach to building job recommender systems is that it works well with implicit feedback when explicit rating is difficult (Al-Otaibi and Ykhlef, 2012). Although users may not be able to rate the jobs they are applying for, there are other observable measures of engagement such as whether the user clicks on a job description on the online portal (presumably to learn more about the listing), whether they submit an application, or whether a recruiter tags a candidate to a job (Nigam et al., 2019). Furthermore, as the user profile expands

Figure 2.19: The content-based filtering job recommendation system uses a job's attributes to recommend jobs that have similar attributes to those Alice has applied to in the past (adapted from Almalis, Tsihrintzis, Karagiannis, and Strati (2016)).



through prolonged engagement with the system, the quality of the system improves given that the available data from which the model can draw inference is continually augmented through user engagement (Al-Otaibi and Ykhlef, 2012).

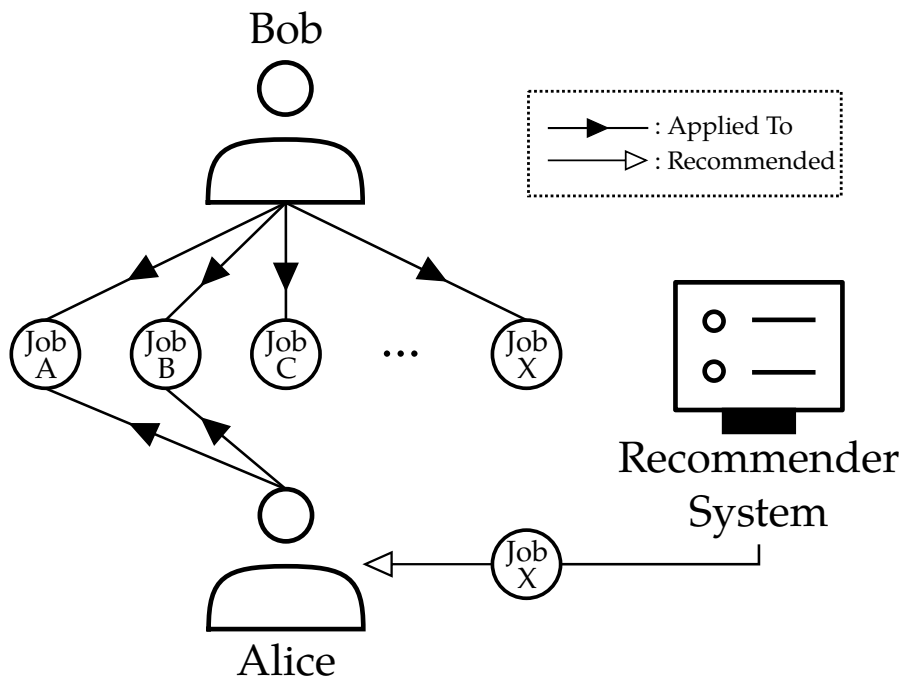
A disadvantage of the content-based filtering approach is *overspecialisation*, which refers to cases where users receive recommendations that are too similar to their user profile and do not receive recommendations that are diverse. This may be an issue for users who seek a job that is a side-step from their current and previous roles, as the system will show only the jobs that are most similar to their profile.

The *cold-start problem* affects content-based filtering systems as they rely entirely on the user's engagement with the system. Because these systems require user-item interactions before making recommendations, they are of limited use for new users for which the system has no developed user profile (Yuan et al., 2016). Furthermore, it is difficult to identify the salient features driving the user's decision to engage with a particular item which should be used to recommend similar items.

2.6.2 Collaborative Filtering

Collaborative filtering approaches, visualised in Figure 2.20, recommend items based on what similar users have previously preferred (Rafter, Bradley, and Smyth, 2000).

Figure 2.20: Bob applied to jobs A, B, C, and X. Alice applied to A and B. The collaborative filtering job recommendation system will recommend job X to Alice since her choices are similar (although not identical) to those of Bob (adapted from Almalis, Tsihrintzis, Karagiannis, and Strati (2016)).



The assumption of the collaborative filtering approach is that users who rate items similarly will have similar preferences in the future. A collaborative filtering job recommender system creates a *rating matrix* by combining the preferences of all its users (based on their engagement with the system), and converts this to a *prediction matrix* which is used to produce recommendations. The method by which the prediction matrix is created further classifies systems using the collaborative filtering approach; *memory based* algorithms compute predictions using the rating matrix directly on the basis of similarity measures such as the Pearson Correlation coefficient (Freedman, Pisani, and Purves, 2007) or cosine similarity, whereas *model based* algorithms compute predictions by reducing the dimensionality of the rating matrix through Matrix Factori-

sation (MF) (Koren, Bell, and Volinsky, 2009) or Singular Value Decomposition (SVD) (Wall, Rechtsteiner, and Rocha (2003); Parhi, Pal, and M. Aggarwal (2017)).

Advantages of job recommender systems using the collaborative filtering approach with memory-based algorithms for generating predictions include the simplicity of system architecture (given that calculating the similarity metric is the only step between collating the rating matrix and returning the ranked prediction matrix) and the fact that there is no need to generate profiles for job descriptions since similarity is calculated between users and not job descriptions. However, since similarity metrics are calculated using all the available user data to generate predictions, these systems can be slow when datasets are large, and scale poorly.

Advantages of job recommender systems using the collaborative filtering approach with model-based algorithms for generating predictions include the ability to be scaled easily. Since dimension reduction is performed to create the model for prediction generation, large pools of user data do not slow these systems, which makes them an appropriate choice for real time applications. However, creating the model is a complex and computationally expensive process that requires careful parameter tuning for optimisation (Dhameliya and Desai, 2019).

A disadvantage of both memory-based and model-based collaborative filtering systems is that they tend to suffer from the *cold-start* problem, in that the system generates poorer recommendations for new users with little engagement on the platform, and new job descriptions featured on the system will not be recommended to users because they have no user engagement associated with them.

2.6.2.1 Matrix Factorisation

MF is the process by which a matrix is decomposed into two or more smaller matrices. In the context of collaborative filtering, MF is used to decompose a user-job interaction matrix \mathbf{M} into two matrices: a user matrix \mathbf{U} and a job matrix \mathbf{V} . A more robust model of interaction may include a global mean μ to center matrix \mathbf{M} , and bias terms \mathbf{b}_U and \mathbf{b}_V for users and jobs respectively. The MF equation is shown in Equation 2.16.

$$\mathbf{M} \approx \mathbf{U} \cdot \mathbf{V}^T + \mu + \mathbf{b}_U + \mathbf{b}_V \quad (2.16)$$

The user matrix represents the user's preferences for different features of the jobs, while the job matrix represents the features of the jobs themselves. The dot product of the user and job matrices combined with the global mean and bias terms yields the predicted interaction for each user-job pair.

Matrices \mathbf{U} and \mathbf{V} are fitted by initialising them with random numbers and iteratively updating them until convergence is reached.

SINGULAR VALUE DECOMPOSITION

SVD is a variant of **MF**. The process of **SVD** decomposes a matrix into three matrices as shown in Equation 2.17, where \mathbf{M} is the initial matrix, \mathbf{U} is a matrix representing the left singular vectors of matrix \mathbf{M} , $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values of matrix \mathbf{M} , and \mathbf{V}^T is a matrix representing the right singular vectors of the original matrix.

$$\mathbf{M} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^T \quad (2.17)$$

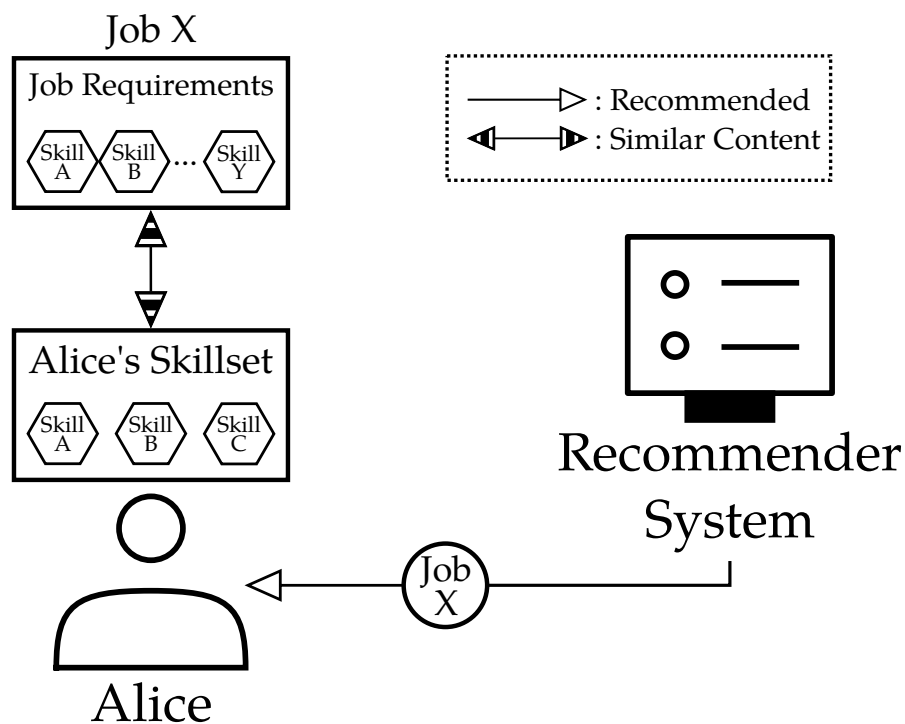
In the field of job recommender systems, **SVD** is used as a collaborative filtering method and has been shown to be particularly effective when applied to recommendation tasks (Wall, Rechtsteiner, and Rocha (2003); Parhi, Pal, and M. Aggarwal (2017)). **SVD** involves decomposing the user-job interaction matrix into its constituent parts, and then using these parts to make predictions about the user's interactions with new jobs. The user-job interaction matrix is a matrix where each row represents a user, each column represents a job, and each cell contains the interaction between the user and job (for example, one of {Applied; Did Not Apply}).

After decomposing the user-job interaction matrix using **SVD**, the resultant matrices \mathbf{U} , $\mathbf{\Sigma}$, and \mathbf{V}^T can be used to predict the user's interaction with a new item as follows: the user's row in matrix \mathbf{U} is multiplied by the diagonal matrix $\mathbf{\Sigma}$ to create a vector of weights. This weight vector is multiplied by the relevant job column in the \mathbf{V}^T matrix to generate a predicted interaction term for the job.

2.6.3 Knowledge-Based Filtering

Knowledge-based filtering approaches, visualised in Figure 2.21, recommend items based on inferences about the user's suitability and preferences which are extracted from user data or queried through a series of prompts.

Figure 2.21: Knowledge-based filtering job recommendation systems recommend jobs to Alice by comparing the attributes of Alice (for example, the skills in their skillset) with the attributes of the jobs (for example, the skills the job requires).



Job recommender systems that use a knowledge-based filtering approach are distinct from other approaches in that they have *functional knowledge* (Burke, 2002); they have knowledge about how a particular job meets a particular user's need and can reason about the relationship between a suitability or preference and a possible recommendation. Ramezani et al. (2008) note three ways in which systems reason in this way: conversational case-based reasoning systems elicit a job seeker's query incrementally through an interactive dialogue with the user. Constraint-based reasoning systems provide recommendations based on a set of constraints gathered

from the user. Rule-based systems provide recommendations to users using explicit rules that map between the user's needs and job positions. However, as discussed in §2.6.1, there is an additional method of knowledge-based filtering which related literature tends to classify as *content-based filtering*, and that is the method of providing recommendations to users based on connections that can be inferred from the properties associated with the user and the properties associated with the items for recommendation.

In knowledge-based filtering approaches, the user profile may be any knowledge structure that supports this inference, such as the set of skills associated with a user extracted from their CV (shown in Figure 2.21), demographics, engagement with jobs on an online portal (for example, jobs clicked on, jobs applied to), or ontologies about job categories and hiring company information (Lee and Brusilovsky, 2007).

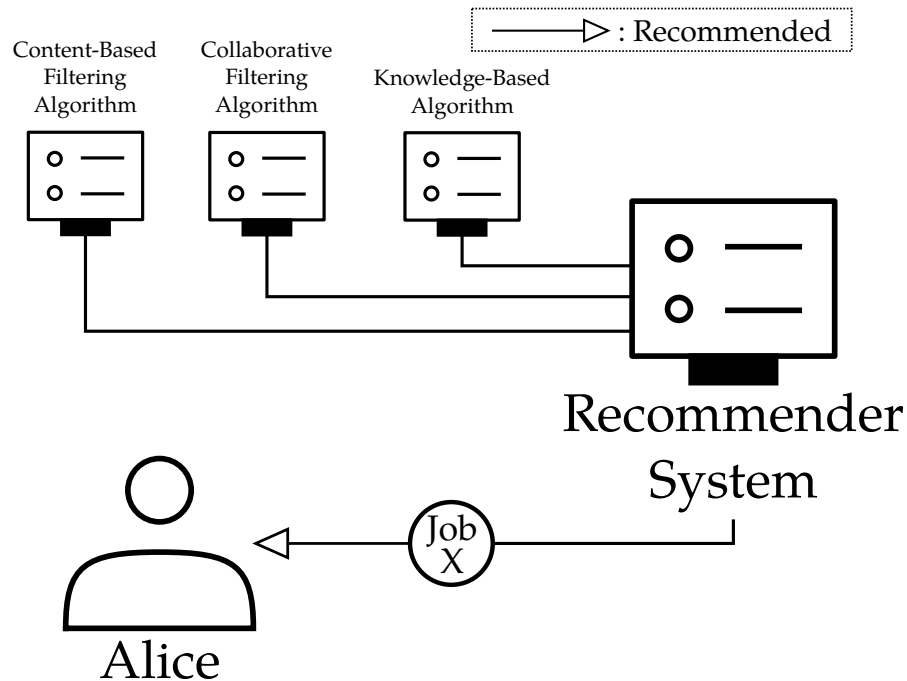
An advantage of knowledge-based filtering is that systems using this approach tend to avoid the cold-start problem which limits the content-based and collaborative filtering approaches. Since users are usually required to give explicit information regarding their preferences before using the service (or provide a source that they can be extracted from, such as a CV), informed recommendations can be made immediately. However, this requires a knowledge acquisition process which needs to be carefully designed to extract meaningful insights from user-provided data (for example, extracting salient entities from candidate CVs) in order to be able to reason about the relationship between a user's suitability for various jobs and generated recommendations.

2.6.4 Hybrid Filtering

Hybrid approaches, visualised in Figure 2.22, use a combination of collaborative filtering, content-based filtering, and knowledge-based filtering methods.

Job recommender systems utilising a hybrid approach tend to combine content-based filtering methods with either collaborative filtering methods or knowledge-based methods (Dhameliya and Desai, 2019), and can be further classified depending on the hybridization method. This may involve: weighting different methods before combining them; switching between methods depending on context; or running one method directly after another using the output of one method to influence the other (Burke, 2002).

Figure 2.22: Hybrid recommendation systems combine two or more techniques to provide better recommendations to Alice (Almalis, Tsihrintzis, Karagiannis, and Strati, 2016).



Since hybrid methods combine various other approaches, they tend to share some of the advantages and disadvantages of their component methods but overcome some of the issues associated with single approaches. For example, content-based recommender systems may suffer from *overspecialisation* in that users may receive job recommendations that are too similar to their user profile. This can be overcome by combining the content-based approach with the collaborative filtering approach in a hybrid system, so users are also shown jobs that may be less similar to their own profile but found to be good recommendations for similar users.

2.7 EVALUATING HUMAN ANNOTATIONS

Since many NLP tasks aim to replicate human behaviour, it is common practice in the field to develop and train models on datasets that have been annotated manually by human agents according to a specific schema or adhering to a set of rules. When sourcing annotations, it may be necessary to evaluate individual annotator's contributions; it

is not uncommon for annotations to be of low quality as a consequence of poor task comprehension, poor compensation, or the absence of a feedback mechanism that penalises poor quality contribution. Methods of evaluation are necessary to ensure that annotations are representative of human ability. There are two main facets of human annotation evaluation: accuracy versus a gold standard; and IAA.

2.7.1 Accuracy versus Gold Standard

Accuracy is calculated in the same way as described in §2.5.4.1. Although this is an informative metric to evaluate the competence of individual annotators, it is only calculable when a gold standard has been defined and established. For many complex tasks for which automatic NLP solutions are sought, it is not feasible to create a gold standard until the task has sufficiently been defined and communicated to annotators, and annotators can be shown to generally agree on the correct classification of items. In these instances, IAA is a more informative metric than accuracy versus a gold standard.

2.7.2 Inter-Annotator Agreement

IAA refers to the extent to which annotators agree with one another. There are several ways to measure IAA, including Cohen's κ , Fleiss' κ , and F_1 .

Cohen's κ is calculated as shown in equation 2.18, where p_o is the relative observed agreement among two raters, and p_e is the hypothetical probability of chance agreement. Since Cohen's κ can only be calculated between two raters, Cohen's κ can be calculated between each annotator pair and then averaged to show agreement between multiple raters (referred to as *pairwise* κ), or Fleiss' κ can be used, which is shown in equation 2.19, where \bar{P} is the mean of relative observed agreement and \bar{P}_e is the mean of hypothetical probabilities of chance agreement. In cases where there are multiple raters and each item is annotated by exactly two independent raters, pairwise Cohen's κ and Fleiss' κ are equivalent.

$$\text{Cohen's } \kappa = \frac{p_o - p_e}{1 - p_e} \quad (2.18)$$

$$\text{Fleiss' } \kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \quad (2.19)$$

κ ranges between 0 and 1, and higher values of κ statistics indicate greater agreement between annotators. Landis and Koch (1977) put forward a widely-accepted interpretation of the κ statistic which is summarised in Table 2.2.

Kappa (κ)	Interpretation
< 0	Poor
0.01 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost perfect

Table 2.2: Interpretation of Fleiss' Kappa (Landis and Koch, 1977).

Krippendorff's α (Hayes and Krippendorff, 2007), given by equation 2.20, is an alternative measurement for IAA that addresses multiple categories, as in the case of ER tasks. Additionally, the α statistic ignores incomplete or missing annotations, which is suitable when individual items have only been annotated by a subset of of the annotator pool. However, the α statistic is particularly sensitive to extreme disagreement, and there is no academic consensus regarding its interpretation, and for this reason it is often reported in conjunction with the κ statistic.

$$\alpha = 1 - \frac{\text{observed disagreement}}{\text{expected disagreement}} \quad (2.20)$$

2.8 EQUALITY, DIVERSITY, AND INCLUSION IN RECRUITMENT ALGORITHMS

EDI refers to the principles and policies that ensure fair treatment and opportunities for all. When designing automatic solutions that have an impact on people, in this case job seekers and recruitment personnel, it is imperative that active steps are taken to ensure that biases are addressed and mitigated.

The hiring process has long been fraught with bias and discrimination which still persists (Quillian et al., 2017). Bias can be based on race, gender, age, disability, and other characteristics which may limit the employment opportunities for historically excluded groups.

Institutional bias occurs when hiring processes unfairly advantage or disadvantage certain workers. For example, under the guise of culture fit, companies may be selectively hiring from a homogeneous pool of privileged individuals and rejecting equally capable candidates with diverse backgrounds.

Systemic bias occurs when prejudices are embedded within the hiring process, disadvantaging those from historically excluded groups. For example, by only considering applicants from top universities (that typically intake high proportions of privileged individuals), companies may reject capable candidates that did not have access to sufficient education resources prior to university application.

Automation bias occurs when individuals give undue weight to information that is perceived to have come from an algorithm. This is particularly important to consider when designing algorithms that influence human decision-making; any recommendation given by a model must be explicable in understandable terms to the user so that they may consider it with an appropriate level of importance.

One particular high-profile example of bias in hiring algorithms is that of the discontinued AI recruiting tool developed by Amazon which was shown to be biased against women (Dastin, 2018). The model was trained on applications made to the company over the previous 10 years which were particularly male-dominated, reflecting patterns across the technology domain. As a consequence of this biased training data, terms such as *women's* (for example, *women's chess club captain*) negatively affected the likelihood of application success when using the recruiting tool. A further example of bias in hiring algorithms is that of an undisclosed company selling a CV screening tool that gave disproportionately strong weighting to features in the text that should not have

been considered, such as the forename *Jared* or whether the candidate played *high school lacrosse* (Gershgorn, 2018).

It is important to note that, without active measures in place to mitigate them, biases will arise in recruitment algorithms by default (Bogen and Rieke, 2018). Furthermore, although an essential process, it is not enough to just remove or obscure sensitive characteristics of users such as gender and ethnicity, since this will not prevent models from reflecting patterns of bias.

In order to develop recruitment algorithms with EDI in mind, the following bias mitigation methods should be considered: removing all sensitive characteristics from input data; investigating associations and correlations between sensitive characteristics removed from input data and the balance of data; and ensuring that each component of a modular job recommendation system is transparent and explicable in function, so any output can be queried by a user who wants to know why any given recommendation was made.

Methods such as feature importance analysis can be used to provide insight into the relative significance of different elements or criteria in job recommendation algorithms. For TML models such as Logistic Regression, this process is relatively straightforward as learned feature importance weights are readily accessible and interpretable. However, this process is considerably more difficult with DL approaches as levels of abstraction preclude simple inference of relative feature importance. Model interpretability is an ongoing research problem in the field of NLP, and libraries have been developed to facilitate human understanding of NNs (Kokhlikyan et al., 2020) and attention-based models such as BERT (Chefer, Gur, and Wolf, 2021).

RELATED WORK

3.1 OVERVIEW

This chapter describes related work to the field of the thesis. First we describe research into job recommendation systems in §3.2, detailing two open challenges *Recsys 2016* and the *Kaggle Job Recommendation Challenge*, and we organise previous literature by general approach. We then describe research in Natural Language Processing (NLP) that has influenced the direction of this thesis in §3.4, specifically in word embeddings, Entity Recognition (ER), and Natural Language Inference (NLI).

3.2 JOB RECOMMENDATION SYSTEMS

A common approach to solving the information overload problem (explained in more detail in §1.1) in the field of applicant profile-job description matching is through the use of Job Recommender Systems (Gugnani and Misra (2020); S. Yang et al. (2017); Özcan and Öguducu (2017)). A recommender system is a tool that generates recommendations of items for a user, often without an explicit set of criteria from the user. A more detailed explanation of recommender systems can be found in §2.6. Figure 3.1 shows the basic architecture of a job recommender system with relevant sections of this chapter shown in parentheses.

In the past decade, recommender systems have been used to great effect in several domains (Ricci et al., 2010). Notable examples of commercial recommender systems include Netflix¹ for recommending films and television programmes to a viewer (Bennett and Lanning, 2007) and Amazon² for recommending products to online shoppers (Linden, B. Smith, and York, 2003).

¹ <https://netflix.com>

² <https://amazon.co.uk>

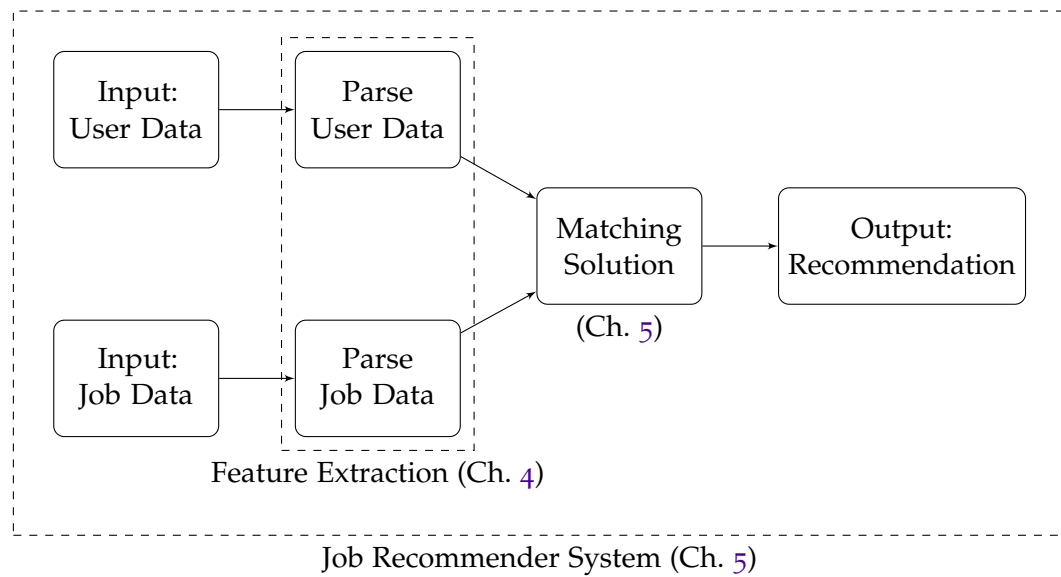


Figure 3.1: Architecture of a generic matching system. Relevant sections of the related work are shown in parentheses.

Job recommender systems, such as the system integrated with employment-focused social media platform LinkedIn³, are similar to generic recommendation systems where job seekers are the *users* of the system and the job listings are the *items* for recommendation. Although there are many similarities between generic recommender systems and job recommender systems, modifications to system design are required in order to overcome field-specific challenges. Dhameliya and Desai (2019) note three such challenges: reduced feedback mechanisms, timeliness, and bilateral matching.

Reduced feedback mechanisms refers to the limited opportunities for which job recommender systems are able to receive feedback on generated recommendations compared to standard recommender systems. For example, while shoppers on Amazon may leave a written review for a purchased product along with a rating on a five-star scale, job seekers using online portals are often not able to rate the jobs to which they have submitted their applications, resulting in reduced feedback mechanisms. To address this, alternative measures of feedback are required, often implied through the user's engagement with the online platform. Sections 3.2.3 through 3.2.6 discuss alternative measures of feedback in more detail.

³ <https://www.linkedin.com/>

The second challenge of job recommender systems is the issue of *timeliness*, which refers to the transient nature of job listings; while products on Amazon may be listed indefinitely, job listings tend to have a relatively short lifespan, and will exist on a platform for as long as the recruiting agents dictate. This may be as short as a few days for time-sensitive roles. In the field of e-commerce, customers with similar preferences to another customer can be identified by finding comparisons between the products each customer has liked or bought. In this way, recommendations can be based on previous matches between the product and the customer. If customers *A* and *B* have bought similar sets of products, and customer *A* then buys product *X*, then *X* may be a good recommendation for customer *B*. This approach, known as collaborative filtering, is discussed in more detail in §3.2.4. However, in many cases this cannot be directly applied to the field of job recommendation due to the issue of timeliness; for example, at the time *B* actively seeks job recommendations, job *X* may no longer be available on the online portal. To overcome this challenge, job recommendation systems may need to utilise data from historical, inactive job listings and compare and apply this data to live job listings in order to generate effective recommendations for active jobs.

The third challenge noted by Dhameliya and Desai (2019) is that of the bilateral matching requirement, which refers to the fact that a match is only successful when the job applicant is a good fit for a job listing *and vice versa*. Recommender systems in domains such as e-commerce usually only require unilateral matching; if a product is suitable for a customer then the customer may choose to purchase the product, and it is not usually necessary to consider if the customer is a suitable owner for the product. However, the set of jobs that are suitable for a candidate may be limited by the requirements or hiring preferences of the jobs or recruiting agents. If a job is recommended to a candidate because the details of the job meet their personal requirements, but the candidate does not meet the requirements of the job listing, then the recommendation is poor since the candidate is unlikely to progress with their application. To address this challenge, jobs recommended to a given candidate must either be filtered to include only those for which they meet the requirements (which may be difficult given that qualifications and requirements may be embedded in free text), or alternatively the job recommender system should be driven by data from *successful* applications (i.e. where an offer is made to the candidate post-application), rather than unsuccessful applications or applications where the outcome is unknown.

Related to the issue of bilateral matching is the issue that, although the same job may be recommended to multiple candidates while it is open for accepting applications on the online platform, in many cases only one candidate can ultimately be successful in their application. A candidate suitable for a job may make an application which is unsuccessful due to the presence of another, more suitable candidate who made an application. In this way, the ultimate outcome of a given application may not be an absolute indication of the candidate's suitability for the job. To address this issue, it may be pragmatic to consider alternative indications of suitability in application data, for example, whether the candidate was invited to interview or not. Experimentation and discussion of this particular issue are detailed in §5.5.3.1.

Several survey papers review different approaches to building job recommender systems (Dhameliya and Desai (2019); Siting et al. (2012); Al-Otaibi and Ykhlef (2012); Tripathi, Agarwal, and Vashishtha (2016)), where the function of the system is purely to recommend jobs to candidates and not vice versa. Approaches to constructing job recommender systems tend to fall into one of four categories; content-based filtering (§3.2.3), collaborative filtering (§3.2.4), knowledge-based filtering (§3.2.5), and hybrid approaches (§3.2.6). Related work in job recommender systems is visualised in a taxonomy in Figure 3.2.

Additionally, there are two historic challenges regarding job recommendation: the 2012 Job Recommendation Challenge hosted on Kaggle⁴ and the 2016 RecSys Challenge⁵. These challenges invited participants to develop and submit job recommendation solutions that were to be trained and evaluated on provided data. Although no submitted solutions to the Kaggle Job Recommendation Challenge were subsequently published, several submissions to the 2016 RecSys challenge were detailed in published research papers which are analysed in this review of related work.

⁴ <https://www.kaggle.com/c/job-recommendation>

⁵ <http://2016.recsyschallenge.com/>

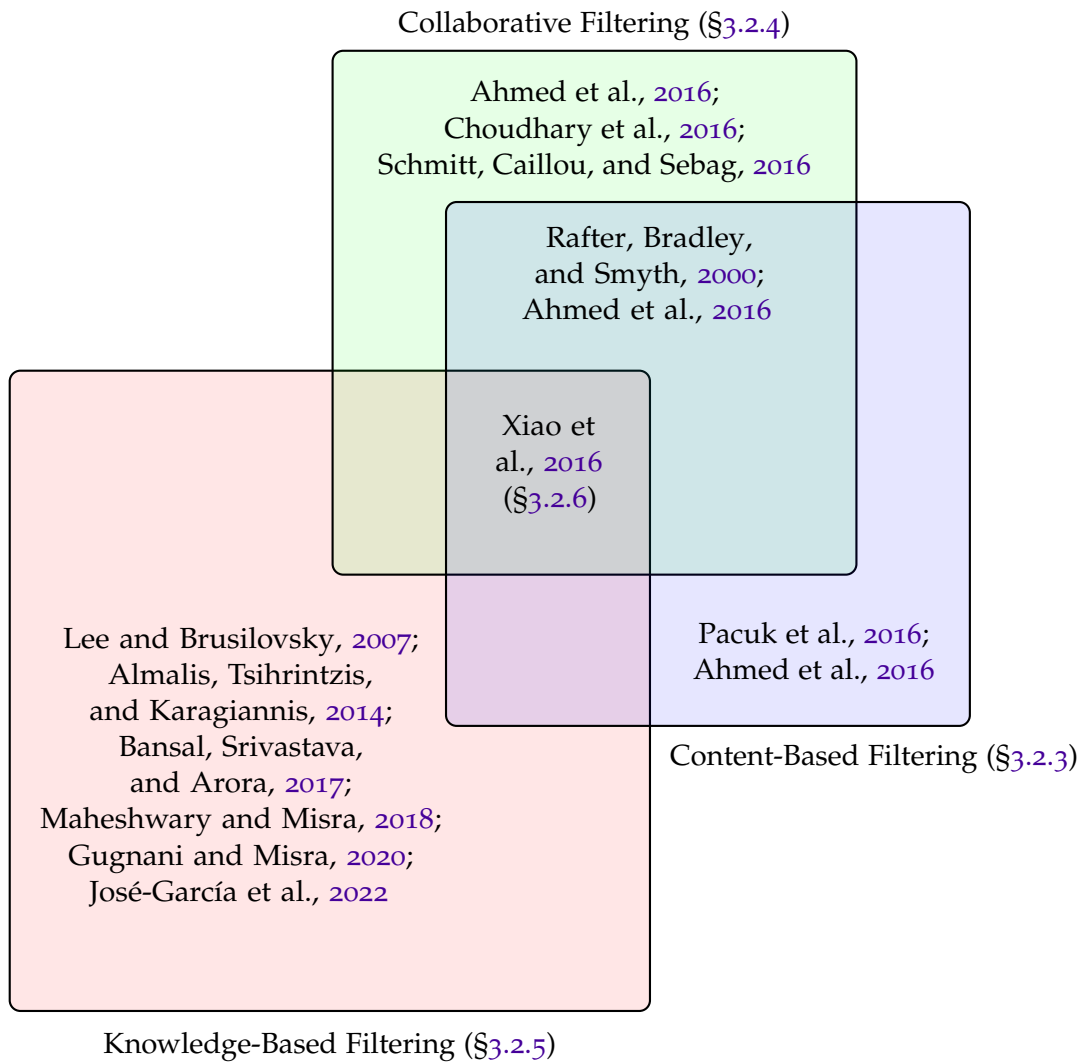


Figure 3.2: Taxonomy of Job Recommendation Systems.

3.2.1 RecSys 2016

The 2016 RecSys challenge⁶ was a competition co-hosted by XING⁷, a career-oriented social networking site and competitor of LinkedIn⁸.

Participants in the challenge were issued the following task: given a XING user and their associated user features, predict which job postings the user will positively interact with (for example, clicking on them, or bookmarking them for later review).

Specifically, at most 30 recommendations were to be generated for each of the 150k target users, and competing systems were evaluated using a bespoke scoring metric which combined precision, recall, and *user success* as shown in Equation 3.1. Although it is not known why such a non-standard scoring metric was chosen, according to the competition organisers this metric reflects typical use cases on the XING platform.

$$\begin{aligned} \text{score}(S, T) = \sum_{(u,t) \in T} & 20 \times (P_2 + P_4 + R + \text{userSuccess}) \\ & + 10 \times (P_6 + P_{20}) \end{aligned} \quad (3.1)$$

where:

- S represents the predicted set of 150,000 (u, r) tuples, where r is the ordered list of recommended items for user u .
- T represents the test set of 150,000 (u, t) tuples, where t is the ground truth set of relevant items for user u .
- P_k calculates the precision at position k between the recommended list r and the relevant item set t .
- R calculates the recall between the recommended list r and the relevant item set t .
- userSuccess calculates whether at least one relevant item was recommended for user u , described by Equation 3.2.

⁶ <http://2016.recsyschallenge.com/>

⁷ <https://www.xing.com/>

⁸ <https://www.linkedin.com>

$$\text{userSuccess}(r, t) = \begin{cases} 1, & \text{if } r \cap t \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (3.2)$$

The systems developed to compete in the RecSys challenge are of particular interest to the field of job recommendation given that they can be relatively evaluated; differences in model performance can be reliably attributed to model architecture since models were trained and tested on the same data. However, there is one particular caveat that concerns the use of the RecSys challenge in job matching research; the task structure does not account for the issue of bilateral matching (described in more detail in §3.2). In short, systems are rewarded for predicting with high precision which jobs a given user will apply to, but the suitability of the user for the job is not considered. It may be the case that candidates *interact* (click on, bookmark, etc) with jobs that are not suitable for them, and although this job would be a poor recommendation for the user, the interaction is classified as *positive* in the RecSys challenge data and systems are encouraged to recommend that job to the user.

Additionally, the data for the RecSys challenge is no longer accessible to the academic community, so novel methods of job recommendation cannot be reliably compared to those submitted as part of the RecSys challenge.

Nonetheless, there are several insights regarding feature engineering and model architecture that can be drawn from candidate systems for the RecSys challenge that may influence research into a job matching solution.

Xiao et al. (2016) achieved first place in the challenge with a hybrid model that combined knowledge-based filtering, content-based filtering, and collaborative filtering with an ensemble model. In addition, *Hawkes Process* (Hawkes, 1971) was used, which models the temporal patterns of users' historic activity to predict the items the user will interact with next. The overall recommendation model features ensemble learning from four different components, each of which calculates a relevance score for a user-item pair. The relevance scores from each component are combined in an ensemble Gradient Boosting Regression Tree (GBRT) layer. The four components comprise a Logistic Regression model, a GBRT model, an eXtreme Gradient Boosting (XGBOOST) model, and a Temporal Intensity (TI) model using Hawkes Process. The TI model

formulates user activity history as a time sequence (i.e. $\{t_1, t_2, \dots, t_n\}$), and models a conditional intensity function $\lambda(t)$ using a temporal point process to calculate the probability that the next user interaction will happen at time t (where $t > t_n$). The temporal intensity function for user-item pairs (u, i) is shown in Equation 3.3, where $\lambda_0^{u,i}$ and $\alpha^{u,i}$ are the (u, i) -th entries of the base intensity matrix Λ and self-exciting⁹ matrix \mathbf{A} respectively, and the summation of $\gamma(t, t_j^{u,i})$ over time gives the dependency of the user's historical activities.

$$\lambda^{u,i}(t) = \lambda_0^{u,i} + \alpha^{u,i} \sum_j \gamma(t, t_j^{u,i}) \quad (3.3)$$

This model uses **TF-IDF** to identify and extract keywords in both user and item features and applies a method of topic modelling, Latent Dirichlet Allocation (**LDA**), to extract latent topics. Cosine similarity is applied to these topics to calculate user-user, item-item, and user-item similarities. The justification for developing the ensemble model proposed by Xiao et al. (2016) is that the combined process is able to capture both semantic relevance of user-item interactions as well as modelling the temporal characteristics of those interactions, and that the knowledge-based filtering approaches (achieved by calculating the similarity of user and item features) are able to address the cold-start problem associated with content-based and collaborative filtering approaches.

Pacuk et al. (2016) achieved second place in the challenge with a primarily content-based recommender system which used **XGBOOST** to compute the probabilities that a given user u would interact with item i . Of the identified feature groups contributing to the model's F_1 score, *event based* features were shown to have the most importance, which were percentages of items from the set of items a user had interacted with that had some property corresponding to item i . The second most important listed feature was *item global popularity*, which refers to the total number of interactions corresponding to item i across all users. Although *collaborative filtering most similar* is listed as the third most important feature set in this model, this not only includes features calculating similarity between users by comparing item sets they interacted with, but also similarity between item i and the items that user u has interacted with, which is in fact a form of content-based filtering.

⁹ Matrix \mathbf{A} is *self-exciting* in that the occurrence of past points makes the occurrence of future points more probable (Reinhart, 2018).

Bansal, Srivastava, and Arora (2017) propose a knowledge-based filtering approach (defined as *content-based filtering* in the original study, but reclassified here as per our definitions outlined in §2.6.1) to job recommendation. This model applies LDA to job data to extract *topics*, which are then matched with mentioned job roles in user data using two feature similarity techniques: cosine similarity (see §2.4.2.1) and *flexible string matching*, the latter of which is ostensibly the size of the intersection between extracted terms from each user and job¹⁰. Given that string matches appear to be performed using the entire morphological form of the term, it is not clear which part of this process is *flexible*, given that terms expressed in different ways will cause this matching process to fail. The authors do not provide the results of their system using the established scoring metric shown in Equation 3.1, nor do they provide code to replicate their work, so it is infeasible to evaluate the performance of their contribution relative to the model submitted by the challenge winners.

3.2.2 Job Recommendation Challenge

The Kaggle Job Recommendation Challenge (KJRC)¹¹ was hosted on data-science community site Kaggle in 2012. Sponsored by employment website CareerBuilder¹², the challenge required competitors to predict which jobs users applied to based on their previous applications, demographic information, and work history. Insights discovered from the data were to be used to allow CareerBuilder to improve its job recommendation algorithm, said to be a core part of its website and a key element in improving user experience.

The challenge provided clearly documented training, development and evaluation data as part of the challenge, which is still available for download by any Kaggle user on the challenge page.

Unfortunately, there are no papers nor code repositories associated with successful entries. Various methods are suggested and discussed by competitors in the challenge discussion forum, from which it may be inferred which approaches were more successful than others (for example, collaborative filtering is mentioned many times in the

¹⁰ This appears to be the case from the description in §5.1, although the provided example appears to contradict this.

¹¹ <https://kaggle.com/c/job-recommendation/>

¹² <https://www.careerbuilder.co.uk/>

discussion), but competitors appear reluctant to share intricate details about successful models (for example, the 4th place competitor provides a high-level approach for overcoming cold-start issues, but refers to the technique as ‘black magic’).

Although the data is useful for training and testing an applicant profile-job description matching solution, it is limited in that it can only provide unidirectional recommendations; there is no data to indicate whether an application was successful, and therefore it is not possible to evaluate the strength of an individual application. However, the leaderboard associated with the KJRC provides an appropriate target for an applicant profile-job description matching solution developed as part of this research project.

3.2.3 *Content-Based Filtering*

As defined in §2.6.1, content-based filtering methods leverage item-item similarities in order to recommend unseen items to a given user that are similar to the items the user has rated favourably (or in implicit systems, items that user has interacted with). Methods that calculate item-user similarities (for example, using extracted skill terms) are classified in this thesis as knowledge-based filtering methods, although work in this area often conflates both item-item and user-item approaches as content-based filtering methods.

Related work that focuses solely on content-based filtering techniques is scarce, which may in part be due to the comparatively greater performance of hybrid systems, even in cases where data sparsity is high. Ahmed et al. (2016) compare the performance of item-item, user-user, and hybrid models at varying levels of data sparsity using the XING job recommendation data described in §3.2.1. Although low sparsity (10%) yields comparable results for item-item and user-user methods (using the XING scoring equation for evaluation, Equation 3.1), sparsity between 30% and 90% yields considerably stronger results (between 2× and 3.5×) for item-item based methods. However, hybrid methods outperform even the stronger of the two singular methods (also between 2× and 3.5×), which suggests that ensemble methods are better able to predict candidate-job interactions, even when sparsity is high.

3.2.4 Collaborative Filtering

As defined in §2.6.2, collaborative filtering methods leverage user-user similarities in order to recommend unseen items to a given user that a similar user has interacted with favourably (or in implicit systems, items that a similar user has interacted with).

The collaborative filtering job recommender system developed by Choudhary et al. (2016) creates clusters of users based on skill-terms extracted from user profiles¹³ and uses the Euclidean Distance and Pearson Coefficient to calculate similarities between skill terms. Although Choudhary et al. (2016) claim that the system is successful in recommending jobs based on a user's current skill set by combining it with similar skills in the global dataset, it is not clear how recommendation performance varies across candidates who may be in the centre of user clusters or on the periphery, where recommendation may be poorer. Nonetheless, the principle of leveraging preferences of similar users to guide recommendations for other users is potentially very powerful, and will likely feature in a successful hybrid solution that combines the collaborative approach with the content-based approach (discussed in §3.2.6).

Schmitt, Caillou, and Sebag (2016) developed MAJORE (MATCHing JObs and RESumes), a collaborative filtering job recommendation system with a dedicated module for overcoming the cold-start problem. The corpus, which was not made publicly available, contains job posts on an online portal, user CVs, and instance where users had *clicked* on the job description, subset to include only users and items that had made or received at least 5 clicks. The authors found that convolutional metrics may, in part, compensate for the sparsity of the user-item interaction matrix. However, the authors design a bespoke experimental task and goal (for each job: remove a click (user-item interaction); rank all CVs; calculate the rank of the CV with the removed click; calculate recall over 20 independent samples of click removal), which precludes the interpretation of the effectiveness of MAJORE compared to other job recommendation systems.

¹³ The process by which skill terms were extracted by Choudhary et al. (2016) is not detailed. This is discussed in chapter 4.

3.2.5 Knowledge-Based Filtering

As defined in §2.6.3, knowledge-based filtering methods leverage user-item connections in order to recommend unseen items to a given user that feature attributes which are associated with the attributes of the user.

Proactive, the knowledge-based job recommender system developed by Lee and Brusilovsky (2007), utilises five components: a web spider, an ontology checker, a profile analyzer, a preference analyzer, and a user interface generator. The web spider acquires and parses job information from external sources and outputs to an ontology checker. Two kinds of ontology are used: the first about job category, and the second about company information (for example, the industry or domain, number of employees). The profile analyzer component generates job recommendations by comparing the differences of distance in the *weights* between an explicitly selected *favourite job* and open job positions. The authors do not state where these weights are extracted from, but note that the ontological relationships between jobs help calculate the weight values and provide structural understanding of a user's interest which reduces initial efforts to acquire knowledge about users. The preference analyzer component interprets explicitly defined user preference and refines the recommendations generated by the profile analyzer component, and the user interface generator extracts further information about user engagement with the system in order to enhance the user profile.

Although the research conducted by Almalis, Tsihrintzis, and Karagiannis (2014) defines itself as a content-based filtering approach, because it utilises *skills* extracted from candidate CVs, this thesis considers this a knowledge-based filtering approach (see §2.6.1 for a more detailed explanation). This work uses the *skills* contained in job positions as a structural representation of each of the items, converted into an n -dimensional vector with each element (the skills) mapped to an attribute with a value (the required skill level) and weight (the importance of that skill to the job position). It is implied that there is an automatic process that converts the unstructured content of the job description into the structured form based on attributes, values, and weights, but this is not stated explicitly. The user profile, which consists of the candidate CV only, is similarly converted to an n -dimensional vector, and the Minkowski distance at p values of 0.5, 1, 2, and 4 was used as a metric of similarity between each user profile and item representation.

José-García et al. (2022) developed C₃-IoC¹⁴; a tool that matches user skill profiles to relevant job roles. The tool uses keyword extraction to identify technical skills in a user’s uploaded CV by matching against a technical skill bank constructed through a combination of web scraping, ‘text mining techniques’, and the O*NET¹⁵ skill database. Additionally, a questionnaire was provided to the user in order to derive the level at which the user embodied non-technical skills, which was manipulable by the user through radar charts. User skill profiles and job roles were represented as skill vectors weighted by the skill level the user embodied or the level at which the job required that skill. Similarity between user u and job j skill weightings were calculated using the bespoke metric shown in equation 3.4, where S is the fixed set of N skills, $S = \{s_1, s_2, \dots, s_N\}$, and higher similarity indicated a better match between user and job.

$$\text{sim}(u, j) = 1 - \frac{\sqrt{\sum_{s \in S} \max(j_s - u_s, 0)}}{\sqrt{\sum_{s \in S} j_s}} \quad (3.4)$$

The advantage of C₃-IoC over similar job recommendation systems is that it incorporates the relative strength of skills. This is an important part of the recommendation process that is neglected by systems that treat the presence or lack of skill terms as binary values. For example, a user who states they are a *novice* at the programming language *Python* may be shown they have the necessary skillset to qualify for a Senior Software Development position that lists *expertise in Python* as a requirement for the job if skill levels are not taken into consideration.

3.2.6 Hybrid Filtering

An example of a job recommender system using a hybrid approach is CASPER (Rafter, Bradley, and Smyth, 2000) which features both content-based filtering and collaborative filtering methods. In this system, an automated collaborative filtering component gathers data regarding job listings a user has viewed, how long the user viewed the

¹⁴ Originally hosted at <https://www.c3-ioc.co.uk/>, now accessible only through the Internet Archive <https://web.archive.org>

¹⁵ <https://onetonline.org/>

listing, and which listings the user returned to view at a later date. This data is combined with data from a personalised case retrieval (PCR) module, which performs a two-stage process to produce job recommendations. Stage one of the PCR module finds suitable job matches by calculating the weighted sum of the individual feature similarities between a target job description (provided by the user) and other job descriptions. Stage two of the PCR module personalises the retrieval results to the user by using a form of k-nearest neighbour (Cunningham and Delany, 2020) to calculate relevance of the recommendation to the user, and suppresses recommendations that are shown to be not relevant. Rafter, Bradley, and Smyth (2000) show the strength of the hybrid approach over singular approaches, and show that a combination of collaborative and content-based recommendation techniques can be leveraged to improve recommendations.

3.3 LARGE SCALE ANALYSIS OF JOB DESCRIPTIONS

Alabdulkareem et al. (2018) perform large scale analysis of job adverts to investigate the distinction between high- and low-wage occupations in terms of skill requirements. No definition of skills is provided, but the authors use the O*NET database of skills in their analysis. The authors identify two skill clusters using a variant of relative frequency analysis, and interpret these as ‘social-cognitive’, associated with high-wage occupations, and ‘sensory-physical’, associated with low-wage occupations. The authors reason that this polarisation constrains the ‘career mobility’ of workers, since low-skill workers are ‘stuck’ relying on a low-wage skill set, and propose that strategies need to be developed to mitigate the negative aspects of automation that replaces a proportion of low-wage occupations.

Fareri et al. (2020) develop a quantitative measure to gauge the readiness of employees within a large firm with respect to the *Industry 4.0* paradigm, which refers to the sociotechnical revolution of technology in the workforce. The authors consider a variety of structured databases for job profile characterisation, including ESCO and O*NET, and compute summary statistics for 6 groups of skills: *everyday execution*, *operational skills*, and *functional skills* for both 4.0 profiles and non-4.0 profiles. The authors show that the 4.0-ready profiles are more adequate to carry out operative activities through the relevant support of their soft skills. Additionally, 4.0-ready profiles are stronger on *transversal skills*, and the authors propose that, for managing the introduction of

new technologies, there should be an important *soft component*, which is becoming increasingly central in the digital era.

Lyu and Jin Liu (2021) perform a large scale analysis of job adverts in the US energy sector, and find an increasing demand from 2010 to 2019 for *soft* skills, including social, cognitive, people management, project management, and customer service skills. No formal definition of a skill term is given, but the authors note that skills are pre-identified in the supplied data (provided by Burning Glass Technologies, an employment analytics firm), and provide a count of 13,565 unique skills in the data. A further trend the authors identify is that of the relatively flat requirement for *hard* skills over the same time period in which *soft* skill requirements increase. The authors show that *products* and *marketing* skills are the most valuable to energy firms, and contribute the highest to firm productivity. However, these same skills are the least commonly required in the energy sector, which highlights a potential change in strategy that firms may need to address when hiring employees.

Junhua Liu, Ng, Wood, et al. (2020) developed the Industrial and Professional Occupation Dataset (IPOD), composed of 475k job titles associated with around 200k users on occupational social network LinkedIn¹⁶. Each job title is annotated with its level of seniority, domain of work, and location. Subsequently, this corpus was used to develop *Title2vec* (Junhua Liu, Ng, Gui, et al., 2022), a contextual job title vector representation. By using *Title2vec*, the authors conducted large scale analysis of job adverts, focusing on discrepancies between employee-job interaction in Asia (Singapore) and the US (Denver, CO). The researchers found that, in both regions, employees tend to stay between 2 to 3 years at a given position, but job retention tends to be higher in the US. Linear regression was performed to determine whether the number of years in education is able to predict the length of employment, which showed a significant effect for Asia but no significant effect for the US.

Piróg and Hibszer (2022) performed large scale analysis of job adverts to determine the extent to which employers require the experience listed in job adverts. Data used for analysis comprised 17k job adverts in the environmental sciences domain across six European countries. In this particular domain, approximately half of job adverts required some work experience (between a few weeks to a year), and the remainder required considerable experience (at least one year). However, this requirement varied by

¹⁶ <https://www.linkedin.com/>

the language in which the job advert was written; for example, job adverts written in Polish expressed the lowest requirement for professional experience, with over 90% of job adverts in this field listing no required experience. Conversely, English job adverts listed higher requirements, and German job adverts required the highest professional experience requirements of those included in analysis. Word co-occurrence statistics were computed to show the words associated with certain groups of professions, which were then ranked by importance. For example, in Polish adverts, the strongest relationship between the word *experience* was *interpersonal* [skills], and the researchers suggest that employers look for documented professional competences, and interpersonal skills acquired primarily through collaboration with people using a foreign language. In German adverts, however, the word *experience* most strongly related to the words *analytical*, *responsibility*, and *inter-disciplinary*, and the researchers suggest that employers who advertise in German seek candidates experienced with working with geographic information systems, performing analyses where data sources and knowledge from Earth science-related disciplines is utilised. The researchers conclude that work experience capital is not a basic condition for finding employment related to a candidate's degree, but can enhance candidates' competitiveness on the labour market.

3.4 NATURAL LANGUAGE PROCESSING IN JOB DESCRIPTIONS

3.4.1 Word Embeddings

Word-level and document-level embeddings have been used in the field of applicant profile-job description matching for different purposes. Gugnani and Misra (2020) utilised both word2vec and doc2vec models in a proposed matching system. Vector representations of words were used to calculate the probability that a word appearing in a resume was a *skill term*, which were considered by the authors to be an important aspect in evaluating the strength of a job recommendation¹⁷. Vector representations of documents were used in a separate part of the matching system for finding similar job descriptions to a given job description.

Maheshwary and Misra (2018) reported greater accuracy and F_1 -score on an applicant profile-job description matching task when using doc2vec to create vector

¹⁷ The role of skill terms in matching solutions is discussed in more detail in §3.4.2.

representations for applicant profiles and job descriptions compared to using a bag-of-means approach where average word2vec embeddings were used as a feature set, which suggests that document embeddings are better at representing the document as a whole than taking the average of word embeddings.

E. Smith, Weiler, and Braschler (2021) used word embeddings in a skill-extraction task to identify semantically related words to known skill terms. Word2vec was trained on 10,000 unlabelled job descriptions, and mean similarity between unknown terms and skill terms, and in a separate method, between *syntactic headwords* and skill terms, where *syntactic headwords* were defined as the terms that *indicate the presence of a nearby skill-phrase*, for example the term *degree in* in the skill phrase *degree in business administration*, or the *experience* in the phrase *project management experience*. The authors found that simple syntactic methods such as dependency tree parsing outperformed word embeddings, but noted that this was a ‘small-data problem’, in that word embeddings typically perform well in tasks where large amounts of training data are available. It appears that the authors did not leverage pretrained word embeddings for this particular task, and chose instead to train from first principles on their small dataset. It is possible that leveraging pretrained word embeddings, which have been exposed to large quantities of natural language, may be more effective at ER, especially after fine-tuning on a smaller, more task-specific dataset. Nonetheless, this research does support the notion that simpler algorithms may be more suitable to this particular application of ER, and it is worth including them in experimentation even though larger, more complex language modelling methods such as word embeddings, are available.

3.4.2 Entity Recognition

In traditional machine learning approaches, feature extraction refers to the process of building derived values (*features*) from initial data to facilitate the subsequent learning and model establishment steps. In the context of applicant profile-job description matching, this involves parsing unstructured text and extracting salient details from applicant profiles or job descriptions that are used as input to a recommendation model.

Applicant profile-job description matching systems that use feature extraction tend to use the *skillset* contained in the input data as the features to be extracted and used as input to a recommendation model (Almalis, Tsihrintzis, and Karagiannis (2014);

Choudhary et al. (2016); Hoang et al. (2018); Gugnani and Misra (2020)). The underlying assumption is that a high similarity between the set of skills an applicant has compared with the set of skills a job description requires is a strong indicator of a good fit.

However, there are two main issues in existing literature regarding skill extraction for feature selection. Firstly, there is no academic consensus on the definition of a skill-term, which makes the comparison of different extraction methods a difficult task. Secondly, there is no agreed-upon optimal method for the extraction of skill-terms from applicant profiles or job descriptions, so there is a need for research into the evaluation of different methods.

Literature regarding the extraction of skill-terms from applicant profiles and job descriptions tends to utilize one of three approaches to defining the problem. The first approach is to avoid problem definition entirely by offering no explicit definition for a skill-term. Bastian et al. (2014) allows the users of a service (LinkedIn) to define skill-terms themselves without explicit guidance from the researchers nor a formal definition. Other work avoids problem definition by assuming that anything contained in a user-defined *Skills* section of an applicant profile qualifies as a skill-term (Maheshwari, Abhishek, and Reddy (2010); Kivimäki et al. (2020); Karakatsanis et al. (2017)). Other work employs field experts to annotate terms as skills, and terms with high inter-annotator agreement are classified as skill-terms (Gugnani and Misra, 2020). In some cases, research into the extraction of skill-terms from applicant profiles and job descriptions offers no explanation of what a skill is whatsoever (Gugnani, Kasireddy, and Ponnalagu (2019); Choudhary et al. (2016)).

The second approach to problem definition in skills extraction research is to provide the reader with examples of skill-terms in lieu of a formal definition. Shi et al. (2020) show an annotated *Key Qualifications* section with skill-terms highlighted, such as *machine learning* and *natural language processing*. However, some terms that could be perceived as skill-terms appear in the example and are not highlighted (for example, *generative and discriminative models*), which is evidence of the ambiguity of this method of problem definition. Hoang et al. (2018) use the same method of problem definition by providing an example annotated job listing, but in this example they include parts of job titles in the annotations (for example, *financial* in *financial accountant*) and exclude other terms that could be perceived as skill terms (for example, in the phrase *monitoring*

budgets, developing forecasts, and investigating variances, only the terms *budgets* and *forecasts* are classified as skill terms; the associated verbs and *investigating variances* are excluded).

The third approach to problem definition is to refer to public databases of skill terms such as O*NET¹⁸ or those defined in the official frameworks such as the European Qualifications Framework (EQF), part of the European Skills/Competences, Qualifications and Occupations Commission (ESCO) (Khobreh et al., 2016). This framework defines skills as *the ability that enables the learner to apply knowledge and use know-how to perform tasks*, and makes the distinction between skills and competences, which are defined as *the proven abilities to use knowledge, skills and personal, social and/or methodological abilities, in work or study situations and in professional and personal development*. As at February 2021, the ESCO database contains 13,485 skills and competences. The main limitation of using public databases of skill terms is that they are not effective for detecting new skills or detecting known skills expressed in new ways, and require continual updating in order to retain their usefulness. In areas of industry that feature constant development of new techniques, for example, machine learning or computer programming, new methods and techniques will elude skill databases until they have been identified by the database maintenance team and added. ESCO is updated with new terms annually¹⁹ which means that skill extraction methods using this database may be up to a year out of date.

There are several methods for skill extraction that have been proposed. Gugnani and Misra (2020) found that a mixed-method approach was the most effective, combining a dictionary match method, a part-of-speech tagging method, a named entity recognition method, and a word embedding method to generate a probability that a given word or phrase in a candidate profile was a skill term.

Dictionary matching is the most straightforward of the approaches, and involves the compilation of one or more *skills dictionaries*. Words and phrases are checked against these dictionaries, and if there is a match, the term is likely to be a skill term. In the Gugnani and Misra (2020) system, skill dictionaries were compiled from online sources such as Wikipedia²⁰, O*NET, Hope²¹, and various other uncited online public dataset resources with terms identified as skills. Cumulatively, the skill dictionaries contained

¹⁸ <https://onetonline.org/>

¹⁹ Changes to ESCO are detailed in the changelog at https://ec.europa.eu/esco/portal/escopedia/ESCO_v1.

²⁰ <https://www.wikipedia.org/>

²¹ <https://www.computerhope.com/>

53,293 terms, and a feedback loop was included in the overall system architecture so that new terms that were eventually classified as skills were added to the dictionaries. Associated with each of these terms was a *weight* value corresponding to the likelihood of the term being a skill, but the process by which this weight value is assigned for the original skill terms is not described in the paper.

POS tagging is the process of assigning *part of speech* tags, such as *noun* and *verb*, to words in a sentence (Toutanova et al., 2003). POS tags can be used to influence the likelihood of a term being classified as a skill. Gugnani and Misra (2020) combined a POS tagging system with hard-coded rules such as *if a sentence has a comma separated list of nouns, where one or more nouns is a skill then the other set of nouns are probably skills*.

NER is the process of identifying and classifying *named entities* in text into pre-defined categories (for example, *organisation*, *person*) (Ratinov and Roth, 2009). Some services, such as IBM Watson NLU²², also identify *keywords*, *entities*, and *concepts* from text, which can be leveraged to aid in the classification of skill terms. Gugnani and Misra (2020) used Watson NLU to parse candidate profiles, and extracted the terms that were identified as keywords, entities, and/or concepts. Alongside this list of terms, weights were assigned to each based on how many of these services extracted the term (for example, a term identified as both a *keyword* and an *entity* is assigned a greater weighting than a term only identified as a *concept*).

Word embedding methods are explained in detail in §2.4.2. Using similarity metrics such as cosine similarity on word embeddings, it is possible to quantify the semantic similarity between two words. If a term has high similarity to a known skill term, that term is likely to also be a skill term. Gugnani and Misra (2020) trained a word2vec model on a corpus consisting of 1.1 million job descriptions from a variety of domains and the Wikipedia pages of all the terms in the skill dictionaries. In the embedded word2vec space, new terms were compared to each term in the skill dictionaries, and the highest cosine similarity was taken as the likelihood that the term was also a skill term. In the case of multi-word phrases (for example, *computer programming*, *ability to work under pressure*), an average of the word vectors for each word was taken as the embedding for the skill phrase.

The terms and their weightings extracted from the dictionary match, POS tagger, NER, and word2vec methods were combined with additional parameter weights in a formula

²² <https://cloud.ibm.com/docs/natural-language-understanding/categories.html>

to calculate a *relevance score* for each term, which was the likelihood that the term was a skill term. Terms with a relevance greater than 0.35 were considered relevant, and therefore skill terms. The additional parameter weights and the relevance threshold were reported to have been derived from *empirical evaluation*, but no further details were given to support this.

Shi et al. (2020) utilise attention-based methods in the development of *Job2Skills*, a salience and market-aware skill extraction method which was shown to improve the quality of LinkedIn job targeting skill suggestions and job recommendation. Attention-based methods such as the Transformer (Vaswani et al., 2017) have been firmly established as state of the art approaches in sequence modelling, language modelling, and machine translation. They mimic the cognitive attention process of enhancing salient parts of input data. In doing so, they overcome some of the issues associated with RNNs and CNNs, such as the sequential processing limitation and difficulty learning long-range dependencies between associated words. *Job2Skills* (Shi et al., 2020) first uses an *in-house skill tagger* to find all possible skill mentions and uses a feature-based regression model to link these possible skills to known skill entities, although the details are not provided for either the skill tagger or the feature-based regression model. Attention-based methods such as BERT (Devlin et al., 2019) were used to encode the skill terms and the sentences in which they appear in order to train a gradient descent model on a classification task, where skill terms were classified as either *salient and market-aware* for that particular job posting or not. Although Shi et al. (2020) show that attention-based methods can be used to great effect in skill extraction, they treat the initial problem of isolating and classifying skills as solved and out-of-scope, although the related work described in this section suggests that this claim is made prematurely. However, attention-based methods have been shown to be particularly effective, which suggests that these methods should be considered when developing a feature extraction system.

It is clear there is a need for a formal definition of each of the salient entities in applicant profiles and job descriptions, and a public, labelled dataset that can be used to develop and evaluate feature extraction systems. The first research question of this thesis focuses entirely on this aspect, and involves a list of entity classifications and their definitions in the form of an annotation scheme, as well as the construction of a public dataset of job descriptions with salient entities identified and labelled.

Towards the end of our research, and thus too late to be of use in our experiments, the SkillSpan (M. Zhang, Kristian Nørgaard Jensen, et al., 2022b) data was also published which shares methodology similar to ours. The authors present a dataset for skill extraction consisting of 14.5k sentences (12.5k annotated spans). Sentences were compiled from three web sources: an unnamed large job platform; the Danish Agency for Labour Market and Recruitment; and StackOverflow²³ jobs, which features job listings exclusively in the technology domain. Annotations were conducted by three annotators over the course of eight months, whereby job descriptions were reviewed and *Skill* and *Knowledge* spans were identified according to ESCO definitions. Using this data, various Transformer-based architectures were applied for ER. The most successful of these, ‘JobBERT’, was a BERT model that had been pre-trained on 3.2M unlabelled sentences from job descriptions. F_1 scores for this model on the combined task (that is, the identification of both *Skill* and *Knowledge* spans) approached .60.

3.4.3 Natural Language Inference

To our knowledge, there has not yet been any research dedicated to addressing the job matching problem using the NLI paradigm, which frames the job description as the *premise* and the user profile as the *conclusion*, and aims to predict the relationship between the two. However, there is a wide body of research into the general problem of NLI, which serves as a useful foundation for our work into the job matching problem.

The SNLI corpus (Samuel R. Bowman et al., 2015) is a collection of 570,000 human-written English sentence pairs, each labelled with a single classification denoting the relationship between hypothesis and conclusion: {Entailment; Contradiction; Neutral}. The dataset is accessible online²⁴ under a CC-BY-4.0 license. During data collection, approximately 2,500 human annotators on the Amazon Mechanical Turk (AMT) platform were presented with a photograph and an accompanying caption, and asked to produce three alternative captions: one that is true, one that could be true if more information were available, and one that is false. The photographs accompanying the sentence pairs are not available in the SNLI corpus. Approximately 60,000 sentence pairs were subsequently annotated in a validation task, where 4 annotators (not including the

²³ <https://stackoverflow.com/>

²⁴ <https://huggingface.co/datasets/snli>

original hypothesis author) labelled each sentence pair. IAA statistics are shown in Table 3.1.

Label	Fleiss κ
Contradiction	.77
Entailment	.72
Neutral	.60
Overall	.70

Table 3.1: IAA for the SNLI Dataset (Samuel R. Bowman et al., 2015).

The attention-based neural architecture proposed by Parikh et al. (2016) *decomposes* the NLI problem into sub-problems that may be solved in parallel, vastly decreasing training and evaluation times compared to standard neural architectures. Attention mechanisms are described in more detail in §2.5.1.2. After embedding the input sentences, they are softly aligned using attention, and the soft alignment is decomposed into sub-problems which are solved separately and subsequently merged to produce the final classification. This approach is referred to as decomposable attention since the approach *decomposes* the problem in this way. The feed-forward networks accept the softly aligned input sequences separately, which leads to linear complexity (length of premise + length of hypothesis) as opposed to quadratic complexity (length of premise \times length of hypothesis). The decomposable attention model was trained and evaluated on the SNLI data, and outperformed LSTM models which featured many more parameters (382K vs. 3.0M). It is important to note that the SNLI corpus features three classes for prediction rather than two, since a limitation of the decomposable attention model is its reduced ability to distinguish *neutral* relationships between sentence pairs. When applying the NLI paradigm to job matching, there is no corresponding *neutral* relationship; either a user is suitable for a given job, or they are not. Therefore, the application of a similar attention-based neural architecture may perform well in the dichotomy of job application status prediction. Experiments on this topic are discussed in §5.5.3.3.

The neural architecture proposed by R. Yang et al. (2019) utilises the same decomposition technique as Parikh et al. (2016), but modifies several components that were deemed slow and unnecessary, such as multi-way alignment mechanisms and dense

connections between stacked blocks. The proposed model encompassed three components which collectively give its name, *RE2*: *residual* vectors, which are the previously aligned features between input sequences, *embedding* vectors, which are the original point-wise features, and *encoded* vectors, which are the contextual features. The implementation of each layer is kept as simple and lightweight as possible, and the residual connections between consecutive blocks are augmented to provide richer features for the alignment process; the three parts in the input of alignment and fusion layers are: the original point-wise features (*embedding* vectors), previously aligned features processed and refined by previous blocks (*residual* vectors), and the contextual features from the encoder layer (*encoded* vectors). With this approach, the authors demonstrate improved performance on the SNLI corpus compared to a variety of NLI models, including the decomposable attention model proposed by Parikh et al. (2016). By reducing complexity and training times while achieving comparable or greater performance, this model architecture is a suitable choice for NLI tasks.

One particular challenge associated with developing ER systems on human-labelled data is the lack of perfect agreement between annotators, and methods are required to resolve disagreement between given labels (IAA is discussed in more detail in section 2.7.2). Rodrigues and Francisco Pereira (2017) proposed a method for addressing disagreement that does not require label aggregation methods; the Crowd Layer, which allows a model to learn directly from noisy annotations. The Crowd Layer accepts as input the output layer of the NN and learns an annotator-specific mapping from the output layer to the labels of the different annotators, which captures annotator-specific reliabilities and biases. The addition of the Crowd Layer was shown by Rodrigues and Francisco Pereira (2017) to increase the performance of a model on an NER task compared with the same model without the Crowd Layer using a variety of label aggregation methods. The performance of a CNN with the Crowd Layer was shown to be comparable to a CRF model, while considerably reducing training time (a few minutes versus several hours).

EXTRACTING SALIENT ENTITIES FROM JOB DESCRIPTIONS

4.1 CHAPTER OVERVIEW

This chapter addresses the first research question of this project:

RQ1 *How can salient entities in applicant profiles and job descriptions be identified and extracted for use in an applicant profile-job description matching solution?*

Section 4.2 explains the necessity for a public dataset of job descriptions with salient entities identified with an associated schema in order to develop automatic solutions for ER. Section 4.3 describes the process of selecting a corpus of job descriptions to form the foundation of the labelled dataset. Section 4.4 details the process by which the schema for the annotation task was developed over six rounds of testing, making incremental changes to task delivery and ancillary materials in order to improve annotator accuracy and inter-rater reliability. Section 4.5 describes the process by which an accuracy threshold was chosen, where Workers who met or exceeded this threshold on a task were invited to contribute to the live corpus, which is described in §4.6. Section 4.7 provides some statistics and analysis of the live corpus. Section 4.8 describes the different methods applied for ER, which are evaluated in §4.9. Section 4.10 discusses the application of ER methods beyond the current task in the context of job recommendation systems. Section 4.11 discusses the ethical implications of the research contained in this chapter. Section 4.12 details the publication of materials. Finally, section 4.13 summarises the research described in Chapter 4.

4.2 INTRODUCTION

The progression of research addressing **RQ1** is represented visually in Figure 4.1.

Time-consuming tasks, such as reading and evaluating many articles of text, are suitable candidates for developed automatic methods. These methods aim to replicate the

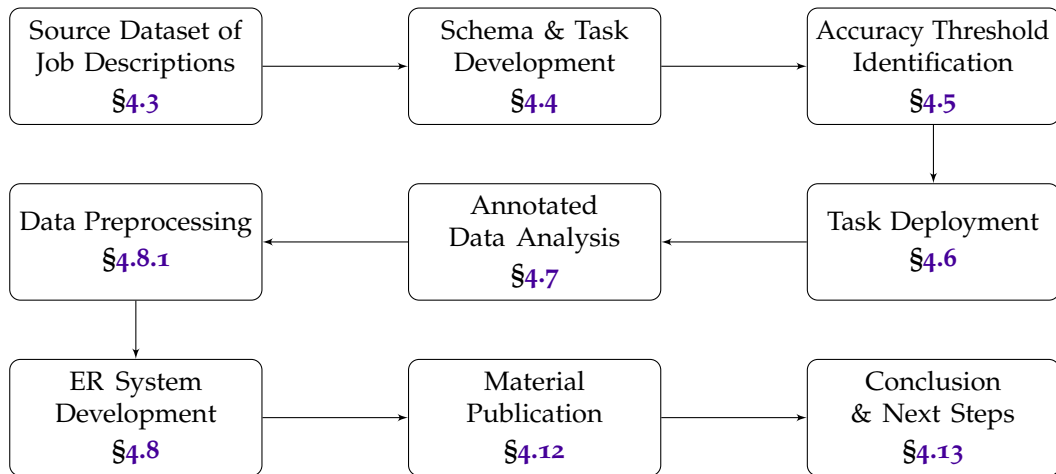


Figure 4.1: The flow of the research project dedicated to address Research Question 1.

human decision-making process at a much greater speed, and make fewer random errors in doing so as such tasks are highly susceptible to human errors when performed manually. The task of evaluating whether a job description is an appropriate recommendation for a candidate (or vice versa) is one such human task suitable for replacement with an automatic method. However, a prerequisite for this process is the ability to identify which components of a job description or candidate profile are important when evaluating whether a given job or candidate is a good recommendation or not. With the added difficulty that job descriptions and candidate profiles are non-uniform and often unstructured, methods are required to parse text, and to identify and extract the salient entities within.

The identification and extraction of salient entities in job descriptions and candidate profiles can be considered an Entity Recognition (ER) task. In order to develop an ER system, the following items are required:

- a dataset of job descriptions and/or candidate profiles with tagged salient entities
- a machine learning algorithm suited to ER that can be trained to recognise the salient entities in free text

Prior to this research project, there were no publicly available datasets of job descriptions with tagged salient entities. Given that this data is a requirement for developing

an ER system for extracting salient entities, the first step to address this research problem was to create and publish such a dataset by sourcing annotations.

The following components involved in creating a labelled ER dataset in this domain mark the first of our original contributions:

- A publicly available unlabelled dataset to form the foundation for the labelled dataset
- A list of entity classifications and their definitions in the form of an annotation schema for salient entities
- A process by which annotations are collected, usually by distributing the task to human annotators on a crowd-sourcing platform
- A benchmark ER system trained on this data to act as a baseline of comparison for future work

Firstly, a dataset of job descriptions was sourced to form the foundation of the human labelled dataset. This process is detailed in §4.3.

4.3 UNLABELLED JOB DESCRIPTION DATA ACQUISITION

The dataset that was selected for the current research was the collection of job descriptions that was published on data sharing site Kaggle¹. This is available under a Creative Commons license.

Although this data contained several data fields pertaining to the job descriptions (such as *Job Title*, *Location*, *Contract Type*, and *Domain*), only the *Full Description* field for each was retained, since the task of identifying and extracting entities focuses solely on this text field. Summary statistics for the unlabelled dataset are shown in Table 4.1.

4.3.1 *Discrepancies of Job Description Style and Purpose*

Job descriptions in this dataset span a wide variety of domains, and include both technical and non-technical, permanent and part-time, entry-level and high-level positions.

¹ <https://kaggle.com/airiddha/trainrev1>

Number of items	244,768
Average Token Count per item	244.9
Token Count Standard Deviation	130.3
Token Count Minimum	4
Token Count First Quartile (Q_1)	154
Token Count Median (Q_2)	226
Token Count Third Quartile (Q_3)	312
Token Count Maximum	2,125

Table 4.1: Unlabelled Job Description Corpus Statistics.

The tone, structure, and style of job descriptions broadly fall into one of two categories. The first category employs a formal style, and tends to list responsibilities and requirements clearly with little to no superfluous text. The second category has a more informal style, often addressing the reader directly, and includes superfluous phrases and colloquialisms.

Job Description 4.1 is an example from this dataset that falls into the first category.

Job Description 4.1: An example Job Description with a formal tone.

A subsea engineering company is looking for an experienced Subsea Cable Engineer who will be responsible for providing all issues related to cables. They will need someone who has at least 10-15 years of subsea cable engineering experience with significant experience within offshore oil and gas industries. The qualified candidate will be responsible for developing new modelling methods for FEA and CFD. You will also be providing technical leadership to all staff therefore you must be an expert in problem solving and risk assessments. You must also be proactive and must have strong interpersonal skills. You must be a Chartered Engineer or working towards the qualification. The company offers an extremely competitive salary, health care plan, training, professional membership sponsorship, and relocation package.

In this Job Description, the responsibilities and skill requirements are listed clearly, for example: *developing new modelling methods, providing technical leadership, problem*

solving. Additionally, the qualification requirements and domain of industry are clear, and there are no unnecessary details.

By comparison, Job Description 4.2 is an example from the dataset that falls into the second category.

Job Description 4.2: An example Job Description with an informal tone.

Do you have a passion for Swimming? Can you motivate others to improve their skills? If the answer is yes and you have a ASA UKCC Level 2 or STA Full Qualification, this could be the job for you! We are looking to develop and grow our existing swimming instructor program for both children and adults and we are looking for a enthusiastic individuals [sic] to join our team on a self-employed basis and become part of our Swim Academy programme across our sites. We are actively seeking individuals who can teach swimming including a wide range of disciplines including Coaching, Triathlon, Adult and Child, Disabilities, Synchro, Waterpolo, and Diving. You will stand out because of your motivational and fun style of teaching, passion for Swimming and your ability to inspire people, ensuring pupils enjoy your sessions week on week. You will be a vibrant presence in our facilities and willing to go that extra mile. To apply for this opportunity you will need to have proof of your qualification and public liability insurance. An Enhanced Criminal Record Bureau CRB Disclosure is required for these posts.

Here, skill and qualification requirements are embedded in questions directed at the user (for example, *do you have a passion for **Swimming**?, can you **motivate others**...?*) and conversational phrasing (for example, *You will stand out because of your **motivational and fun style of teaching***).

The distinction between job descriptions written as technical documents and those written as marketing materials is made internally by Tribepad, who define the two categories of document as *Internal* and *External* job summaries respectively. However, in many online sources of job descriptions, the distinction is not made clearly and both styles are prevalent, and some documents are written employing aspects of both types. This represents a key complexity in the task of extracting salient entities from job descriptions; since both categories are valid job descriptions and evident in the corpus,

an automatic solution for entity extraction needs to be equally effective across both categories of job descriptions.

4.4 SCHEMA DEVELOPMENT

In the field of NLP, an annotation schema is a collection of entity classifications and their definitions which can be used to develop documentation and training materials that enable human annotators to perform a labelling task, ensuring consistency and accuracy in the process. Creating a labelled dataset requires some degree of human annotation. Given that the final dataset needs to be large enough to train machine-learning models, a number of human annotators are required, and annotation crowdsourcing platforms such as Amazon Mechanical Turk (Le and Mikolov, 2014) have been developed to streamline the process of recruiting, training, and distributing work to annotators.

The classifications of entities to be extracted in the data annotation task, and their definitions, were defined through an iterative process of performing the annotation task in conjunction with definitions from previous work (Gugnani and Misra (2020); Shi et al. (2020); Hoang et al. (2018)), and the European Skills, Competences, Qualifications and Occupations (Commission, Directorate-General for Employment, and Inclusion, 2019), which is part of the European Qualifications Framework (Khobreh et al., 2016). The definition of the *Skill* classification proposed here combines the *Skill*, *Knowledge*, *Language skills and knowledge*, and *Transversal skills* classifications used by ESCO. Our motivation to remove the distinction between these subcategories was largely due to the difficulty posed to non-experts in reliably identifying the correct category, and several borderline cases exist that could conceptually be included in multiple categories. For example, the current ESCO dataset² lists the term *use communication techniques* as a *Skill*, but the term *communication skills* is listed as a *Transversal skill*. By removing the distinction between subcategories of *Skill*, we eliminate the requirement of domain expertise for the annotation task.

Five distinct entity classifications were defined after a period of task development. However, these classes are difficult to define, and even with the definitions shown below, borderline cases exist. These ambiguities, and our methods of resolving them, are discussed later in this section.

² v1.1.1, accessed 2023

- **Skills**

- Examples: *computer programming, French, data analysis, Microsoft Word, leadership, unloading cargo, problem solving, honesty, graduate recruitment strategy*
- They are:
 - * Tasks that can be performed (for example, *unloading cargo*)
 - * Attributes pertaining to an individual (for example, *honesty*)
 - * Abilities that enable people to perform tasks (for example, *problem solving*)
- Includes domain-specific *hard skills* and domain-general *soft skills*
- Includes specific knowledge (for example, *understanding of marketing strategies*)
- May be validated with a qualification or experience, but these are not part of the Skill (for example, in the sentence *a Bachelor's degree and two years experience in data analysis*, only the term *data analysis* is a Skill)

- **Qualifications**

- Examples: *Bachelor's Degree, chartership, National Pool Lifeguard Qualification, three A-levels*
- They are official certifications obtained through taking a course or passing an exam or appraisal
- Includes driving licenses and security clearance

- **Experience**

- Examples: *2 years experience, minimum of 5 years experience*
- They are quantified by length of time
- Does not include what the experience is of or in - for example, in the sentence *this job requires at least 10 years of experience as a CEO*, only the words *at least 10 years of experience* are the Experience

- **Occupations**

- Examples: *Teaching Assistant, CEO, Data Analyst, Chef de partie*

- These are job titles
- Includes abbreviations and acronyms - for example, both *Chief Executive Officer*, and *CEO*, are Occupations

- **Domains**

- Examples: *aerospace, oil industry, education, human resources*
- These are areas of industry in which someone might have knowledge or experience

Worked examples showing labelled entities in sentences from job descriptions are included in Appendix A.1.1.

These entity classes and their definitions, along with a series of clarification questions and worked examples (Appendix A.1.1), were delivered to a number of human annotators over six iterations. In each iteration, modifications to the task, the entity classifications, the delivery mechanisms, and supporting materials were made in order to minimise inter-annotator disagreement. Human annotators and the work items assigned to them were different across testing rounds in order to ensure that the task was not affected by annotators recalling earlier versions of task instructions or items they had already annotated. Details of the annotation rounds are described in the following sections.

ROUND 1

In the initial testing round of annotations, 3 human annotators³ were given the first edition of labelling instructions and 5 items (referred to as a Human Intelligence Task (HIT)) for annotation. Each HIT required the annotator to read the given text, identify the salient entities contained within as defined in the ancillary instructional material, highlight the full span of each entity, and choose the relevant label for each entity span. DataTurks⁴ was used as the annotation platform due to its relatively simple setup procedure and easily navigable interface. Figure 4.2 shows one example HIT as it appeared to the annotators, and Figure 4.3 shows the same item labelled according to the concurrently supplied instructional material.

³ Annotators during initial testing rounds were colleagues of the author and recruited through word-of-mouth.

⁴ <https://github.com/DataTurks>

Workforce Classification	Private
# Classes	6
# Annotators	3
Platform	DataTurks
Item Description	Job Descriptions
# Work Items	5
Average # Sentences per Work Item	10.6
Total # Tokens	1,098
Instruction Details	Class Descriptions Only
Annotator F1	0.3589
κ with O label	0.3806
κ without O label	0.5885

Table 4.2: Summary statistics of the first round of the annotation process.

The screenshot displays the DataTurks platform interface. At the top, there is a navigation bar with a home icon, a 'Project Guidelines' button, and a 'Pre Tagged' status indicator. On the right side of the navigation bar, there are several checkboxes: 'AutoClose on Selection' (checked), 'Autolabel Same Text in Document' (unchecked), and buttons for 'Undo', 'Clear All', and a search icon.

Below the navigation bar, a text box contains the instruction: 'Click on the document and then drag to select text and select a label. More queries? See Demo Videos'. Below this instruction, a 'source' field is populated with 'Airiddha Kaggle Job Data'.

The main content area is titled 'Entities' and features a horizontal menu with filters: 'HARD SKILL', 'SOFT SKILL', 'QUALIFICATION', 'EXPERIENCE', 'OCCUPATION', and 'DOMAIN'. The 'EXPERIENCE' filter is currently selected. Below the filters, a large text box contains the following job description text:

Our client is an expanding asbestos company dealing with all asbestos and environmental disciplines. Currently they are seeking for a keen and enthusiastic Asbestos Quality Manager to work for them in the South East. The successful candidate must hold the Asbestos S****/CCP qualifications and must have a minimum of five years experience working within this role and within the asbestos arena. The role will involve attending operational meetings, dealing with asbestos UKAS accredited laboratory work, surveying and airmonitoring/air testing, managing, maintaining and developing Quality Managing Systems and UKAS accreditation to ISO17020 and ISO17025. Carrying out audits, inspections and quality control. Liaising with UKAS. Advising on training requirements. Complying with UKAS documentation and procedures. In general, you will be highly motivated, have excellent communication skills, both written and verbal, and have good IT skills, be meticulous and have good attention to detail. A competitive salary commensurate with experience, company car and along with other benefits awaits the successful candidate. Future Select specialize in recruiting staff in the Asbestos Industry. We currently have over a **** clients that are looking for candidates that have from 3 months experience to people with over 20 years in the industry. We have vacancies for Assistant Asbestos Surveyors, Basic Asbestos Surveyor, Lead Asbestos Surveyors, Asbestos Analysts, Asbestos Project Managers, Asbestos Junior Consultants, Asbestos Senior Consultants, Asbestos Laboratory Technicians, Asbestos Removal Contracts Managers, to Asbestos Director level positions. Salaries ranging ****k. We are recruiting for Asbestos clients in all regions throughout the UK. We are inundated with applications, will endeavour to get back in touch, however if you have applied to Future Select and you have not heard from us after a week, on this occasion, you will not have been successful. Your details will be saved on our system and you will be contacted in the future if a vacancy matches your skills

At the bottom of the interface, there are four navigation buttons: '← Previous (left)', '⏭ Skip (ctrl+q)', 'Move To Done (ctrl+enter)', and '→ Next (right)'.

Figure 4.2: Example unlabelled HIT on the DataTurks platform.

The screenshot displays the DataTutk interface. At the top, there are navigation icons (back, home) and a 'Project Guidelines' button. A 'Pre Tagged' label is visible. On the right, there are checkboxes for 'AutoClose on Selection' (checked) and 'Autolabel Same Text in Document' (unchecked), along with 'Undo' and 'Clear All' buttons. Below this is a text box with instructions: 'Click on the document and then drag to select text and select a label. More queries? See Demo Videos'. The 'source' is identified as 'Airiidha Kaggle Job Data'.

The main content area is titled 'Entities' and features a filter bar with categories: HARD_SKILL, SOFT_SKILL, QUALIFICATION, EXPERIENCE, OCCUPATION, and DOMAIN. The text below is a job advertisement for an Asbestos Quality Manager, with various phrases highlighted in different colors corresponding to the filter categories. For example, 'asbestos' is highlighted in blue (DOMAIN), 'Asbestos S****/CCP qualifications' in green (QUALIFICATION), 'minimum of five years experience' in yellow (EXPERIENCE), and 'Asbestos Quality Manager' in red (OCCUPATION). Other highlighted terms include 'attending operational meetings', 'dealing with asbestos UKAS accredited laboratory work', 'surveying and airmonitoring/air testing', 'managing, maintaining and developing Quality Managing Systems', 'UKAS accreditation to ISO17020 and ISO17025', 'Carrying out audits, inspections and quality control', 'Liaising with UKAS', 'Advising on training requirements', 'Complying with UKAS documentation and procedures', 'motivated', 'communication skills', 'IT skills', 'meticulous', 'attention to detail', 'recruiting staff', 'Asbestos Industry', 'Assistant Asbestos Surveyors', 'Basic Asbestos Surveyor', 'Lead Asbestos Surveyors', 'Asbestos Analysts', 'Asbestos Project Managers', 'Asbestos Junior Consultants', 'Asbestos Senior Consultants', 'Asbestos Laboratory Technicians', 'Asbestos Removal Contracts Managers', and 'Asbestos Director'.

At the bottom of the text area, there are navigation buttons: '← Previous (left)', '⌕ Skip (ctrl+q)', 'Move To Done (ctrl+enter)', and '→ Next (right)'.

Figure 4.3: Example labelled HIT on the DataTutk platform. Labels are for illustrative purposes only, and do not necessarily reflect the correct entity spans or classifications.

Results from Round 1 are shown in Table 4.2. Although Cohen's κ , or Fleiss' κ when adjusted for multiple raters, is a widely used measure of IAA (see §2.7.2 for an explanation of IAA), there have been several issues raised regarding its application in specifically entity annotation tasks (Hripcsak and Rothschild, 2005), in particular in cases where class distribution is unbalanced and where unannotated tokens (that is, tokens that do not fall under any given entity classification and are consequently left without annotation, including stop words such as *and* and *the*) are more prevalent than annotated tokens. In these cases, κ is calculated twice under two separate conditions: evaluating all tokens in the data, and evaluating only the annotated tokens in the data.

Pairwise F_1 calculated on annotated tokens only has been suggested as a better measure for agreement in annotation labelling tasks (Deleger et al., 2012). We thus compute the micro F_1 on annotated tokens as the focal method of IAA, but Fleiss' κ statistics are provided to give additional insight.

Using Landis and Koch (1977)'s interpretation of the κ statistic (see Table 2.2), results from Round 1 show *Fair* agreement between annotators. However, annotators reported feeling overwhelmed by the density of the work item (on average, 10.6 sentences per item), which may have led to poorer engagement with the task and consequently a greater error rate. To address this feedback, work items were reduced from *full job descriptions* to *individual sentences from job descriptions*. Although this alleviates the item density problem, it was necessary to investigate the effect this had on errors introduced by the lack of context. It is possible that annotators used information from preceding sentences when classifying a given sentence, and by removing this context we may be inadvertently reducing annotator performance.

Round 2, detailed in the next section, was conducted to investigate this effect.

ROUND 2

Results from Round 2 are shown in Table 4.3. Including splitting work items by sentence to reduce work item density, three changes in total were made between Rounds 1 and 2 to facilitate annotation and reduce task ambiguity. The annotation platform Amazon SageMaker Ground Truth⁵ was used as an alternative to DataTurks, since the live task was planned to be hosted on the SageMaker platform, and keeping the interface consistent between the testing phase and live data collection phase would make test

⁵ <https://aws.amazon.com/sagemaker/groundtruth/>

Workforce Classification	Private
N classes	6
<i>n</i> annotators	3
Platform	Amazon SageMaker
Item Description	Sentences from Job Descriptions
# Work Items	53
Average # Sentences per Work Item	1
Total # Tokens	1,177
Instruction Details	Class Descriptions, 3 Worked Examples
Annotator F1	0.4064
κ with O label	0.5443
κ without O label	0.4729

Table 4.3: Summary statistics of the second round of the annotation process.

results more representative of live results. Examples of unlabelled and labelled [HITs](#) on the SageMaker platform are shown in [Figures 4.4](#) and [4.5](#) respectively. Secondly, worked examples were created and included in the instruction document to improve the clarity of task instructions and address some consistent annotation errors.

Instructions Shortcuts

Highlight entities in job descriptions.

ISEB qualified to Foundation level or able to demonstrate a strong understanding of software testing methodology.

Labels

- 1 Hard Skill
- 2 Soft Skill
- 3 Qualification
- 4 Experience
- 5 Occupation
- 6 Domain

No entities to label

Figure 4.4: Example unlabelled HIT on the Amazon SageMaker platform.

Instructions Shortcuts

Highlight entities in job descriptions.

QUA × HAR ×

ISEB qualified to Foundation level or able to demonstrate a strong understanding of software testing methodology.

Labels

- 1 Hard Skill
- 2 Soft Skill
- 3 Qualification
- 4 Experience
- 5 Occupation
- 6 Domain

No entities to label

Figure 4.5: Example labelled HIT on the Amazon SageMaker platform.

Although IAA increased from Round 1 to Round 2, κ statistics failed to reach levels beyond the lower class boundary of *moderate* agreement (Landis and Koch, 1977). This may, in part, have been due to the ambiguity regarding the distinction between the entity classes *Hard Skill* and *Soft Skill*. Although the supplied annotation instructions defined *Hard Skill* as a *domain-specific skill that one could attain a qualification for*, and a *Soft Skill* as a *domain-general skill that could only be supported with anecdote or experience*, annotators reported uncertainty when deciding whether a term belonged to either category. For example, the terms *organise and attend site visits* and *contribute to quality improvement projects* were classified as *Hard Skills* by one annotator and *Soft Skills* by another. In other cases, uncertainty regarding the definition of terms makes it difficult to classify as either *Hard Skill* or *Soft Skill* - for example, the term *performing FAT under formal witnessed conditions* is not simple to classify given that *FAT* is an acronym of *Factory Acceptance Tests* that only annotators with specific knowledge of that area of industry would know.

The distinction between the *Hard Skill* and *Soft Skill* classes was originally made in line with previous work in the field of skill extraction (Qin et al., 2020), since the distinction serves as an indicator of salience; typically, *Hard Skills* (for example, *Python*) weighted more heavily in terms of importance to a recruiting agent compared with *Soft Skills* (for example, *communication skills*), and consequently the two entity classes should not be treated with the same weighting in an applicant profile-job description matching solution. However, there are alternative methods that can be employed to evaluate the relative importance of a given skill term that do not require an explicit distinction between *Hard Skills* and *Soft Skills* at the annotation level, such as the use of deep learning methods and market supply signals using engineered features to model skill salience (Shi et al., 2020). To minimise task difficulty and maximise IAA, the distinction between the *Hard Skill* and *Soft Skill* entity classifications was removed in subsequent rounds of annotation task development.

ROUND 3

Round 3 results, shown in Table 4.4, saw slight overall improvement to IAA. Subsequently, a series of informal interviews were conducted with the development annotators which provided some insight into the rationalisation behind inter-annotator disagreement. In addition to the informal interviews, deeper text analysis of

Workforce Classification	Private
N classes	5 (merged <i>Hard Skill</i> and <i>Soft Skill</i>)
n annotators	5
Platform	Amazon SageMaker
Item Description	Sentences from Job Descriptions
# Work Items	43
Average # Sentences per Work Item	1
Total # Tokens	1,037
Instruction Details	Class Descriptions, 5 Worked Examples
Annotator F1	0.4116
κ with O label	0.4442
κ without O label	0.6937

Table 4.4: Summary statistics of the third round of the annotation process.

Gold Standard	We seek applications from talented engineers with significant experience in Process Engineering within the nuclear sector.
Annotator #1	We seek applications from talented engineers with significant experience in Process Engineering within the nuclear sector.
Annotator #2	We seek applications from talented engineers with significant experience in Process Engineering within the nuclear sector.
Annotator #3	We seek applications from talented engineers with significant experience in Process Engineering within the nuclear sector .
Annotator #4	We seek applications from talented engineers with significant experience in Process Engineering within the nuclear sector .
Annotator #5	We seek applications from talented engineers with significant experience in Process Engineering within the nuclear sector .

Table 4.5: An example of disagreement in the third round of the annotation task.

Classification colours: **Skill**, **Occupation**, **Domain**, **Experience**

Problem	Solution
Annotators are not reading the instruction document, and are using their own class definitions for the classification task.	→ Include the phrase <i>Please read the instructions document carefully</i> in all possible locations, and add a clarifying paragraph to appreciate that terms may conflict with annotators' own definitions, and to use task definitions.
Annotators are over-classifying in work items where no entities exist, because "there must be one in here somewhere".	→ Include a worked example that contains no entities to show that this is a possible scenario.
Annotators are ignoring terms that are not relevant to the job itself.	→ Include a <i>clarification question</i> to explain that all entities should be classified.
Annotators are including the application of the Skill in the classification.	→ Include a <i>clarification question</i> to explain where classification should start and end.

Table 4.6: A few examples of identified problems causing inter-annotator disagreement in the annotation task, and formulated solutions to address them.

disagreement was performed in order to understand its causes, with a view to compiling a list of actionable changes to mitigate disagreement. An example of a disagreement from this round is shown in Table 4.5.

Where consistent disagreements between annotators were identified, an interpretation of the cause of the discrepancy was formulated, and for each identified issue, a solution was developed to address it in the form of amending the task instructions and ancillary documentation. A shortlist of these is shown in Table 4.6. These changes were made prior to Round 4 of the annotation development task.

ROUND 4

Results from Round 4, shown in Table 4.7, outperformed those of previous rounds, and concluded the private testing phase. Disagreement between annotators in annotation tasks on some level is inevitable, no matter how clear the definitions and supporting material, and a method for disagreement handling is required. There are several

Workforce Classification	Private
N classes	5
n annotators	5
Platform	Amazon SageMaker
Item Description	Sentences from Job Descriptions
# Work Items	60
Average # Sentences per Work Item	1
Total # Tokens	1,056
Instruction Details	Detailed Class Descriptions, 8 Worked Examples
Annotator F1	0.5145
κ with O label	0.6126
κ without O label	0.6500

Table 4.7: Summary statistics of the fourth round of the annotation process.

methods of addressing disagreement, such as majority agreement, simply removing the items containing disagreements, or probabilistic aggregation methods in which annotators are identified as *trustworthy* or otherwise on gold-standard tasks and weighting their annotations accordingly (Hovy et al., 2013). Alternatively, rather than extracting the single objective classification for each entity through agreement resolution methods, it may be possible to learn a classifier directly from the annotations by assigning a distribution score to each label (Rodrigues and Francisco Pereira, 2017). This is discussed in more detail in §4.8.

ROUND 5

Round 5 began the first of two rounds of testing with Workers recruited through the Amazon Mechanical Turk platform. In total, 20 Workers completed the annotation task that was composed of the work items from Rounds 3 and 4 combined.

Results for Round 5 are shown in Table 4.8. Agreement statistics here were poor, and disagreement was more prevalent compared to the private rounds of testing. There could be a number of reasons for this - one possibility is that, since the SageMaker platform does not allow any kind of Worker screening, nor is there any option of

Workforce Classification	Public
<i>n</i> annotators	20
Platform	Amazon SageMaker
Pay per HIT	\$0.48
Required Qualifications	None (not available)
Annotator F1	0.0918
κ with O label	0.2124
κ without O label	0.1281

Table 4.8: Summary statistics of the fifth round of the annotation process.

Workforce Classification	Public
<i>n</i> annotators	29
Platform	Amazon Mechanical Turk
Pay per HIT	\$0.08
Required Qualifications	Number of HITS Approved > 5,000, Approval Rate > 95%
Annotator F1	0.0400
κ with O label	0.1849
κ without O label	0.1828

Table 4.9: Summary statistics of the sixth round of the annotation process.

rejecting an individual Worker's contribution, Workers are not incentivised to follow the instructions nor make any effort to provide a considered contribution. This theory is supported by several Workers whose contributions contain no extracted entities, despite the presence of several entities in the assigned items.

For this reason, the annotation platform SageMaker was rejected in favour of the standard AMT platform which offers several possibilities for Worker screening and evaluation.

ROUND 6

Round 6 of the annotation task was deployed on AMT using a bespoke interface that was developed to closely resemble the platform used in the latter stages of private development. An interactive example of this interface can be found in the repository in which the published data is located⁶.

Participation in this round was limited to AMT Workers who had at least 5,000 HITs approved in previous tasks on the platform, and had achieved at least a 95% approval rate on their contributions. These requirements are in line with standard practice for ensuring data quality (Peer, Vosgerau, and Acquisti, 2014).

Results for Round 6 are shown in Table 4.9. Although IAA remained comparatively low, deeper analysis of individual Worker response showed that there were some capable annotators in the workforce who had made a useful contribution to the task, and others who had evidently not read the instructions or had severely misunderstood the task.

One method of ensuring that only Workers who have read and understood the instructions contribute to the live task is by including a *qualification round*; a series of work items for which the gold standard has been established and can therefore be used to appropriately evaluate the contribution of the Worker. Workers who take part in this qualification round and score above a given threshold are given a bespoke qualification on the AMT platform and can be invited to take part in the live task.

However, there is no academic consensus regarding an appropriate accuracy level to require from Workers, especially for ER tasks where there are many methods for evaluating worker performance. The higher the threshold is set, the greater the confidence one has in the quality of the contribution of the Workers, but higher thresholds limit the diversity of the Worker pool and can make the collection of data dependent on the sustained efforts of a small workforce, which leads to overall slower data collection rates.

§4.5 details the process by which we identified an accuracy threshold through experimentation and investigation into the relationship between average Worker accuracy and resultant Entity Recognition model performance.

⁶ <https://github.com/acp19tag/skill-extraction-dataset>

4.5 ACCURACY THRESHOLD IDENTIFICATION

Experimentation was performed using a standard dataset used in ER tasks; the CoNLL-2003 Shared Task: Language Independent Named Entity Recognition (NER) (Sang and Meulder, 2003).

The premise of this investigation was that recently developed ER models are able to learn directly from noisy human annotation, eliminating the need for label aggregation (Rodrigues and Francisco Pereira, 2017), and that examining the relationship between Worker performance (varied by artificially inducing noise) and resultant model performance may yield an appropriate threshold to require of Workers before admitting them to contribute to the live corpus.

Two distinct types of noise were investigated based on the cause of annotator misclassification: *random* noise, where annotators make random errors (that is, where any incorrect classification is equally likely to be selected); and *systematic* noise, where annotators make consistent errors (for example, by consistently misclassifying class *A* as class *B*). We also investigate the effect of reducing noise by artificially correcting annotated labels to simulate higher performance. This method can only be implemented to simulate *random* de-noise, and doing so allows us to observe the effect of annotators performing at higher levels of accuracy than observed in the data, which is the likely consequence of increasing the minimum accuracy level required of Workers.

We induced both forms of noise and random de-noise from proportions of 0 to 1 in increments of 0.02 which yielded simulated corpora with average Worker accuracy (versus gold standard) between 0 and 100%. A separate model was trained on each simulated corpus using a CNN with the Crowd Layer proposed by Rodrigues and Francisco Pereira (2017) (discussed in more detail in section 3.4.3), and we observe the relationship between average Worker accuracy and resultant model performance. Code for reproducing these experiments is made publicly available⁷.

Model F_1 is shown at varying levels of Worker accuracy in Figures 4.6 and 4.7. We observe a lower threshold of Worker performance at around 40% Worker accuracy, below which resultant model performance is poor (< 50 model F_1). This is especially prevalent when inducing systematic noise (See Figure 4.7), which shows a steep drop in model performance when Worker accuracy is below 40%. Model performance increases

⁷ https://github.com/acp19tag/conll_noise_induction

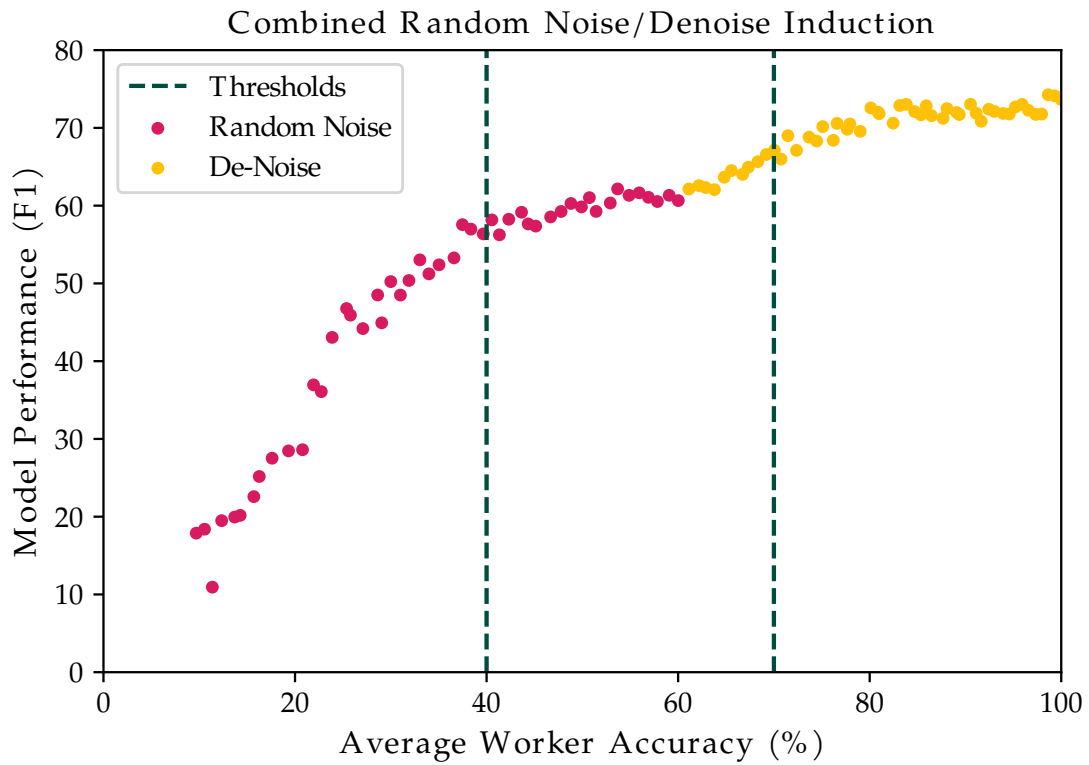


Figure 4.6: Graph to show the relationship between average Worker accuracy (%) and resultant trained model F_1 after artificially inducing and removing random noise in Worker annotations.

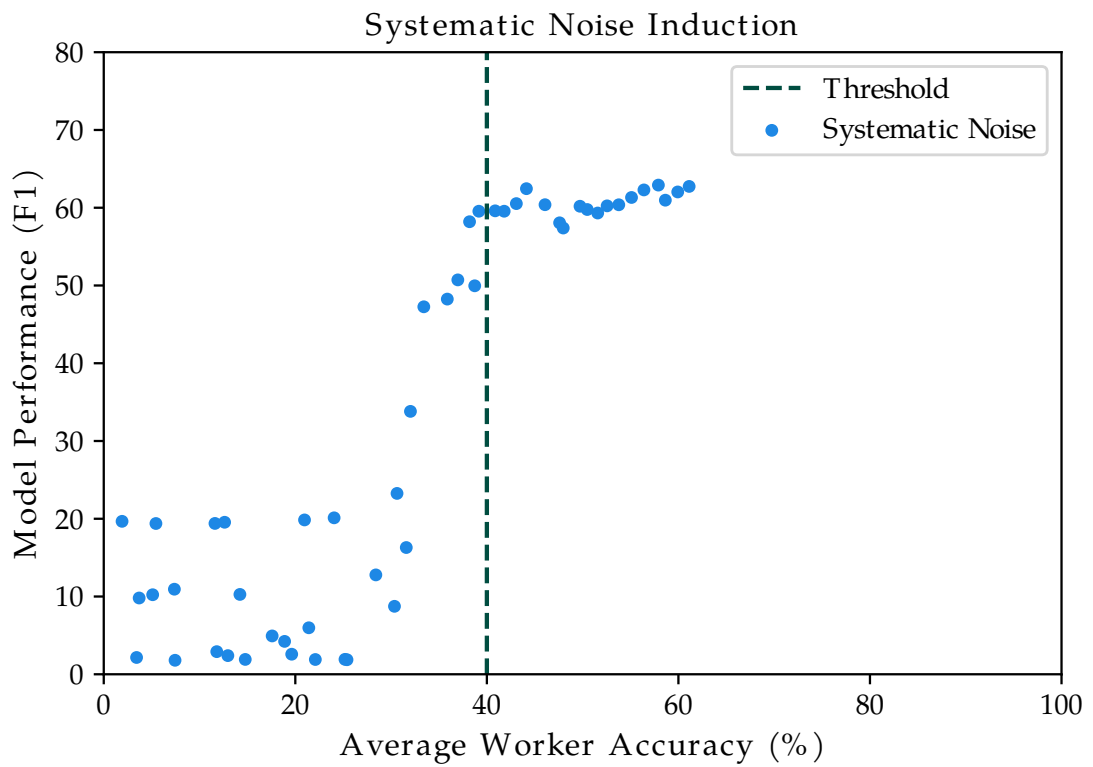


Figure 4.7: Graph to show the relationship between average Worker accuracy (%) and resultant trained model F_1 after artificially inducing systematic noise in Worker annotations.

steadily with increasing Worker accuracy when noise is artificially reduced above actual Worker performance (shown in yellow in Figure 4.6), and begins to plateau between 70% and 80% Worker accuracy. This guided our decision to use 70% as our threshold.

An additional consideration is that actual Worker accuracy increases with increased observations; correctly identifying no entities in one HIT may yield perfect Worker accuracy, but this does not provide any evidence that the Worker has sufficiently understood and applied the entity classifications as defined by the schema. For this reason, we also required Workers to have annotated at least 100 of the 140 non-O label tokens in the Qualification set in order to reasonably evaluate their performance. Although Workers were encouraged to complete all HITs in the Qualification set, many Workers did not.

4.6 CORPUS DEVELOPMENT

Results from the Qualification task are shown in Figure 4.8. Of the 178 Workers who took part in the Qualification task, 39 achieved an accuracy greater than 70% on the qualification task and had annotated more than 100 tokens, and consequently only these Workers were invited to contribute to the live task.

Annotations were collected over a three week period in July 2021, and collection terminated when all 10,000 sentences had each received annotations from two separate Workers. Task allocation was managed by the AMT platform, and tasks were assigned ad hoc to qualified Workers with no limit imposed on how many items each individual Worker could accept.

Communication with active Workers during this period was monitored, but no queries were received concerning the task instructions or ancillary material.

4.7 CORPUS STATISTICS

Table 4.10 lists general statistics of the annotated corpus, and Table 4.11 shows the distribution of class labels in the annotated corpus after aggregation to yield one label per token. Similarly, Table 4.12 shows the distribution of class labels for the test set generated by the author of the annotation schema. We observe a similar distribution in both corpora.

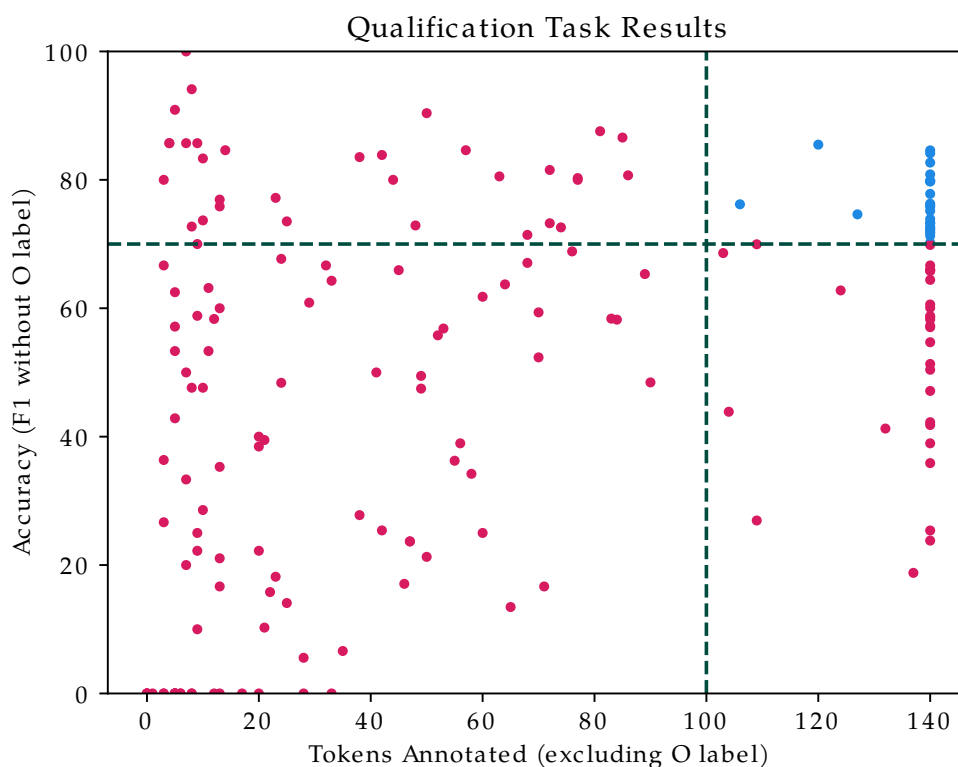


Figure 4.8: Graph to show the Results for the Qualification Task. Chosen criteria thresholds of 100 annotated tokens and 70% accuracy are shown in green. Workers who met these criteria are shown in blue, and Workers who did not are shown in red.

Sentences	10,000
Tokens	245,606
Avg. tokens per sentence	24.6
Annotation spans (post aggregation)	18,617
Annotated tokens (post aggregation)	79,826
Avg. tokens per annotation	4.3
Number of independent Annotators	25

Table 4.10: Annotated corpus statistics.

Label	Frequency	Proportion
Skill	66,732	28.56%
Occupation	6,117	2.62%
Domain	3,705	1.59%
Experience	1,328	0.57%
Qualification	1,944	0.83%
None	153,802	65.83%
Total	233,628	

Table 4.11: Class distribution for the live, aggregated corpus (one label per token).

Label	Frequency	Proportion
Skill	2,136	25.19%
Occupation	306	3.61%
Domain	100	1.18%
Experience	29	0.34%
Qualification	68	0.80%
None	5,839	68.87%
Total	8,478	

Table 4.12: Class distribution for the test set.

Cohen's κ on all tokens	0.49
Cohen's κ on annotated tokens only	0.73
Krippendorff's α	0.55
F_1 on annotated tokens only	0.90

Table 4.13: IAA on the live corpus, calculated by averaging pairwise comparisons between all combinations of annotators where both annotators labelled a shared item.

As discussed in section 4.4, although Cohen's κ (see section 2.7.2) is the standard measure of inter-annotator agreement, there have been several issues raised regarding its application in entity annotation tasks (Hripcsak and Rothschild, 2005), especially in cases where class distribution is unbalanced and where unannotated tokens are much more common than annotated tokens. In these cases, Cohen's κ is calculated twice under two separate conditions: evaluating all tokens in the data, and evaluating only the annotated tokens in the data.

Typically, including *None* labels in the calculation would show an inflated value of κ since the *None* label is by far the most prevalent, and the high frequency of cases in which neither annotator has labelled a token tends to raise the observed agreement level. However, this is not the case in our data. Distribution of Worker contribution is neither uniform nor Gaussian, and the intersection of work between the majority of worker pairs is small (< 10 sentences or < 250 tokens). Since κ is calculated between each pair of annotators that contributed to at least one shared item and averaged across all pairs, there are several pairs of annotators that show an indeterminate κ agreement; if both annotators in a given pair have identified no entities across all reviewed tokens, the expected agreement p_e will be equal to 1, and κ will be indeterminate with a denominator of 0. For the κ statistics shown here, a case of indeterminate kappa between annotator pair i, j is interpreted as perfect agreement ($\kappa_{ij} = 1$).

Pairwise F_1 on annotated tokens only has been suggested as a better measure for agreement in ER tasks (Deleger et al., 2012). We thus compute the micro F_1 on annotated tokens as the focal method of IAA, but Cohen's κ and Krippendorff's α statistics are provided to give additional insight (see Table 4.13).

Table 4.14 shows the top 5 most frequent entities for each class after label aggregation, and Figure 4.9 shows the distribution of token counts for annotated spans for each

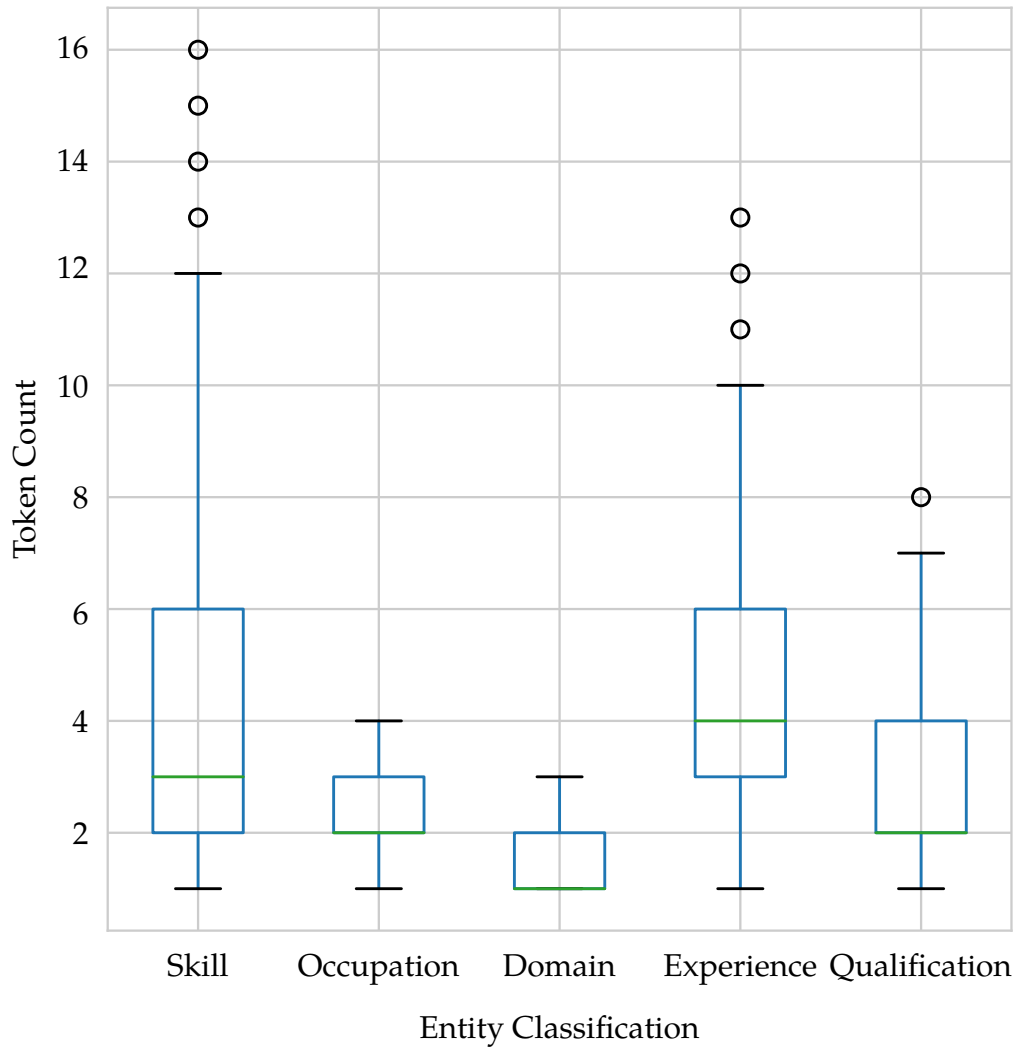


Figure 4.9: Annotated span token count distribution by class in the live corpus after label aggregation.

Skill	Occupation	Domain	Experience	Qualification
experienced	sous chef	sales	experience	graduate
communication	general manager	marketing	2 years experience	driving licence
experience	manager	business	5 years	acca
enthusiastic	sales executive	recruitment	2 years	aca
communication skills	project manager	it	at least 2 years	degree

Table 4.14: Frequent entities of each class in the live corpus after label aggregation.

class after label aggregation. As expected, the most frequent Skill terms appear to be terms considered *Soft Skills* (for example, *communication*), which tend to be evident across job descriptions of several domains. The distribution of token counts within Skill spans appear to be positively skewed, and token counts greater than the modal single-token count become decreasingly prevalent. Assuming that class annotations are largely correct, this indicates that Skill terms are typically only a few tokens in length, but longer token spans of up to around 17 tokens are still evident (for example, *ensure all risk information across the group is consistent and reported in a clear and timely fashion*). The existence of spans with large token counts further supports the need for span identification methods more sophisticated than simple word banks and keyword matches; with increasing numbers of tokens within spans, it is less likely that an exact match will be found in the corresponding span bank.

Token count distributions for the Occupation and Domain classes are similar in that lengths tend to vary by only 1 or 2 tokens about the modal count. Likewise, the Experience and Qualification classes show similar positively skewed token count distributions. It is possible that some of the spans that comprise several tokens have been misclassified as a single entity. For example, in the Qualification span *AAT, ACA, CIMA, or ACCA part qualification or working toward one*, this multi-token span should, according to our proposed schema, be a sequence of four independent single-token Qualification entities: *AAT, ACA, CIMA, and ACCA*.

4.8 ENTITY RECOGNITION METHOD DEVELOPMENT

All entities were labelled using the BIO scheme, where each labelled entity is prefixed with either a *B*, denoting the beginning of a span, or *I*, denoting that entity is inside a span. Tokens not assigned a label are assigned an *O* label. Although error is inevitable in human labelling tasks, it is feasible to mitigate some aspects. Preliminary analysis suggested that there were three sources of noise that could be mitigated prior to model training (referred to here as *preprocessing*): label aggregation; reclassification of *Experience* spans; and splitting multi-term spans.

Preprocessed data is included alongside raw data in the public repository associated with this research project⁸.

4.8.1 Data Preprocessing

LABEL AGGREGATION

There are several established methods of label aggregation, such as majority agreement, simply removing items containing disagreements, or probabilistic aggregation methods in which annotators are identified as *trustworthy* or otherwise on gold-standard tasks and weighting their annotations accordingly (Hovy et al., 2013). Alternatively, rather than extracting the single objective classification for each entity through agreement resolution methods, it is possible to learn a classifier directly from the annotations by assigning a distribution score to each label (Rodrigues and Francisco Pereira, 2017).

Since each token is annotated by two independent Workers, a simplification of the method of Hovy et al. (2013) was used for disagreement, where labels were assigned preferentially from higher-performing Workers inferred from qualification task results.

RECLASSIFICATION OF EXPERIENCE SPANS

Preliminary analysis yielded a number of insights. According to the schema, *Experience* spans must be quantified by length of time (for example, *2 years experience*). A number of spans classified as *Experience* did not meet this criteria (for example, *experience managing clients*), but did meet the criteria for the *Skill* classification.

⁸ <https://github.com/acp19tag/skill-extraction-dataset>

A *re-classification* step was therefore added to the preprocessing pipeline in order to identify and correct these errors. Regular expression and *inflect*⁹ Python packages were utilised to identify all spans that did not contain an expression of time (in word or number form) and reclassify the entire span from *Experience* to *Skill*. This reduced the number of Experience spans from 239 to 144 (40% reduction), which were manually checked. No other classes were affected.

SPLITTING MULTI-TERM SPANS

A second finding from preliminary analysis was that annotators tended not to split lists of entities into separate spans, choosing instead to identify everything included in the list as one single span of the relevant entity type. For example, the sequence *Asbestos Surveyors, Lead Asbestos Surveyors, Asbestos Analysts* was annotated as one single entity, whereas this should be three distinct entities with commas denoting the boundaries. This was addressed clearly in one of the worked examples contained in the ancillary materials, so it is possible that this particular error was made in order to reduce overall **HIT** time commitment rather than due to a fundamental misunderstanding of the task instructions.

The correct splitting of entities is important for our task for two reasons. Firstly, it represents an issue for model training, in that if the training data does not reflect the correct distinction between multiple consecutive entities of the same type, it is unlikely that the resultant model will be able to, and will achieve poor performance when evaluated on the test set which features accurate entity separation.

Secondly, the intended use of a system trained to identify and extract entities from job descriptions is for feature extraction in a larger system developed to match applicant profiles and job descriptions (**RQ2**). For this purpose, it is important that entities are discrete to ensure that each are evaluated independently to more accurately represent the requirements of a job from its description or an applicant from their profile.

All instances of punctuation were re-classified with the *None* label, and in cases where this split an annotated span, the following tokens became the start of a new span. Affected items were then manually checked to ensure legibility. The class distribution for the data after the preprocessing steps is shown in Table 4.15.

⁹ <https://github.com/jaraco/inflect>

Label	Token Freq.	Token Prop.	Span Freq.	Span Prop.
Skill	65,632	28.09%	13,663	69.98%
Occupation	5,964	2.55%	2,735	14.01%
Domain	3,628	1.55%	2,284	11.70%
Experience	800	0.34%	248	1.27%
Qualification	1,716	0.73%	595	3.05%
None	155,888	66.72%	-	-
Total	233,628		19,525	

Table 4.15: Class distribution for the preprocessed data.

4.8.2 Baseline CRF Model

CRF (Lafferty, Mccallum, and Fernando Pereira, 1999), as described in §2.5.1.1, was implemented as the baseline model for this task for a number of reasons. Firstly, **CRF** models are well-suited for sequential data, which is the case in **ER** tasks which involve identifying entities in sequences of text, and **CRF** models are able to incorporate the dependencies between neighbouring words when generating predictions. Furthermore, **CRF** models are able to capture both local contexts (such as features pertaining to a single word) as well as global contexts (such as features that consider the sequence of words), which aids the modelling of dependencies between words and captures long-range patterns in input sequences.

A visualisation of the **CRF** model for the **ER** task is shown in Figure 4.10.

CRF models allow for the incorporation of various local and global features which can improve model performance. For this experiment, the **NLTK**¹⁰ method of feature preparation was used, which includes syntactic features such as **POS** tags as well as morphological features of words along with the same features for neighbouring words.

The **CRF** model was trained over 100 epochs using L1 and L2 regularization coefficients found during parameter optimisation through Randomized Search.

¹⁰ <https://www.nltk.org/>

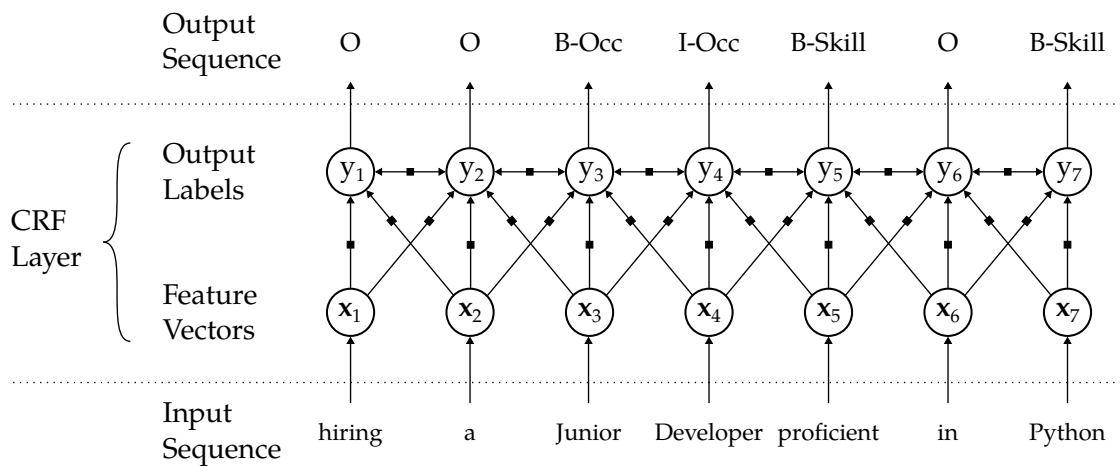


Figure 4.10: A visualisation of the CRF model for ER. The output sequence shown includes entity classifications with their BIO tags, for example *B-Occ* refers to the *Beginning* of an *Occupation* entity span.

4.8.3 BiLSTM-CRF Model

The BiLSTM-CRF model (Z. Huang, Xu, and K. Yu, 2015) is essentially an augmentation of the previously described CRF model with a BiLSTM model. The underlying principle behind this augmentation is that the BiLSTM leverages deeper language meaning while the subsequent CRF layer handles the sequential logic of the labelling process.

The BiLSTM-CRF model is visualised in Figure 4.11.

On the Computational Natural Language Learning (CoNLL) dataset NER task (Sang and Meulder, 2003), the BiLSTM-CRF was shown to marginally outperform CRF, LSTM, BiLSTM, and LSTM-CRF models (Z. Huang, Xu, and K. Yu, 2015), which makes this a suitable model for inclusion in this experiment as an augmentation on the established baseline CRF model.

The keras¹¹ implementation of BiLSTM-CRF was used in the current experiment, which featured the following layers:

1. An input layer with dimensionality equal to the length of the maximum sequence length in the training data
2. A 300-dimensional word embedding layer

¹¹ <https://keras.io/>

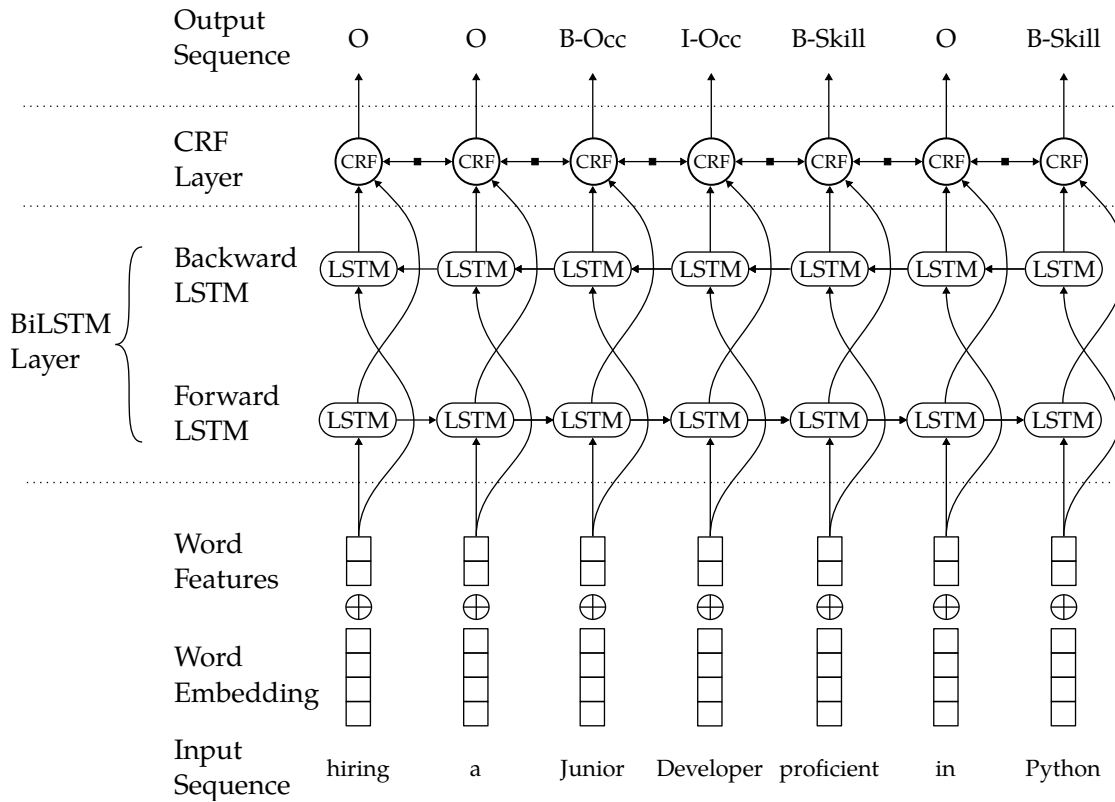


Figure 4.11: A visualisation of the BiLSTM-CRF model for ER. The output sequence shown includes entity classifications with their BIO tags, for example *B-Occ* refers to the *Beginning* of an *Occupation* entity span.

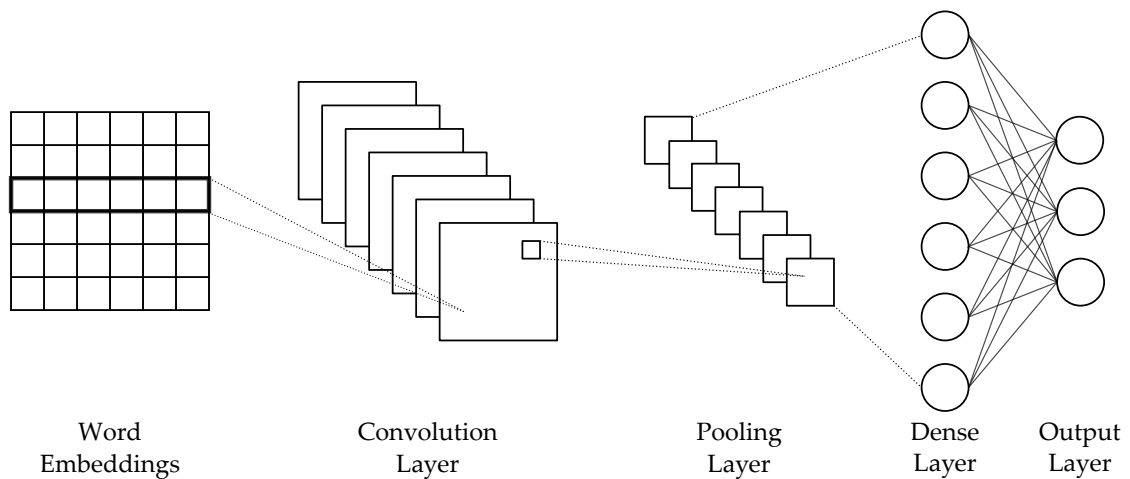


Figure 4.12: A visualisation of the CNN model for ER.

3. An LSTM layer
4. A second LSTM layer with bidirectionality configured
5. A Time Distributed dense layer with ReLU activation
6. A CRF layer

4.8.4 Convolutional Neural Network

CNN models (LeCun, Bengio, and Hinton, 2015) are able to capture local patterns and features in data without the need for explicit feature engineering, and have been shown to perform well at NER tasks (X. Zhang and LeCun, 2016). Typically, CNN models are large, multi-layered NNs that recognise and extract high level features from data. The CNN model is visualised in Figure 4.12.

The particular issue that we aim to address with CNN is that of the noise associated with disagreement between annotators. As described in §4.8.1, models tend to rely on label aggregation methods to produce a single output label for each token in the data. In our experiments, we use the Qualification task results to guide label aggregation; the annotations from the Worker who scored more highly on this task are accepted preferentially. However, in applying label aggregation in cases of annotator disagree-

ment, we are rejecting the information contained in the annotations supplied by the second annotator; If the cause of the disagreement was a mistake or misunderstanding of the entity classes on behalf of the second annotator then applying label aggregation in this way correctly eliminates noise that would otherwise harm the model training process. However, if the cause of disagreement was that an entity span was ambiguous, or one annotator leveraged domain knowledge to guide classification (for example, that the term *certhe* is an acronym for *certificate of higher education*, and is therefore a *Qualification*), then applying label aggregation removes this insight.

As implemented in §4.5, the Crowd Layer (Rodrigues and Francisco Pereira, 2017) was appended to the CNN model in this setting, which enables the model to learn directly from noisy annotations rather than relying on label aggregation methods for the ER task. The Crowd Layer is discussed in more detail in section 3.4.3.

The architecture used in the current experiment was constructed to match the architecture used by Rodrigues and Francisco Pereira (2017):

1. a 300-dimensional word embedding layer initialised with pre-trained weights of 6B Global Vectors for Word Representation (GloVe) (Pennington, Socher, and Christopher D Manning, 2014)
2. a 5x5 convolutional layer with 512 features
3. a GRU cell with a 50d hidden state
4. a fully connected layer with softmax activation
5. a crowd layer, which is removed during evaluation

4.8.5 Transformer-Based Models

An explanation of transformer models such as BERT is detailed in §2.5.1.2. Training a transformer model from first principles requires a considerable amount of computational resources, time, and data. Fine-tuning a pre-trained model, significantly reduces the time and resource requirement, and can often converge faster on small datasets (Radford and Narasimhan, 2018). Fine-tuned pre-trained BERT models have achieved state-of-the-art (SOTA) performance on many NLP tasks, including NER (Luoma and Pyysalo (2020); Xinyu Wang et al. (2021) Şapcı et al. (2023)) largely due to

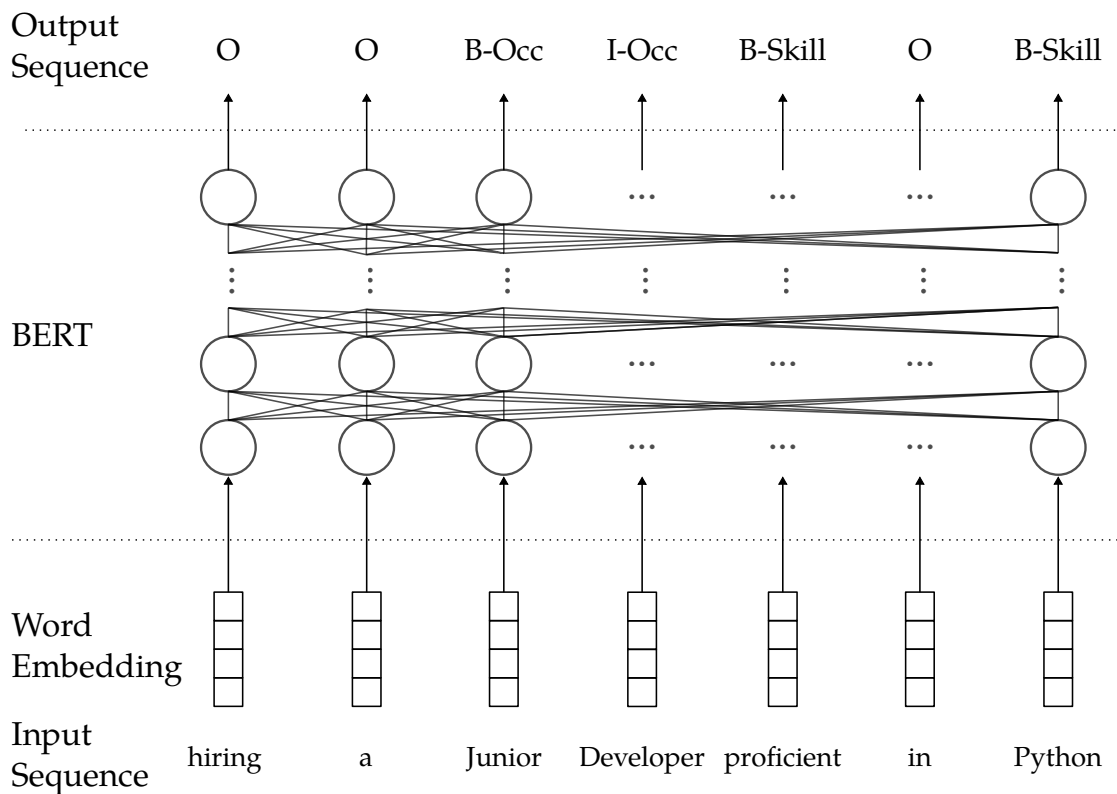


Figure 4.13: A visualisation of the BERT model for ER.

their use of contextual word embeddings (§2.4.2) and bidirectional attention mechanisms.

The BERT model applied to the ER task is visualised in Figure 4.13.

Five configurations for BERT were included in this experiment: the *cased* and *uncased* variants of the *base* model, the multilingual variant, and two distilled variants *DistilBERT* and *ALBERT*. These variants are explained in the following sections.

BERT, BASE, UNCASSED

The BERT-base-uncased model (Devlin et al., 2019) is the most commonly applied variant of BERT¹², and can be fine-tuned for ER tasks. The *uncased* attribute indicates that text is converted to lowercase before tokenisation, which effectively halves the size of the

¹² When considering historic download counts from HuggingFace <https://huggingface.co/bert-base-uncased>

vocabulary and has been shown to achieve more stable performances relative to *cased* variants (M. Jiang et al., 2020).

BERT, BASE, CASED

The BERT-base-cased model (Devlin et al., 2019) is identical to the uncased version, but lowercase conversion is not applied prior to tokenisation. In tasks where case information is helpful (for example, NER), the cased variant of BERT often outperforms the uncased variant, which is why this variant was selected for inclusion in this experiment.

BERT, BASE, MULTILINGUAL

The BERT-base-multilingual model (Devlin et al., 2019) was pretrained on the 104 languages that had the largest Wikipedia corpora. Although the training and test data for this experiment is entirely English, the syntax and form are atypical for natural language, and feature unusual sentence forms such as unpunctuated lists of skill terms in lieu of descriptive sentences. Multilingual BERT, having been trained on a large number of different languages with different sentence forms, may perform well on job description data and was therefore chosen for inclusion in this experiment.

DISTILBERT

DistilBERT (Sanh et al., 2020) uses *knowledge distillation* during model pre-training to reduce the size of BERT by 40% (66M vs 110M) while retaining 97% of its language understanding capabilities and being 60% faster. Its inclusion in this experiment was motivated largely by the reduced size; smaller models are preferable when performance is comparable.

ALBERT

ALBERT (Lan et al., 2020) is similar to BERT in that it uses a transformer encoder, but with three main distinctions that allow for greater parameter efficiency: the factorisation of embedding parameters, cross-layer parameter sharing, and inter-sentence coherence loss. ALBERT was shown by Lan et al. (2020) to perform equally well as BERT-base on a

variety of NLP tasks at a considerably lower parameter count (12M vs 110M). Similarly to the DistilBERT model, the inclusion of ALBERT in this experiment was largely due to the reduced size.

4.9 ENTITY RECOGNITION EVALUATION

Models were evaluated using precision, recall, and F_1 score (§2.5.4) on all entity classes with the exception of the *None* (*O*) label, consistent with standard ER tasks. Unless otherwise stated, F_1 averages are calculated using the *micro* method, which aggregates the contributions of all classes and is preferable in cases where there is class imbalance.

4.9.1 Baseline CRF Model

Label	P	R	F_1	Support
B-Skill	0.69	0.37	0.48	676
I-Skill	0.53	0.71	0.61	1429
B-Qualification	0.72	0.50	0.59	26
I-Qualification	0.39	0.23	0.29	40
B-Occupation	0.90	0.65	0.75	137
I-Occupation	0.93	0.71	0.81	164
B-Experience	0.86	0.67	0.75	9
I-Experience	0.42	0.76	0.54	17
B-Domain	0.53	0.40	0.46	60
I-Domain	0.34	0.28	0.31	39
micro avg	0.58	0.60	0.59	2597
macro avg	0.63	0.53	0.56	2597
weighted avg	0.61	0.60	0.58	2597

Table 4.16: Results for CRF model (trained on preprocessed data). Precision, Recall, and F_1 -Score are presented.

4.9.1.1 Error Analysis

We observe instances of errors in classification from the baseline CRF model and identify two main sources of error: specific versus general application of the skill classification, and the misclassification of multi-entity spans.

SPECIFIC VS. GENERAL APPLICATIONS OF SKILLS

Our annotation schema states that, when a Skill is applied to a particular task, the details of the task should only be contained in the skill-term if it is a specific application (for example, *creating technical documentation*) and not a general application (for example, *cleaning kitchens*, where only *cleaning* should be classified as a Skill). The CRF model is largely unable to distinguish between specific and general applications, and tends to include the application in either case. Examples of this are shown below, with the general application of the skill in parentheses, where the model incorrectly treats all tokens in each example as part of a classified span:

- *taking responsibility for the kitchen (in the absence of the senior chefs)*
- *training and developing new members (of the brigade)*
- *leading continuous improvement in business operations (with attention to our warehouse team and suppliers)*

MULTI-ENTITY SPAN CLASSIFICATIONS

As part of data preprocessing, large annotated spans that contain multiple discrete entities were split by punctuation (see §4.8.1). However, the CRF model often fails to split entities appropriately, and includes multiple entities of the same entity type within one span. This is true in particular of the Skills class, and contributes to the poor recall of the *B-Skill* label (see Table 4.16). Examples of this are shown below, where the CRF model has identified the entirety of each example as one span, but the correct divisions are notated by parentheses:

- *(communication) and (influencing) skills, ability to (embrace and apply leading practice tools and techniques), proven (customer service) orientation and (collaborative) approach*

- (*assessing patient cases*) and (*devising and advising care plans*)
- (*respond to internal and external stakeholder queries*) in a timely manner and (*proactively seek to resolve stakeholder issues*)

4.9.1.2 Implications and Solutions

These two sources of error appear to be failures of the CRF model caused by an inability to correctly *terminate* an identified span. If the entities were used as features for a job recommendation system, these limitations would have the effect of reducing the number of features, which might present an issue for some recommendation algorithms (for example, a bipartite graph matching approach).

A potential solution to these issues would be to use contextualised word embeddings (Turney and Pantel, 2010), which assign each token a single vector based on its context and, to some extent, capture the semantics of the word. An ER model that takes the semantics of words into account may be better able to distinguish between specific and general applications of skills, and may be better suited to identifying sensible termination points for spans to prevent multi-entity span classifications.

4.9.2 Competitive ER Models

Model results are summarised in Table 4.17, and full score breakdowns are included in Appendix A.2.

TRANSFORMER ARCHITECTURES

All BERT variations were shown to outperform the CRF baseline (.73; .72; .65 vs .59 F_1), which reflects common patterns in contemporary NLP research. The *cased* variant of BERT-base outperformed both the *uncased* and *multilingual* variants (.73 vs .72; .65), indicating that case information is of particular importance in the ER task in that it contains information that is evidently useful for distinguishing between entity classifications. For example, in the phrase *looking for an experienced General Nurse*, the casing of the term *General* is an indicator that this term is part of the *Occupation* span *General Nurse*. Additionally, pretraining BERT on multilingual data does not appear to confer any

Model	Precision	Recall	F_1
CRF (baseline)	0.58	0.60	0.59
BERT base cased	0.70	0.75	0.73
BERT base multilingual cased	0.72	0.72	0.72
BERT base uncased	0.63	0.67	0.65
CNN w/ Crowd Layer	0.71	0.55	0.62
DistilBERT base uncased	0.55	0.55	0.55
BiLSTM CRF	0.53	0.45	0.49
ALBERT	0.48	0.50	0.49

Table 4.17: Results for the ER task. Precision, Recall, and F_1 -Score are presented.

particular advantage, which is perhaps unsurprising given that the entirety of the corpus is in English.

Results for the two low-parameter transformer architectures, DistilBERT and ALBERT, are considerably poorer, and fail to meet benchmark-level performance (.55; .49 vs .59). It is important to note that the reason for this cannot be reliably attributed to the lack of data available for fine-tuning; both of these models were shown to approximate BERT-base performance after being fine-tuned on comparatively smaller corpora on a variety of tasks, including the Corpus of Linguistic Acceptability (CoLA) which contains a training set of 8.5k items (Warstadt, Singh, and Samuel R Bowman, 2018), the Microsoft Research Paraphrase Corpus (MRPC) which contains a training set of 3.7k items (Mohamed, Eldesoky, and Ali, 2015), the Recognising Textual Entailment corpora (RTE) which contains a cumulative training set of 2.5k items (A. Wang et al., 2018), and the Winograd Schema Challenge corpus (WNLI) which contains a training set of just 643 items (Levesque, Davis, and Morgenstern, 2012). For comparison, the current corpus features a training set of 10k items.

It is possible that the relatively poor performance of the low-parameter transformer architectures is in part due to the choice of task; although these architectures have been shown to perform comparably with BERT-base on a wide variety of tasks including sentiment analysis, paraphrasing, sentence similarity, and NLI (Sanh et al. (2020), Lan et al. (2020)), the current task of ER requires token classification as opposed to single-sentence classification or multi-sentence inference, and may be unsuitable for these

low-parameter transformer architectures. It appears that the knowledge distillation process of DistilBERT and the methods of increasing parameter efficiency of ALBERT relative to BERT-base do so at the cost of performance on token-level tasks such as ER.

CNN AND THE CROWD LAYER

The CNN model with the inclusion of the Crowd Layer exceeded baseline performance by a small margin (.62 vs .59 F_1). Interestingly, there was an unusual discrepancy between the precision and recall for the CNN model, with a relatively high precision (.71) and a relatively low recall (.55), indicating that the model was generally poor at detecting entity classes, but entities that were detected were generally classified correctly. This pattern is reflected in the original implementation of the CNN with Crowd Layer model proposed by Rodrigues and Francisco Pereira (2017), which similarly displayed a relatively high precision (.71) and a relatively low recall (.47) when applied to the CoNLL NER dataset.

One of the main advantages of the inclusion of the crowd layer to the CNN model is that it enables the model to learn class labels directly from noisy data without requiring label aggregation methods. However, the current dataset features two annotations per token, which is considerably fewer than the number of annotations per token in the CoNLL data, which contained between 1 and 8 annotations per token (median 5.0). Hence, although there may be several datasets for which the inclusion of a crowd layer would offer a significantly stronger performance, the current dataset does not benefit in the same way, and offers comparable performance to the CRF baseline.

BiLSTM-CRF

Although the BiLSTM-CRF is a direct augmentation of the CRF baseline model, its results are relatively poorer (.49 vs .59). There are a number of reasons why this may be the case. Firstly, the amount of training data is relatively small (10,000 sentences; between 248 and 13,663 instances of each class), which may not be enough to allow the BiLSTM layer to learn class-indicative features of language at the same level as the CRF model which captures contextual information from pre-defined features (§4.8.2). Although visualisation of development accuracies by training epoch show that the BiLSTM-CRF models were not overfitting, it is possible that the hyperparameters chosen for this

experiment were sub-optimal; a range of learning rates, embedding layer dimensions, and hidden layer dimensions were chosen for hyperparameter tuning but the width of testing bands was limited by time and available resources, since sequential models such as LSTM cannot be parallelised beyond batches and are consequently slow to train (Vaswani et al., 2017).

4.10 APPLICATION OF ENTITY RECOGNITION MODELS

Although ER in job descriptions and user profiles is certainly useful in its own right for tasks such as the identification of global skill requirements, skill shortages, and predicting future trends, the purpose of ER described in this thesis is primarily to facilitate the automatic matching of user profiles and job descriptions using NLP. In this chapter, we evaluate the performance of various ER models by comparing the set of token labels that the models have predicted with the set of token labels that are contained in the test set, which were human-labelled according to the annotation schema. During training, models learnt patterns in the input sequences that aided the identification and classification of token spans, and this aspect of the model is evaluated and used to rank model performance. Although the ability to extract term sequences according to our schema is likely to be useful when evaluating the fit between a given user profile and job description, it is possible that some pertinent entities are missed as a result of an inexhaustive or incomplete schema. Furthermore, the models proposed in this chapter that have been developed to extract the listed entities may be inadequate, by design or by insufficient training, for the purpose of feature extraction in a job recommendation system.

Typically, feature extraction processes accept raw data as input and return a structured set of features which are used for model training and inference. In doing so, the information contained in the input data that was not selected as part of a feature is lost. Ideally, this is either noise or features that are unsuitable for model inference due to irrelevance or bias. If feature extraction performs poorly, however, this process may inadvertently exclude useful information which will hinder the ability of the model to learn and result in a less useful model.

The alternative to using feature extraction in a TML model for job recommendation is to use a DL approach, which requires no explicit feature extraction process, and the

model learns which features are salient during the training process, and selects or attends to these accordingly. The main disadvantage of this approach is that, especially in the case of job descriptions, the input sequences are long (approximately 300 tokens), and many DL methods are poorly equipped to handle input sequences of this size, and attention computation and memory requirements scale quadratically with respect to the length of the input sequence (Fournier, Caron, and Aloise, 2023). For the application of these models, a method is required to address this, whether through feature extraction, input sequence truncation, or through the use of a model specifically designed to handle large input sequences. Experimentation and discussion on this particular issue is detailed in §5.5.4.

4.11 ETHICAL CONSIDERATIONS

The main ethical consideration for this research is the use of crowdsourcing data. Sabou et al. (2014) raise three issues regarding the use of crowdsourcing in research: how to acknowledge contributions; how to ensure contributor privacy and well-being; and how to deal with consent and licensing issues.

Since data was crowdsourced through the AMT platform, Workers were anonymised through the use of a unique Worker ID, and their details were restricted with the exception of general statistics regarding their past performance on the platform, and the general location (for example, *EU West*). Worker IDs were removed from the published data to ensure that their contribution to this data does not affect their future assigned work on the AMT platform.

To ensure Worker well-being, all contributions were compensated at a rate equivalent to UK minimum wage (at time of data collection); during task development, the time it took annotators to complete each HIT was recorded and averaged to calculate an estimate for the live task.

A further ethical consideration of this research is that models trained to identify and classify salient entities in CVs and job descriptions may rely on conventional linguistic norms, which introduces the possibility of misinterpretation, bias, or inequitable exclusion of candidates from underrepresented backgrounds. For example, the articulation of skills and qualifications in candidate CVs may exhibit varying linguistic nuances that are associated with their cultural or socioeconomic background, and models

trained to detect these terms may be less proficient at recognising diverse expressions, consequently placing minority candidates at a disadvantage if those terms are then used as input for a job recommendation system. Our approach, which involves training ER models to detect skill terms expressed in a variety of ways in natural language, addresses this issue to a greater extent than methods that rely on gazetteers or word-banks of known terms, but is still limited by the prevalence of diverse expressions in the available training data.

4.12 PUBLICATION OF MATERIALS

A public GitHub repository¹³ was created which contains the following:

- The annotated dataset described in this chapter, following 2003 CoNLL NER conventions
- Individual Worker responses, Worker ID, and associated accuracies on the qualification task
- The schema for the annotation task, with annotation guidelines, worked examples, and *Frequently Asked Questions* section
- The Python script for data preprocessing (described in §4.8.1)
- The Python script for aggregating Worker responses into a single label with user-specified aggregation methods
- The Python script for training and evaluating the CRF model on the provided data
- The Python script for loading a trained CRF model to extract entities from a user prompt

These materials are published under a *no rights reserved* Creative Commons BY license¹⁴, allowing for commercial and academic use with attribution.

¹³ <https://github.com/acp19tag/skill-extraction-dataset>

¹⁴ <https://creativecommons.org/licenses/by/4.0/>

4.13 CONCLUSION

In this chapter, we have presented a novel corpus for ER in the recruitment domain, annotated with five entity types: *Skill*; *Qualification*; *Occupation*; *Experience*; and *Domain*. These types are not available in standard NER corpora, but are the most relevant to this domain for tasks such as job recommendation. This data is an ideal training set for the ER task and is a suitable size for fine-tuning a pre-trained model. This corpus is our first original contribution to the field of NLP in online recruitment.

Additionally, we have presented an annotation schema to facilitate the collection of additional data, and a baseline CRF model for ER, and have suggested methods for schema development, task construction, and corpus creation. All resources associated with this paper are made publicly available¹⁵ under a Creative Commons BY license. Included in these resources is a Datasheet (Gebru et al., 2018) that fully describes the data and the method of its collection.

Our published dataset has been used in contemporary NLP research to develop skill extraction systems (M. Zhang, Goot, and Plank, 2023), and has been cited several times in the field of skill extraction and job recommendation (M. Zhang, Kristian Nørgaard Jensen, et al. (2022a); T. Yu et al. (2023); N. Li, Kang, and De Bie (2023); Naik, Patel, and Kannan (2023)).

To improve upon the baseline CRF, we also present a number of competitive ER models and discuss their performance on the supplied data. We find that transformer-based architectures offer the greatest performance on this task, at least in part due to their use of contextual word embeddings and bidirectional attention mechanisms. We show that models that are able to learn token labels directly from noisy annotations improve upon baseline performance but are unable to match the performance of transformer-based methods.

We apply the ER models as feature extraction methods for the task of job recommendation, discussed in Chapter 5.

¹⁵ <https://github.com/acp19tag/skill-extraction-dataset>

MATCHING CANDIDATE PROFILES AND JOB DESCRIPTIONS

5.1 CHAPTER OVERVIEW

This chapter addresses the second research question of this project:

RQ2 *How can deeper understanding of the candidate and job be used to influence a candidate profile-job description matching solution?*

Section 5.2 introduces the matching problem. Section 5.3 discusses three ways the matching problem can be framed: as a recommendation problem; a text classification problem; or an NLI problem. Section 5.4 describes the KJRC and the description and evaluation of various recommendation algorithms on the associated corpus. Section 5.5 describes the Tribepad corpus and the description of evaluation of various algorithms for the task of predicting applications made by users and the task of predicting the outcome status of applications. Section 5.6 describes the pipeline, or the end-to-end system, of job matching that combines the ER systems developed as in Chapter 4 and the findings from §5.5 to predict application success. Finally, section 5.7 summarises the research described in Chapter 5.

5.2 INTRODUCTION

Where **RQ1** involved the identification and extraction of salient entities in job descriptions and candidate profiles, **RQ2** centers on how these entities (and other information) can be used to develop a *matching system*; an automatic process that evaluates the quality of match between users and jobs. This process can be extended to provide a ranked list of n job descriptions with the highest quality match to a given candidate profile (or n candidate profiles for a given job description), and could be developed into a tool that helps candidates find suitable jobs and vice versa. The task of developing a matching solution is referred to hereafter as the *matching problem*.

Although previous work into job recommendation systems (§3.2) serves as an appropriate foundation, the research outlined in this thesis explores the matching problem by framing it in different ways according to different NLP paradigms: as a recommendation problem (§5.3.1); a text classification problem (§5.3.2); and as a NLI problem (§5.3.3).

Crucially, there are two key issues that will be addressed as part of this research: firstly, the issue of bilateral matching; it is not enough to solely consider unidirectional data when developing models for evaluating candidate-job pairs (for example, when a candidate applies to a job), and bilateral data is required (for example, when a candidate applies to a job, *and is accepted*). To clarify, a job recommender system trained on unidirectional data would treat an application made by a candidate to a particular job as an indicator of a good match for the candidate-job pair, even if that application was subsequently rejected by the recruiting agent due to poor suitability.

The second issue that must be addressed is that of model interpretability; a model that is able to match candidates to jobs with high precision is of little value if the user cannot interpret *why* a particular job was recommended to them. Therefore, for a matching solution to merit implementation in a deployed matching system, any prediction must be able to be queried to generate human-understandable reasoning explaining how and why the prediction was made.

5.3 FRAMING THE MATCHING PROBLEM

The ultimate goal of research into the matching problem is to develop algorithms that are able to alleviate the human processes of jobseekers selecting jobs to apply to and recruiting agents selecting applicants to invite to interview, which can be achieved by ranking the universe of potential items in descending order of suitability. However, the matching problem itself can be viewed through a number of different lenses, each of which frame the goal of the task and the methods that can be developed to address it in different ways.

Firstly, the task can be viewed through the lens of the recommendation problem (§5.3.1), where the task is to recommend n jobs to a user, which can be addressed via the development of a job recommender system and evaluated by comparing the set of recommended jobs with the set of jobs that a user applied to.

Secondly, the task can be viewed through the lens of the text classification problem (§5.3.2), where the task is to predict the outcome status of a given application by treating the concatenated user and job features as the document for analysis, which can be evaluated by comparing the predicted outcome statuses with the observed outcome statuses.

Thirdly, the task can be viewed through the lens of the NLI problem (§5.3.3), where the task is to predict if the attributes of the user *entail* (agree with) the attributes of the job, or otherwise if the attributes of the user *contradict* (do not agree with) the attributes of the job, which can be evaluated by comparing the predicted statuses with those observed.

Across the three paradigms, let \mathcal{U} denote the set of users, and \mathcal{J} the set of jobs. The dataset \mathcal{D} is a set of applications by users to jobs, i.e. $\mathcal{D} = \{\mathbf{a}_{uj} : u \in \mathcal{U}, j \in \mathcal{J}\}$, where \mathbf{a} is either 1 denoting an application was made¹, or a vector of categorical variables $\mathbf{a} = \{a_1, a_2, \dots, a_n\}$ denoting the status transitions of application \mathbf{a} (for example, $\{In\ Review \rightarrow Shortlisted \rightarrow Interview \rightarrow Offered \rightarrow Hired\}$).

We divide \mathcal{D} into a training set \mathcal{R} , validation set \mathcal{V} , and test set \mathcal{T} .

5.3.1 Recommendation Problem

By framing the matching problem as a recommendation problem, the task becomes:

Given a user and their associated details, learn a function that is able to select the n most appropriate jobs for the user.

More formally, the task is to learn a function $\hat{r}(u, j)$, which predicts the probability that user u will interact with job j . For a given user u_x , find $\operatorname{argmax}_{i=1}^n \hat{r}(u_x, j_i)$.

Although, theoretically, the positions of users and jobs as *users* and *items* in the recommender system paradigm are reversible, the research outlined in this thesis focuses on the recommendation of jobs to users, and not the recommendation of users to jobs.

¹ Used in corpora where the outcome of the application is unknown. In this case, training data may be augmented with randomly sampled negative applications, i.e. where an applicant did not apply to a job ($\mathbf{a}_{uj} = 0, \mathbf{a}_{uj} \notin \mathcal{D}$).

Framing the matching problem as a recommendation problem necessitates a recommendation system for its solution. An overview of recommender systems can be found in §2.6 and §3.2.

5.3.2 Text Classification Problem

By framing the matching problem as a text classification problem, the task becomes:

Given a job application made by a user to a job, learn a function that is able to predict the label associated with the application.

The *label* referred to may be any discrete variable associated with the application. For example, a label set {Applied; Did Not Apply} could be used to develop a model to predict whether a user is likely to apply to a particular job or not, or a label of {Hired; Rejected} could be used to develop a model to predict whether a user that made an application is likely to be successful or not, or a label of {Interviewed; Not Interviewed} could be used to develop a model to predict whether a user that made an application is likely to be given an interview for the job or not.

More formally, the goal of the matching problem framed as a text classification task is to learn a function f which can be represented as a mapping from the space of applications to the space of outcome labels: $f : \mathcal{A} \rightarrow \mathcal{Y}$. In this paradigm, training dataset \mathcal{R} is represented as a set of pairs $(\mathcal{A}_i, \mathcal{Y}_i)$, where \mathcal{A}_i is the application which may contain features pertaining to the user and job, and \mathcal{Y}_i the outcome label associated with that application.

5.3.3 Natural Language Inference Problem

While the text classification problem treats the application made by a candidate to a job as a single entity to be classified (as a concatenation of the relevant user and job features), the NLI problem treats the user and job features as two distinct inputs which need to be aligned and analysed as a pair in order to make a label prediction.

Typically, NLI problems in the field of NLP involve logical reasoning. Consider the following example from the SNLI² task:

² <https://nlp.stanford.edu/projects/snli/>

- **Premise:** A football game with multiple males playing.
- **Hypothesis:** Some men are playing a sport.
- **Classifications:** {**Entailment**; Contradiction; Neutral}

In this example, the model would receive the premise and hypothesis as two separate inputs, and would need to reason whether the latter was a logical conclusion that could be drawn from the former (indicating *entailment*), a direct *contradiction*, or neither (*neutral*).

By framing the matching problem as a **NLI** problem, the *premise* is composed of the job features represented as text, the *hypothesis* is composed of the user features represented as text, and the *classifications* are the labels associated with the application (for example, {Applied; Did Not Apply}; {Hired; Rejected}).

More formally, the goal of the matching problem framed as an **NLI** task is to learn a function f which can be represented as a mapping from the space of user profiles and job descriptions to the space of outcome labels: $f : (\mathcal{U} \times \mathcal{J}) \rightarrow \mathcal{Y}$. In this paradigm, training dataset \mathcal{R} is represented as a set of triples $(\mathcal{U}_i, \mathcal{J}_i, \mathcal{Y}_i)$ where \mathcal{U}_i is the user profile, \mathcal{J}_i is the job description, and \mathcal{Y}_i is the outcome label associated with that application.

The motivation for framing the matching problem as an **NLI** task is that human recruiting agents take on a conceptually similar task during the initial stages of the recruitment process, specifically: *does the information contained in this candidate's profile justify offering the candidate the role, an interview for the role, or neither?* Therefore, the reasoning of whether the candidate profile *entails* the job description is suitable for an automatic matching process that frames the problem as an **NLI** task.

It is important to note that the candidate's performance in an interview, which can certainly be inferred from the application status transitions after the *Interviewed* status, is not necessarily evident in the user features in the input data. This is discussed in more detail in §5.5.3.1.

One particular way in which this application of the **NLI** paradigm deviates from typical applications is the relatively large input length. Typically, **NLI** datasets feature premises and hypotheses that consist of only a few tokens. For example, the average length of premises in the **SNLI** dataset is 16.9 tokens, and hypotheses are 9.5 tokens. Text features associated with job descriptions are typically considerably longer, and in many

cases longer than typical maximum sequence lengths for Language Models (LMs). This presents an additional issue precluding straightforward application of NLI methods, since these established methods truncate input sequences to their maximum, and this approach may not be appropriate for the current application. Primarily, this is because user profiles and job descriptions are not typically written with a token limit consideration, and content appropriate for evaluation when determining the suitability of a match may not be distributed in a way conducive to truncation. For example, if jobs list the skills desired of applicants towards the end of the content contained in the job description, truncation would risk eliminating this important content. In order to address this, an additional consideration is required: rather than using raw text features as input, a preceding operation may be implemented which identifies the most salient components and extracts these to use as input. A method that could be applied here is ER, as discussed in Chapter 4, or alternative truncation methods (§5.5.4).

5.4 KAGGLE JOB RECOMMENDATION CHALLENGE

The Kaggle Job Recommendation Challenge (KJRC) and associated corpus was chosen to address the matching problem as framed as a *recommendation problem*. This was a competitive challenge proposed by US employment website CareerBuilder³ and hosted on data science competition website Kaggle⁴. For this challenge, candidates were invited to develop an algorithm that would be able to *predict the jobs that a user applied to* given user features, job features, and historic application data. The KJRC, and the systems designed to address it, are described in more detail in §3.2.2. Although the competition of the KJRC concluded in October 2012, challenge data and associated details remain hosted on the site and can be accessed with a free user account.

The KJRC was chosen to use in this research because it contains real-life observations of applications made by users to jobs, it is accessible to the academic community, and it is of sufficient size to train and evaluate TML models. The 2016 RecSys challenge data⁵, although similarly structured, is no longer available to the academic community, and could not be used as part of this research.

³ <https://careerbuilder.com/>

⁴ <https://kaggle.com/competitions/job-recommendation/>

⁵ <http://2016.recsyschallenge.com/>

The data associated with the challenge was delivered in four distinct files: information regarding job applications on the CareerBuilder platform; information regarding individual users and their demographics; information regarding employment history pertaining to users; and information pertaining to jobs advertised on the platform.

Although the corpus contains a large number of users (390k) and jobs (1.1M), it is important to note that not all users are eligible for every job. For example, a job with a listed closing date that precedes the date the user created their account is not a valid recommendation for the user since it was not possible for the user to make an application for the job. The [KJRC](#) addresses this by assigning all users and jobs to one of 7 ‘time windows’ (referred to as *windows*) according to the time period in which the majority of its applications were made or received. Model performance, measured using [MAP@150](#), was only calculated using items of the matching window. To clarify, although training data included applications made by users to jobs with different windows, the test data only included applications with matching windows between users and the jobs they applied to. This particular feature of the [KJRC](#) is worth noting due to its effect on the difficulty of the task; when making predictions about which jobs a user applied to, the pool of jobs could theoretically be reduced to approximately one-seventh of the available universe, thereby reducing the difficulty of the task relative to a task more closely aligned with a live implementation, where reducing the available universe of jobs for recommendation in such a way would be neither feasible nor appropriate. However, by keeping the conditions for the current research project consistent with those outlined in the [KJRC](#), the supplied baseline models can be readily applied and evaluated.

Unfortunately, the leaderboard of historic contributions to the [KJRC](#) uses a scoring metric that was not made publicly available after the completion of the challenge, so the given baseline methods are the only available method of reliably comparing approaches.

5.4.1 *Corpus Analysis*

Table [5.1](#) contains a summary of the supplied data for the [KJRC](#).

Table Name	Description	Number of Rows
apps	Applications made by users to jobs	1,603,111
users	User demographics	389,708
user_history	User employment histories	1,753,901
jobs	Job details	1,091,923

Table 5.1: A summary of the Job Recommendation Challenge corpus.

APPLICATIONS

There are a number of key aspects regarding the supplied application data that influence the way the prediction task should be approached. Firstly, each record in the application set contains a user id, a job id, and the assigned window id; there is no data available to indicate the strength of the application in the form of a status denoting the outcome of the application, nor a rating of any kind indicating the candidate's evaluation of the job as a good fit for them. This data, which only denotes that an application was made by the user to the job, is referred to in related literature as *implicit* data, and unless negative sampling is applied, only modelling techniques that are able to process implicit data effectively will be of use. For example, standard user-item matrix solutions (such as SVD) would not be feasible for generating predictions.

Secondly, approximately half of the users in the application test set do not feature in the application training set. This is a notable characteristic of the data since the KJRC is constructed, in terms of the task design and model evaluation procedure, to emulate a recommendation system challenge (explained in more detail in §2.6). However, common approaches to building recommender systems often utilise collaborative filtering (§2.6.2; §3.2.4) which tend to suffer from the *cold-start* problem, in that new users with no item interaction history will generally receive recommendations of poorer quality until patterns of interaction have been established, and subsequent recommendations are of higher quality. Since the current corpus features a relatively large quantity of users with no interaction history, it is likely that pure collaborative filtering methods will perform poorly, since they will only generate effective recommendations for approximately 50% of users.

Major	Frequency
Business Administration	12,414
Accounting	7,576
Business Management	5,971
General Studies	5,201
Psychology	4,784

Table 5.2: Frequency counts for the top 5 most common majors in the Job Recommendation Challenge corpus.

Thirdly, there is very little overlap between users and jobs in terms of applications; individual users typically only apply to a few jobs ($\bar{x} = 4.99$), and individual jobs typically receive only a few applications from users ($\bar{x} = 4.38$). The sparsity of an interaction matrix, as defined by Sarwar et al. (2000), is shown in Equation 5.1, where high sparsity denotes a large proportion of elements with no user-item interaction.

$$\text{sparsity} = 1 - \frac{\text{number of non-zero elements}}{\text{total number of elements}} \quad (5.1)$$

The sparsity of the current dataset is extremely high (99.999%; 1,603,111 non-zero elements in a $366,000 \times 321,235$ matrix), which limits the effectiveness of methods that infer similarities between users and jobs by evaluating the extent to which applications overlap.

These challenges will need to be addressed in order to develop a model that is able to effectively predict the jobs that a given user has applied to.

USERS & JOBS

The KJRC corpus features users from all across the USA, and lists a variety of degree types (for example, *High School*, *Bachelor's*) and majors (for example, *Anthropology*, *Agricultural Business*). The five most common majors (ignoring *Not Applicable*) are shown in Table 5.2.

Job Title	Frequency
Administrative Assistant	4,232
Customer Service Representative	4,175
Own Your Own Franchise!	3,701
Sales Representative	3,537
Mobile Tool Sales / Franchise Distributor	3,275

Table 5.3: Frequency counts for the top 5 most common job titles in the Job Recommendation Challenge corpus.

The *Job Title* field associated with the user accounts contains a wide variety of jobs across multiple domains, indicating that this data is not limited to a particular area of industry.

Although job domains are not isolated as fields in this data, analysis of the Job Title reveals that jobs span a wide variety of domains, including Hospitality, Healthcare, and Technology. The five most common jobs (using exact matches of the Job Title field) are shown in Table 5.3.

5.4.2 *Application Prediction Task*

For each user in the pre-partitioned test set, the task is to predict the top 150 jobs that the user was most likely to have applied to. Mean Average Precision (MAP) is calculated on all 150 predictions.

5.4.2.1 *Baselines*

The baseline for the KJRC task was proposed by CareerBuilder and published alongside the challenge data. It is a simplistic rule-based algorithm that fills the job prediction list of 150 using the following criteria, exhausting each in order before moving onto the next, and terminates when either the prediction list is full or all criteria have been exhausted:

1. Most popular jobs that share the user's city, state, and window
2. Most popular jobs that share the user's state and window

The *popularity* of a job is calculated by counting the number of applications to that job in the training data.

We also include a setting that ignores location and simply predicts the 150 most popular jobs in the corresponding window.

5.4.2.2 *Methods*

The following sections describe our methods for addressing the **KJRC**. We implement four algorithms for job recommendation: *k*-Nearest Neighbours; **MF**; Item Similarity, and YouTube Ranking. We select the former two algorithms for their ubiquity in recommendation tasks (§3.2) and the latter to address a characteristic feature of the **KJRC** data.

k-NEAREST NEIGHBOURS

k-Nearest Neighbours (**kNN**) is an example of a collaborative filtering method, which compares past behaviour of users to similar users to generate recommendations. A more in-depth explanation of collaborative filtering is described in §2.6.2 and 3.2.4.

kNN was implemented using the SciPy and scikit-learn libraries and utilised the compressed sparse row matrix class⁶, which complemented the sparse data in the corpus; interaction between user and jobs was binary (that is, {Applied; Did Not Apply}) and the corpus contained a large number of zero-valued elements. The *k*-nearest neighbour jobs of each target user were identified based on similarity scores. Interactions between users and jobs were aggregated to generate predictions for the target user's interaction with each job in the prediction set, and the top 150 were sorted in descending order of application likelihood.

For the approximately 50% of users in the test set with no interaction history, predictions were made using the baseline heuristic which incorporated location information.

MATRIX FACTORISATION

MF is another example of a collaborative filtering approach. It aims to factorise the user-job interaction matrix into two lower-dimensional matrices which can be used to make predictions about how a user might interact with a job they have not yet

⁶ https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html

interacted with. The two matrices represent the latent features of users and jobs which capture preferences and characteristics. MF has been shown to be a powerful method for item recommendation as it is able to incorporate sparse and large-scale user-item interaction data and capture complex relationships between users and items. A more in-depth explanation of MF is detailed in §2.6.2.1.

MF was implemented using the TuriCreate library⁷ which was adapted for implicit data. The default Stochastic Gradient Descent (SGD) method was used, where unobserved items (where users did not apply to a job) were sampled alongside observed items and treated as negative examples.

We include three separate settings of MF: one setting with no side information, a second setting with user and job location data as side information, and a third setting with all available user and job features as side information.

ITEM SIMILARITY

Item similarity is a variation of collaborative filtering (see §2.6.2, §3.2.4) that makes recommendations based on the similarity between the items that target users interact with and other items. Item similarity may be preferable to the standard user-based collaborative filtering in cases where data is sparse (Galron et al., 2018), which makes it an appropriate algorithm for consideration in the KJRC.

Item similarity was implemented using the TuriCreate library⁸. The similarity score between each user and job was calculated by computing the similarity between the job data for each column, then taking a weighted average of per-column similarities to compute the final similarity.

YOUTUBE RANKING

The YouTube Ranking system is a re-implementation of the system described by Covington, Adams, and Sargin (2016). This system was designed to generate recommendations to users on the video-sharing platform YouTube⁹, and address difficulties associated with the platform, specifically the large user base and corpus, responsiveness to new content uploaded to the site, and high sparsity. These particular challenges are

⁷ <https://github.com/apple/turicreate>

⁸ <https://github.com/apple/turicreate>

⁹ <https://youtube.com/>

Recommendation Method	Mean Average Precision @ k				
	5	10	50	100	150
Matrix Factorisation, Location Only	0.0600	0.0515	0.0518	0.0565	0.0591
k -Nearest Neighbours	0.0585	0.0504	0.0533	0.0501	0.0544
Baseline, Location	0.0530	0.0499	0.0475	0.0509	0.0530
Item Similarity	0.0201	0.0173	0.0175	0.0188	0.0199
YouTube Ranking	0.0055	0.0051	0.0053	0.0059	0.0063
Matrix Factorisation, No Features	0.0027	0.0021	0.0019	0.0021	0.0023
Baseline, No Location	0.0004	0.0003	0.0003	0.0004	0.0004
Matrix Factorisation, All Features	0.0000	0.0000	0.0000	0.0000	0.0000

Table 5.4: Job Description Challenge Model Results. Mean Average Precision at k intervals is shown for each of the implemented models. Model results are sorted by decreasing values of MAP at 150.

shared with the KJRC corpus, which makes the YouTube Ranking system suitable for consideration.

The system contains two NNs: the former designed for candidate generation, the latter designed for ranking. The candidate generation network accepts user history and coarse features as inputs, and includes a collaborative filtering module. The ranking network uses logistic regression to predict expected watch time.

Since high sparsity is a key issue the YouTube Recommender algorithm addresses, and this is also a feature of the KJRC corpus, this algorithm is a suitable candidate for job recommendation.

The LibRecommender¹⁰ implementation of the algorithm was used, with text-based user features included as side data.

5.4.2.3 Evaluation

Results for the KJRC task are shown in Table 5.4, where MAP is shown to four decimal places to more clearly distinguish between similarly-performing models.

Using the KJRC metric of MAP@150, only two settings outperformed the baseline heuristic that included location data: MF using location information as side data,

¹⁰ <https://github.com/massquantity/LibRecommender>

and *kNN*. These results seem to indicate that the models are largely unable to infer meaningful insight from text data to aid the prediction of job applications made by users, and relied almost exclusively on similarity between users inferred from previous applications and location data.

It is perhaps surprising that the YouTube Ranking algorithm (Covington, Adams, and Sargin, 2016) performed poorly relative to the baseline given that it was developed specifically to address the joint issues of a large user base and corpus and high sparsity, both of which are shared with the *KJRC*. However, our implementation of the algorithm was unable to leverage its full capacity as two particular aspects of the original architecture proposed by Covington, Adams, and Sargin (2016) were missing or limited: the *watch vector*, which was an average of the embedded video watches in the user's watch history on the YouTube platform; and the *search vector*, which was an average of the embedded search tokens in the user's search history. In the context of job recommendation, the *watch vector* would be an average of job embeddings that the user has applied to or interacted with on the online job portal, and the *search vector* would be an average of the embedded search tokens included in the user's search history on the portal. Unfortunately, the *KJRC* does not include search history in the available data, and users tended to apply to very few jobs, which resulted in a sparsely populated *watch vector*. It is also likely that the embedding process which formed the *watch vector* was unable to capture semantic attributes of the job descriptions, and certainly did not capture the location attributes of the user, which ultimately led to poor performance relative to the baseline heuristic.

It should be noted that the higher performing of the two baselines leveraged two key features of the data: the *popularity* of the job as observed by the frequency of applications to that job in the training data, and the rough *proximity* of the job to the user, which was performed using simple keyword matching: first on *city and state*, then on *state*. The underlying assumption for the proximity aspect of the baseline heuristic is that candidates are more likely to apply to jobs that are close to their current location. This is supported by the significantly stronger results when location is included in the baseline heuristic, as well as a trend of low proximity between users and jobs for observed applications.

However, proximity of the user and job as implemented in the baseline is simplistic, and could be improved to give a more accurate measure of the distance between the

user and job. For example, a user based in San Diego, CA, a job listing based in Los Angeles, approximately 2 hours away, would be predicted with equal confidence using this baseline method to a job listing based in San Francisco, approximately 8 hours away. Since the available data features location data at the *city* level at its most granular, two approaches are possible for improvement. The first is to convert each *city, state* combination into longitude and latitude coordinates¹¹ and use the Pythagorean formula to give the distance between a pair of coordinates as shown in Equation 5.2, where ϕ represents the latitude and λ the longitude. The haversine distance, shown in Equation 5.3¹², accounts for the curvature of the Earth which increases the accuracy of the metric in cases of exceptionally long distance, although the two distance metrics only deviate significantly when the two locations are far enough apart that proximity is unlikely to be a factor, and it may be that the added computational complexity of the haversine is not worthwhile in practice.

$$\text{distance} = \sqrt{(\phi_2 - \phi_1)^2 + (\lambda_2 - \lambda_1)^2} \quad (5.2)$$

$$\text{distance} = \arcsin \left(\sqrt{\sin^2 \left(\frac{\phi_2 - \phi_1}{2} \right) + \cos \phi_1 \times \cos \phi_2 \times \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right) \quad (5.3)$$

The second approach addresses the inaccuracies of using *as the crow flies* distance metrics to approximate commute times by incorporating transit limitations. This can be achieved with the Google Maps API¹³, and can be more informative when the direct distance calculated between a user and a job does not account for significant obstacles such as rivers, lack of roads, or otherwise impassable terrain. For each application, the commute time could be calculated and used as input to the model. By giving models a more accurate representation of proximity between the user and the job, model performance may improve.

We have shown that, by incorporating collaborative filtering approaches combined with location information, model results can exceed baseline performance. However,

¹¹ <https://github.com/kelvins/US-Cities-Database>

¹² The haversine formula should account for the approximate diameter of the Earth (i.e. $\times 12,742$), but since this is true of all distance calculations it can be safely ignored here.

¹³ <https://developers.google.com/maps>

there are notable limitations of the available data which should be addressed. Firstly, there are important features of users and jobs that are omitted from this data, such as *remuneration* which has been shown to be an important aspect for jobseekers (Poll, 2020). Including this data, along with the current salary of users, would be of particular benefit to models addressing the application prediction task; users may be more likely to apply to jobs that offer a greater salary than their current role, which could be incorporated into models to increase the accuracy of predictions.

A further limitation of the data used for the application prediction task is the lack of visibility regarding the *filters* that users applied when viewing the list of available jobs. CareerBuilder¹⁴, the US employment website that proposed the KJRC and provided the associated data, offers four options for job filtering on its online portal: *job type*; *date posted*; *pay*; and *distance*. In the KJRC, the universe of jobs that can be recommended to each user is restricted to the set of jobs that share the same *window id*, where this feature was drawn from the time window where the user or job were at their most active. However, if the user chose to apply one or more of the aforementioned filters, the user *would not have seen* some of the jobs in this universe, which considerably reduces the likelihood of their application. Since these filters would be activate at the point at which the user makes a job application, this information should be able to be queried and would influence the universe of jobs that could be recommended to the user.

Beyond the limitations of the KJRC corpus, there is a fundamental issue concerning the task of *application prediction* and its suitability as a framing of the matching problem, and that is in the distinction between the goal of the task and the desired application of a matching solution. The goal of the *application prediction* task is to *predict the jobs a given user will apply to*, whereas the desired application of a matching solution is to *recommend suitable jobs to a given user*. The suitability of the *application prediction* task for the matching problem relies on the assumption that users will apply to suitable jobs. This is not necessarily the case for two reasons. Firstly, on existing jobseeking portals, there are no restrictions placed on the jobs that a user can apply to or the number of applications they can make; a user can make an application to a job for which they are wholly unsuitable (for example, they meet none of the listed *essential criteria* in the job description), and may do so for a large number of jobs. In cases where this occurs, there is clear divergence between the *suitability* of a given job (which is relatively low) and

¹⁴ <https://www.careerbuilder.com>

Table Name	Description	Number of Rows
apps	Contains information on each application that was made on the system.	23,177,003
career	Describes the employment history of users. Multiple rows may pertain to a single user.	1,334,843
education	Describes the education history of users. Multiple rows may pertain to a single user.	1,212,718
job	Contains information about each job.	183,882
skills	Describes the <i>skills</i> of users (extracted by Tribepad). Multiple rows may pertain to a single user.	5,637,756
user	Contains information about each user.	4,235,600

Table 5.5: A summary of data provided by Tribepad.

the *likelihood of application* (relatively high). Secondly, the matching problem is *bilateral*, in that a good job recommendation for a user is a job which is both suitable for the user and a job for which the user is a suitable candidate. The *application prediction* task is unilateral, and provides no evidence to support that a given user is a good candidate for a given job. The implication of this is that an online portal that utilises models developed for the task of *predicting which jobs users will apply to* may not provide useful information to its users.

5.5 TRIBEPAD APPLICATION CORPUS

The *Tribepad corpus* contains applications collected on the Tribepad recruitment platform along with anonymised user information, job description details, and metadata. Crucially, the Tribepad corpus contains information on the *outcome* of each application, which is necessary for addressing the bilateral matching challenge (§5.2). A summary of the Tribepad corpus is shown in Table 5.5.

We present statistics of the corpus in §5.5.1, and describe our experiments framing the matching problem as an application prediction task in §5.5.2 and a status prediction task in §5.5.3.

Field	Description
application_id	The unique id associated with each application.
user_id	The id of the user that made the application.
job_id	The id of the job that the application was for.
status	The sequence of status codes associated with the application.
modified_time	The date of the most recent modification to the application.

Table 5.6: A summary of data fields in the apps dataset.

5.5.1 Corpus Statistics

In the following sections we describe each of the components of the Tribepad corpus shown in Table 5.5.

APPLICATION DATA

The application data file contains information pertaining to applications made on the Tribepad platform between 2015 and 2022. A summary of the fields contained in this file is shown in Table 5.6.

The *status* field contains the full sequence of status codes that show the transitions on the Tribepad platform as each application matured, for example: *Shortlisted* → *Interview* → *Offered* → *Accepted* → *Hired*. The full list of status codes along with their descriptions can be found in §A.3.1.

There are some elements that may have been present during the original application that do not feature as part of this data, for example: cover letters; supporting documentation; or answers to role-specific pre-screening questions. However, even if these elements were to exist, they would not necessarily be included as part of the user features in the current research project; the matching problem seeks to alleviate the *information overload* and *filter failure* problems that exist in the job and candidate seeking processes, and at this stage of the process the aforementioned features would not yet have been created or addressed. However, the content of such features would likely influence the decision made by the recruiting agent as to whether the candidate should be offered an interview or a job offer. This is discussed in §5.5.3.4.

USER DATA

The *user* file contains data on the location and creation time of individual users. A summary of this file can be found in the Appendix [A.3.6](#).

Although the *career*, *education*, and *skills* files also contain data pertaining to users, the intersection of users in each of these files is incomplete; of the 4,235,600 user ids:

- 3,876,386 (91.5%) are also in *apps*.
- 324,791 (7.7%) are also in *career*.
- 427,926 (10.1%) are also in *skills*.
- 427,926 (10.1%) are also in *education*.
- 174,222 (4.1%) are in all of the above.

Note that, where user data within any file has been redacted via an explicit *redacted* label in place of data, the user associated with the redacted data is omitted from all analysis. Users included in the above intersection statistics do not include those with redacted features.

Intersection analyses yield some notable findings. Most importantly, only a small minority of user profiles are associated with features in all three of career history, education history, and skills datasets. Given that users are aware that recruiting agents take these features into consideration when making their initial hiring decisions, it is unusual that so few users (4.1%) choose to populate this information in their profiles, and yet the majority of users subsequently make applications on the platform (91.5%). Initially, it was theorised that this small proportion may in part be due to the general inexperience of the user base, who may be relatively junior in their stages of career and may not have any career history to list. However, considering the small proportion of users who have listed any skill terms on their profiles (10.1%), it is more likely that the lack of populated features associated with users is due to *job search fatigue*, and the disinclination that users have for spending time populating their profiles on the online portal when they have already invested time and effort in compiling their CV. It is important to note that this data does not contain any protected attributes (for example, sex, race, physical or mental disability), and all personal data was anonymised by Tribepad prior to handover. Additionally, this data is not fully representative of CV

documents, which were not made available due to data protection restrictions. Features pertaining to the user that are contained in this and other datasets in the Tribepad corpus are those that have either been extracted from the CV documents or populated by the users themselves when signing up to the online portal.

USER CAREER DATA

The *career* file contains data on the employment history of users, where multiple rows may refer to a single user. A summary of this file can be found in the Appendix [A.3.2](#).

The text description on each row is non-uniform, indicating that it has been entered by the user without preprocessing. This presents a challenge in that salient information is embedded in free text, and methods are required for extracting the relevant information.

USER EDUCATION DATA

The *education* file contains data on the education history of users, where multiple rows may refer to a single user. A summary of this file can be found in the Appendix [A.3.3](#).

The text description on each row is non-uniform, indicating that it has been entered by the user without preprocessing.

USER SKILLS DATA

The *skills* file contains data on the *skills* of users, where multiple rows may refer to a single user. A summary of this file can be found in the Appendix [A.3.5](#).

Skill terms were extracted using Tribepad's in-house methods which are not publicly available for analysis or comment. Due to the sensitive nature of the data, the original documents from which skill terms were extracted were inaccessible.

JOB DESCRIPTION DATA

The *job* file contains data on the job postings on the platform. A summary of this file can be found in the Appendix [A.3.4](#).

5.5.2 Application Prediction Task

Given a particular user, along with a set of features pertaining to that user, the application prediction task is to predict which jobs the user applied to.

This task is identical to that proposed by the KJRC (§5.4). Although the form and structure of the corpus is largely the same, there is one crucial exception that needs to be addressed, and that is the window-based segmentation that exists in the KJRC corpus but not in the Tribepad corpus. The KJRC corpus segmented users and jobs into 7 windows based on the periods where they were most active and predicted applications were to be limited to jobs in the corresponding window. The Tribepad corpus features no such window-based segmentation. In practice, the pool of potential jobs that could be recommended to a user is restricted to those with a closing date *after* the current date, and within the scope of this task this would be the creation date of the user account. It would not be feasible in a live online job portal to assign a specific *window* to a new user in the same way as the KJRC, and the system should not prevent jobs being recommended to the user purely because the job was posted on the platform too many days prior, as long as the job is still accepting applications. Furthermore, by artificially reducing the pool of potential jobs to recommend as part of this task, the difficulty of the task is reduced, and the performance of models trained and evaluated on this data would be overstated relative to the performance of the same model implemented in production.

Table 5.7 shows the results of the baseline heuristic from the KJRC applied to the Tribepad corpus at different numbers of segmentation windows, where each segment contains approximately $\frac{1}{n}$ of the applications. As described in §5.4.2.1, the baseline heuristic recommends the 150 most popular jobs in the matching window for each user, where popularity is equivalent to the frequency of applications to that job observed in the training data. Although the inclusion of *location* was included as a separate baseline setting, the location fields for both users and jobs appeared do not appear to have been collected via a dropdown question, and are often misspelled (for example, *londoln*, presumably a misspelling of London), refer to a suburb within a city rather than the city itself (for example, *Shelfield Walsall*, where Shelfield is a suburb of Aldridge and Pelsall in the borough of Wallsall in the West Midlands, England), or contain a full or partial postcode (for example, *Dublin 8*, which is a postal district in Dublin, Ireland).

n Windows	Mean Average Precision @ k				
	5	10	50	100	150
1	0.0064	0.0032	0.0006	0.0003	0.0002
2	0.0093	0.0047	0.0009	0.0005	0.0003
3	0.0116	0.0058	0.0012	0.0006	0.0004
4	0.0135	0.0068	0.0014	0.0007	0.0005
5	0.0155	0.0078	0.0016	0.0008	0.0005
6	0.0181	0.0091	0.0018	0.0009	0.0006
7	0.0185	0.0093	0.0019	0.0009	0.0006
8	0.0212	0.0106	0.0021	0.0011	0.0007
9	0.0229	0.0115	0.0023	0.0011	0.0008

Table 5.7: Results for the baseline model (popularity only) on the Tribepad Corpus for the Application Prediction task with different numbers of segmentation windows.

For this reason, applying the baseline heuristic with location achieves equivalent results to the baseline without location, since matches between user and job location on the *city* level are considerably less common and the heuristic resorts to popularity alone when the location cannot be matched.

MAP@150 scores for the baseline heuristic at each of the values of n segmentation windows are poor compared to those achieved by the same heuristic applied to the *KJRC* data (Table 5.4; 0.0530 @ 7 windows). This is likely due in part to the high sparsity of applications in the Tribepad corpus relative to the already sparse *KJRC* corpus.

We observe an increase in MAP with increasing numbers of n segmentation windows. This increase is due to the simplification of the task as evaluated using the given evaluation metric; specifically, MAP is calculated using only positive cases (that is, only same-window applications are included in the test set, and only same-window applications may be presented as part of this heuristic), and the probability of success expected by chance increases with increased granularity of window segmentation. Furthermore, with increased segmentation the task becomes less representative of an online setting, in which assigning windows to users and jobs is unsuitable. For example, at 9 windows, only 67.9% of applications in the Tribepad corpus were featured a user and job with the same assigned window.

As discussed in §5.4.2.3, there is a fundamental issue concerning the task of *application prediction* which is the discrepancy between *predicting which jobs a user will apply to* and *recommending suitable jobs to a user*. A tool that could predict which jobs a user will apply to will have little value to a user if the jobs they apply to are unsuitable, or jobs for which they are unsuitable as a candidate. A more useful tool for a user would be one that could predict the *likelihood of application success* for a job, given the attributes of both the user and the job. The task of *status prediction* is described in §5.5.3.

5.5.3 Status Prediction Task

Given a particular application made by an individual user to a particular job, along with a set of features pertaining to the user, the job, and the application, the status prediction task is to predict the status associated with that application.

The status for prediction is one of the following binary classifications: {Hired; Rejected}, or {Interviewed; Not Interviewed}. The reasoning behind these dichotomies is explained in §5.5.3.1.

The status prediction task is only possible with data describing the historic status codes of individual applications. The current research utilises application data supplied by Tribepad, which includes a sequence of one or more *status codes* pertaining to each application, denoting the historic status transitions of that application (see A.3.1 for a description of individual status codes). Figure 5.1 shows the statuses with probabilistic transitions between states drawn from the Tribepad corpus.

By utilising the status codes for historic applications, we are able to distinguish between successful and unsuccessful applications and directly address the bilateral matching problem (Dhameliya and Desai, 2019). Our work represents a significant original contribution to the field of NLP in online recruitment; to our knowledge, no related work in this field has utilised the outcome status associated with historic job applications in this way. This is likely due to the inaccessibility of this data to the academic community due to data protection and organisation-specific restrictions. However, by virtue of our partnership with recruitment software company Tribepad, we are able to access and utilise this data as part of this research project.

In addition to standard text preprocessing (for example, html tag removal), applications were further subset to only include those made by candidates with at least one

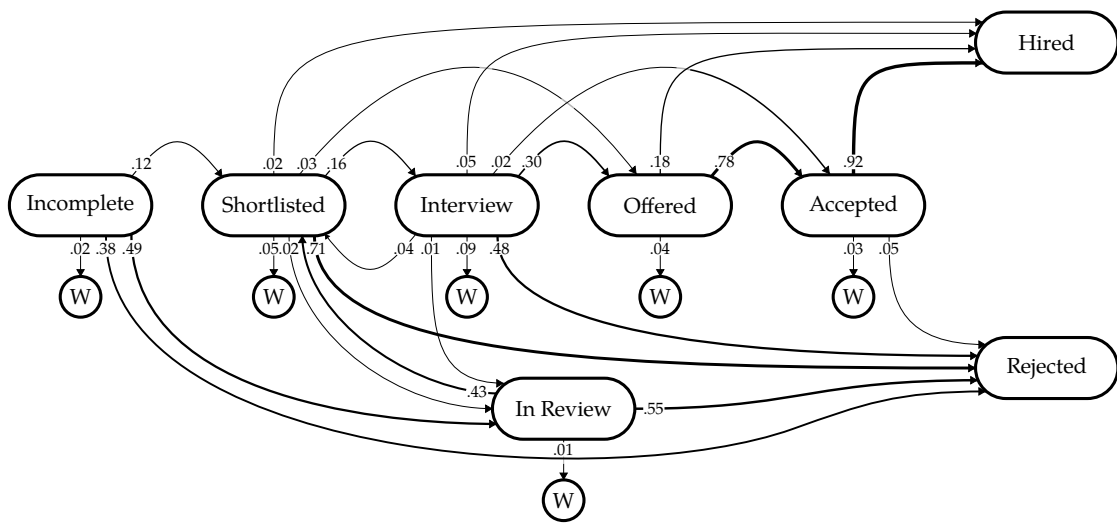


Figure 5.1: Graph to show probabilistic status transitions in the Tribepad corpus. *W* statuses represent withdrawal of the application. Transition line widths are proportional to the probabilities.

Skill listed; since the focus of this research is on the use of skill terms and how they can inform the quality of job recommendation, applications made by candidates with no listed skills were omitted.

Some of the models trained on this data utilise pre-trained word embeddings (see §2.4.2). In these cases, GloVe embeddings trained on Wikipedia data were used (6B tokens, 400K vocab, uncased, 300d). Frequency analyses revealed that > 97% of non-stop-word tokens in the input data were evident in the pre-trained word embeddings, and consequently OoV words were ignored.

5.5.3.1 Status Classification Types

Each application made by a user account to a job in the corpus is associated with a series of statuses denoting the sequence of stages in the hiring process. Although several data codes pertaining to the state of the application are evident in the data, not all of them are suitable for inclusion in the status prediction task. Applications that terminate with a *Hired* status clearly represent successful bilateral matches as this status indicates that the recruiting agents have evaluated the candidate favourably, evidenced by the candidate being offered the job, and the candidate has subsequently accepted

Subdivision	Label	Size
Train	Hired	91,895
Train	Rejected	91,895
Validation	Hired	5,430
Validation	Rejected	5,430
Test	Hired	5,455
Test	Rejected	5,455

Table 5.8: Data sampled from Tribepad Matched Data for the Status Prediction task, using the {Hired; Rejected} dichotomy.

the offer of employment. Conversely, applications that terminate with a *Rejected* status clearly represent unsuccessful bilateral matches; although the candidate has indicated their interest in the job by making an application (indicative of a unilateral match), the recruiting agent has evaluated them poorly, or at least less desirable relative to other candidates, and consequently rejected the application.

However, applications that terminate with *Withdrawn* or *Incomplete* statuses preclude inference of the suitability of the match since we cannot reasonably ascertain the reason for application withdrawal or failure to complete the application. Similarly, the *In Review* and *Shortlisted* status codes preclude inference of match suitability since they indicate the application is immature, and it is infeasible to infer a bilateral match or otherwise while the application is in progress. An application made by a candidate to a job for which they are wholly unsuitable, in that they have no relevant skills or experience and do not meet the minimum listed requirements, will eventually be assigned a *Rejected* label, but the application may also be assigned the *Incomplete* status if they failed to complete the application, the *In Review* status if the recruiting agent has not yet reviewed the application, or the *Withdrawn* status if the candidate decides to withdraw their application.

{HIRED; REJECTED}

The use of the {Hired; Rejected} dichotomy is perhaps the more comprehensible of the two dichotomies investigated as part of this research. Applications with the *Hired* status

Subdivision	Label	Size
Train	Interviewed	181,623
Train	Not Interviewed	181,623
Validation	Interviewed	10,592
Validation	Not Interviewed	10,592
Test	Interviewed	10,673
Test	Not Interviewed	10,673

Table 5.9: Data sampled from Tribepad Matched Data for the Status Prediction task, using the {Interview; No Interview} dichotomy.

are clearly indicative of a bilateral match, and applications with the *Rejected* status are clearly unsuccessful.

By using this dichotomy for the status prediction task, we aim to develop a model that is able to consider an application made by a specific candidate to a specific job, and predict whether the application is likely to be successful (*Hired*) or unsuccessful (*Rejected*).

Data was subset to include all available applications that terminate with a *Hired* status code and an equal number of applications that terminate with a *Rejected* status code were sampled. The corpus was then divided into training, validation, and testing sets using a 90% : 5% : 5% split. Although 80% : 10% : 10% is typical for ML (Gholamy, Kreinovich, and Kosheleva, 2018), the former split is suitable in cases where the dataset is small.

Frequency counts for the data are shown in Table 5.8.

{INTERVIEWED; NOT INTERVIEWED}

Beyond the self-explanatory {Hired; Rejected} dichotomy of applications, we are able to infer some level of application success through the *Interview* status code which often appears as a precursor to either the *Hired* or *Rejected* status codes. Application status transition sequences that contain an *Interview* status indicate that, following the candidate expressing an interest in the job by making an application, the recruiting agent evaluated the candidate favourably enough to extend an offer of an interview.

More specifically, from the recruiting agent's perspective, the information contained in the candidate profile was sufficient to merit an interview for that job. The ultimate post-interview decision may not be a suitable focus for this particular task because the human decision-making process that governs the subsequent part of the hiring process incorporates information that cannot be inferred from the candidate profile (for example, how well the candidate answered interview questions, or the extent to which their personality was deemed a good fit for the company).

By using the {Interviewed; Not Interviewed} dichotomy for the status prediction task, we develop a model that is able to consider an application made by a specific candidate to a specific job, and predict whether the user is likely to be offered an interview for the position (*Interviewed*) or not (*Not Interviewed*).

Data was subset to include all available applications that contained an *Interview* status code in their status transition sequences, regardless of whether the status transition sequence terminated with a *Hired* or *Rejected* status codes (or neither). An equal number of applications that met both of the following criteria were sampled: the application must not contain an *Interview* status code, and also must terminate with a *Rejected* status code (indicating that the application was rejected without interview). Applications that did not include an *Interview* status code but did include a *Hired* code were included in the former category since this indicates that the candidate was sufficiently suitable for the position that no interview was required.

The corpus was then divided into training, validation, and testing sets. Frequency counts for the data are shown in Table 5.9.

The total universe in the {Interviewed; Not Interviewed} dichotomy is larger than the {Hired; Rejected} universe simply because the *Interviewed* class is a superset of the *Hired* class and there is an abundance of negative classes in either dichotomy to sample from to match the quantity of the positive class. However, approximately 30% of the *Interviewed* class is populated by applications which terminate in a *Rejected* status (but feature an *Interviewed* status), and a further 5% of this class is populated by applications that do not contain a *Hired* nor *Rejected* status. As a result of this, the two dichotomies are not equivalent, and there is an intersection between negative classes of the {Hired; Rejected} dichotomy and positive classes of the {Interviewed; Not Interviewed} dichotomy that is not negligible, and for this reason the two are

considered distinct data selection approaches and models incorporating either are trained and evaluated separately.

5.5.3.2 *Baselines*

RANDOM RATE CLASSIFIER

The Random Rate Classifier is a baseline procedure for classification algorithms, where the predicted label for any given item is sampled from the label distribution observed from the training data. For example, if label y_1 is seen in 20% of the data, y_2 in 30%, and y_3 in 50%, then each item in the test set will have a 20% chance of having the label y_1 predicted, 30% for y_2 , and a 50% chance of having the label y_3 predicted.

Note that the features of the item for which the label is predicted have no influence on the predicted label.

Random Rate is an effective baseline for classification tasks; if a classification model scores lower than Random Rate then it is of no value to the application domain.

5.5.3.3 *Methods*

The following sections describe the methods implemented to address the status prediction task on the Tribepad corpus.

There are a number of practical constraints of training ML models, and in practice, there is a need to make simplifications so that computation is tractable within the constraints of available resources. For example, the memory and complexity of transformer models scale quadratically with increasing input sequence length, and sequences beyond a certain limit cannot be processed without modifications to the models or the data.

Experiments were conducted on a secure, remote virtual machine with the following specifications:

- Operating System: Linux v5.15
- CPU count: 2
- GPU count: 1
- GPU type: GRID A100-4-20C

- RAM: 16GB
- VRAM: 20GB

In order to allow for all chosen models to train on the available resources, inputs for the user and job features were each truncated to the first 128 tokens before being passed to the model. We adopt this method as it is the simplest, and allows us to investigate the relative performances of models on the same data. However, alternative truncation methods are possible and may be more suitable. These are discussed in more detail in §5.5.4.

COSINE SIMILARITY THRESHOLD

A simple theory of job suitability is that the more closely semantically related the user profile and job description, the higher the likelihood of application success. This can be observed through the use of word embeddings and cosine similarity thresholds, whereby user features and job features are represented in vector space and the cosine similarity between each observed user-job pair is calculated. Then, a continuous space hill climbing algorithm (Yuret and Winston, 1994) is applied to find the highest F_1 on the data in the train and validation sets, and this cosine threshold value is applied to the test set during evaluation.

LOGISTIC REGRESSION

Logistic Regression is a standard supervised learning model for binary classification tasks. Section 2.5.1.1 provides a more detailed explanation of Logistic Regression. Here, the scikit-learn implementation of logistic regression is trained on a TF-IDF feature array in one of three settings:

- *Hadamard*; Element-wise multiplication (known as the Hadamard product) of TF-IDF vectors for user sequence and job sequence
- *Euclidean*; Euclidean distance between TF-IDF vectors for user sequence and job sequence
- *Concatenated*; concatenated TF-IDF vectors for user sequence and job sequence

Logistic Regression was then performed on the [TF-IDF](#) feature array using the Broyden-Fletcher-Goldfarb-Shanno algorithm (Shanno, 1985), which has been shown to be a powerful method for solving unconstrained optimisation problems (T. Yang, P. Li, and Xiaoliang Wang, 2020).

BILSTM, BIGRU, SUMEMEDDINGS, & AVGEMBEDDINGS

The following four models were implemented using identical model architectures:

- [BiLSTM](#), a detailed description of which can be found in §2.5.1.2
- [BiGRU](#), a detailed description of which can be found in §2.5.1.2
- [SumEmbeddings](#), where the output of the previous layer is *summed*
- [AvgEmbeddings](#), where the output of the previous layer is *averaged*

These models were calibrated for the [NLI](#) framing of the matching problem and accepted the user and job text input sequences as two separate inputs. Tensorflow implementation of the models was used, and the common architecture of the models is shown in Figure 5.2, where the λ layer was replaced as above.

Pretrained [GloVe](#) word embeddings were used in the embedding layer, and hyperparameters such as [LSTM/GRU](#) unit count, learning rate, and dropout were optimised via grid search.

BERT

Bidirectional Encoder Representations from Transformers ([BERT](#)) for text classification was fine-tuned and applied for the status prediction task. A more detailed description of [BERT](#) can be found in §2.5.1.2.

We include two experimental settings of pretrained [BERT](#) models: [BERT-base](#), which features 110M parameters; and [BERT-large](#), which features 340M parameters.

In order to fit the encoded representations of input sequences in memory, inputs for the user and job features were each truncated to the first 128 tokens before being passed to the model. This truncation process, and alternatives, is discussed in more detail in §5.5.4.

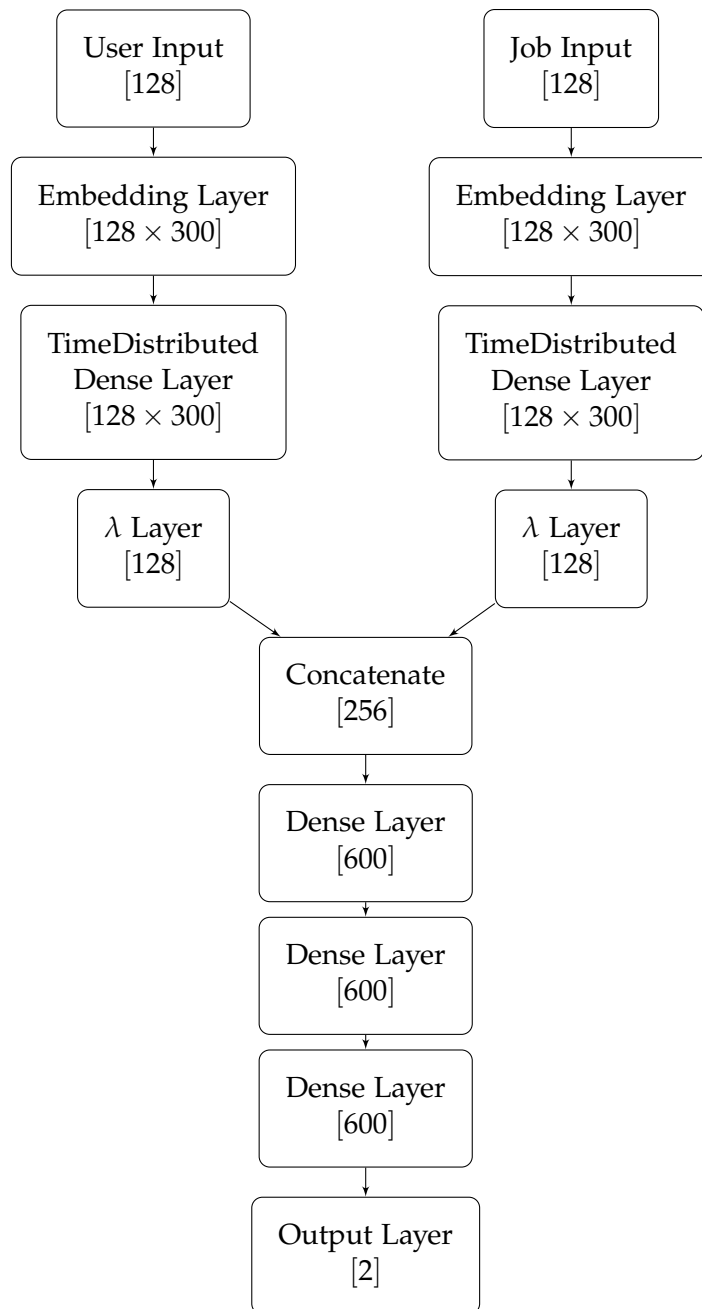


Figure 5.2: Architecture of the BiLSTM, BiGRU, AvgEmbeddings, and SumEmbeddings models for the Application Prediction task on the Tribepad corpus. The λ layer is unique to each setting. Dimensionality of layers is shown in square brackets.

RE2

RE2 (R. Yang et al., 2019), named after the components *Residual vectors*, *Embedding vectors*, and *Encoded vectors*, is a neural architecture for text matching applications and is described in more detail in §3.4.3.

The implementation of each layer is kept as simple and lightweight as possible, and the residual connections between consecutive blocks are augmented to provide richer features for the alignment process; the three parts in the input of alignment and fusion layers are: the original point-wise features (*embedding* vectors), previously aligned features processed and refined by previous blocks (*residual* vectors), and the contextual features from the encoder layer (*encoded* vectors). With this approach, the authors demonstrate improved performance on the SNLI corpus compared to a variety of NLI models, including the decomposable attention model proposed by Parikh et al. (2016). By reducing complexity and training times while achieving comparable or greater performance, this model architecture is a suitable choice for NLI tasks.

The model architecture in our experiments was constructed to match that proposed by R. Yang et al. (2019), and input sequences were each truncated to 128 tokens before being passed to the embedding layer.

DECOMPOSABLE ATTENTION

The current implementation of decomposable attention is based on the work by Parikh et al. (2016), which has shown to be particularly effective when applied to NLI tasks.

The model is comprised of three main components which are trained concurrently:

- An **Attention** component, which aligns the elements of the encoded representations of the user features and job features.
- A **Comparison** component, which compares each aligned term to produce a set of vectors for the user features and job features separately, where each vector is a non-linear combination of the term and its corresponding aligned term.
- An **Aggregation** component, which aggregates the vectors produced by the Comparison component and uses these to predict the output label.

There are three particular advantages of the decomposable attention model when applied to the status prediction task. Firstly, the input sequences of user features and

job features are *directly aligned* using a variant of neural attention, and each aligned subphrase is compared, which is conceptually appropriate for the task of comparing individual skill requirements in the job features with the skills in the skillset contained in the user features.

The second advantage is that of the reduced algorithmic complexity as a result of the *decomposition* component; unlike transformer based architectures such as BERT, the complexity of decomposable attention scales linearly with input sequence length as opposed to quadratically. Combined with the parallelisable feature of the model, decomposable attention models take a relatively short time to train until convergence.

The third advantage of the decomposable attention model is that it can be queried and interpreted, and the aligned subphrases that influence prediction can be visualised, which may be of particular use in the status application task. For example, a user who views a job on an online portal may not only benefit from a score indicating how likely they are to be hired for the role, but also the accompanying information that shows the user *which parts* of their profile were important when calculating their given score.

5.5.3.4 Evaluation

Due to data protection concerns given the personal nature of the user features, experiments were conducted on a secure, remote virtual machine with the following specifications:

- Operating System: Linux v5.15
- CPU count: 2
- GPU count: 1
- GPU type: GRID A100-4-20C
- RAM: 16GB
- VRAM: 20GB

Results for the Status Prediction task on both dichotomies of Tribepad data are shown in Table 5.10, along with the performance of the same models trained and evaluated on the SNLI corpus.

Model	Micro F_1 Score		
	{H; R}	{I; I'}	SNLI
BiLSTM	0.72	0.72	0.76
BiGRU	0.72	0.72	0.79
Decomposable Attention	0.71	0.71	0.82
SumEmbeddings	0.72	0.71	0.80
AvgEmbeddings	0.71	0.71	0.78
BERT base	0.71	0.71	0.78
BERT large	0.71	0.71	0.86
Logistic Regression, Concatenated	0.71	0.71	0.63
RE2	0.68	0.69	0.89
Logistic Regression, Hadamard	0.58	0.58	-
Logistic Regression, Euclidean	0.55	0.55	-
Cosine Similarity	0.54	0.53	-
Random Rate	0.50	0.50	0.33

Table 5.10: Results for the Status Prediction task on the Tribepad and SNLI corpora. Micro F_1 -Scores are presented for each of the models on each of the dichotomies: {Hired; Rejected} and {Interviewed; Not Interviewed}, as well as the established SNLI corpus which features 3 classes.

There are several notable aspects of the results from the status prediction task. Firstly, each of the methods improves on the Random Rate baseline, although the Cosine Similarity Threshold method shows a very minor improvement on baseline performance. The implication of this is that, although this method is theoretically leveraging the semantic content of the user profile and job description, it is too simple a method and does not take into consideration the relative importance of each term when evaluating the strength of the match and is therefore unsuitable for this task.

With the exception of the RE2 method, the balance of the models show relatively similar performance on both dichotomies of Tribepad data, ranging between 0.71 and 0.72 F_1 scores. The range of scores for the same models on the SNLI data is considerably more diverse, with scores ranging between 0.63 and 0.89. There are a number of different reasons that could be contributing to the diversity of scores between the two corpora, for example the difference in the number of labels; the Tribepad corpus features a dichotomy of classes, whereas the SNLI corpus features a trichotomy ({Entail; Contradict; Neutral}). Methods that learn a hyperplane to separate data points, such as Logistic Regression, can be more readily applied to binary classification tasks than multi-class tasks, which may be a contributing factor to the relative success of Logistic Regression on the Tribepad corpus over the comparatively weaker performance on the SNLI corpus.

A further notable aspect of the status prediction task results is the comparative performance of the concatenated setting of Logistic Regression compared to more complex attention-based models when, across contemporary NLP tasks, attention-based models typically outperform TML such as Logistic Regression. When feature engineered models outperform more complex approaches, it is typically because the differences between classes is easily identifiable from extracted features. In this case, it may be a combination of multiple reasons, including the divergence of form from the natural language data that the complex language models have been trained on; while BERT was originally pre-trained on the whole of English Wikipedia, which features syntactically sound text from which the model can learn language rules, the Tribepad corpus features non-uniformly constructed text fields that have been concatenated and therefore lack syntactic cohesion. Although BERT encodes sequential and temporal relationships between words, this model performs comparably to Logistic Regression with TF-IDF features.

An explanation for the performance of Logistic Regression relative to more complex models is that it may be more closely aligned with the process that recruiting agents are performing when selecting candidates to interview or hire. In the task of *NLI*, the semantic content of the premise and hypothesis needs to be understood in such a way that the model is able to distinguish between *entailment* and *contradiction*. When the matching problem is framed as an *NLI* task, we assume that the recruiting agent evaluates the user profile in a similar way before deciding whether to accept or reject the candidate. However, this may not be an accurate representation of the process and it is possible that the real process is much more simplistic, perhaps relying on keyword matching methods, for example, if the user profile contains the skill *Java* for a *Java Developer* role. If this were the case, the matching problem is a comparatively simpler problem than the generic *NLI* task, and in a large enough dataset common patterns of favoured terms can be identified and leveraged for the classification task, which may explain why a relatively simple model such as Logistic Regression is able to perform comparably with more complex models.

As referred to in §5.5.1, there are some elements that may have been present during the original job application that do not feature as part of this corpus, for example: cover letters; supporting documentation; or answers to role-specific pre-screening questions. Given that the status prediction task is to predict the outcome status associated with a given application, it is likely that this information would aid the model in its reasoning; a cover letter may be constructed to very specifically target the essential requirements of a job, suggesting further skills or experience that strengthen the application, or conversely a poor score on a pre-screening questionnaire may weaken an application that appears strong when considering only the content of the user profile. Furthermore, if the aim of such a model is to replicate the human decision-making process of the recruiting agent evaluating the strength of an application, it stands to reason that the model should consider the same evidence that the human agent would have access to. However, the stage at which these systems would be applied precedes the stage at which supporting documentation is constructed and pre-screening questionnaires would be answered. Given that this information does not yet exist at the point at which the user requires the available universe of job descriptions to be evaluated relative to their profile and sorted by suitability, it is not appropriate to include this data in the training and evaluation of matching solutions as we seek to apply them in this

thesis. Should this data exist, however, it may be interesting to view the extent to which supporting documentation, or lack thereof, impacts the decision for a candidate to be invited to interview, hired, or rejected.

5.5.4 Using Short-Form Models on Long-Form Input Sequences

As referred to in §4.10, the transformer-based architectures described in §5.5.3.3 share one crucial limitation; the memory and complexity of transformer models scale quadratically with increasing input sequence length. These experiments were limited by the computational resources available (§5.5.3.4) which necessitated the truncation of input sequences (i.e. job descriptions and user profiles) to 128 tokens each. User profiles were largely unaffected as 15.5% of user profiles were shorter than the 128 token limit. However, job descriptions were severely impacted by this sequence limit, and 99.8% of items exceeded the 128 token limit and were consequently truncated. Figure 5.3 visualises the distribution of token counts for user profiles and job descriptions in the Tribepad data with the sequence limit threshold.

Intuitively, a CV seems to be a longer document than a job description. However, this is not reflected in the Tribepad corpus for two reasons. Firstly, user profiles are not direct transcriptions from candidate CVs, but rather a combination of free-text fields that have been populated by the user when signing up to the Tribepad portal (a more detailed description of user data can be found in §5.5.1). It is possible that, in doing so, the word count relative to the original document is reduced. No such constraints appear to have been imposed on the recruiting agents in the construction of the job description. In the Tribepad corpus, the job description is comprised of four distinct text fields which contribute to the length of the documents: the *external job summary*; the *internal job summary*; the *ideal candidate description*; and the *description of job responsibilities*. Although the *internal job summary* was not made visible to the users when making applications, it is likely that this information was used by the recruiting agents in conjunction with the balance of data fields when evaluating applications that had been made, and is therefore included in our status prediction experiments.

Salient semantic content in natural language has been shown to be non-uniformly distributed across sentence positions (S. Yu et al., 2016). The same is likely true of job descriptions, in which salient semantic content (for example, the list of required skills

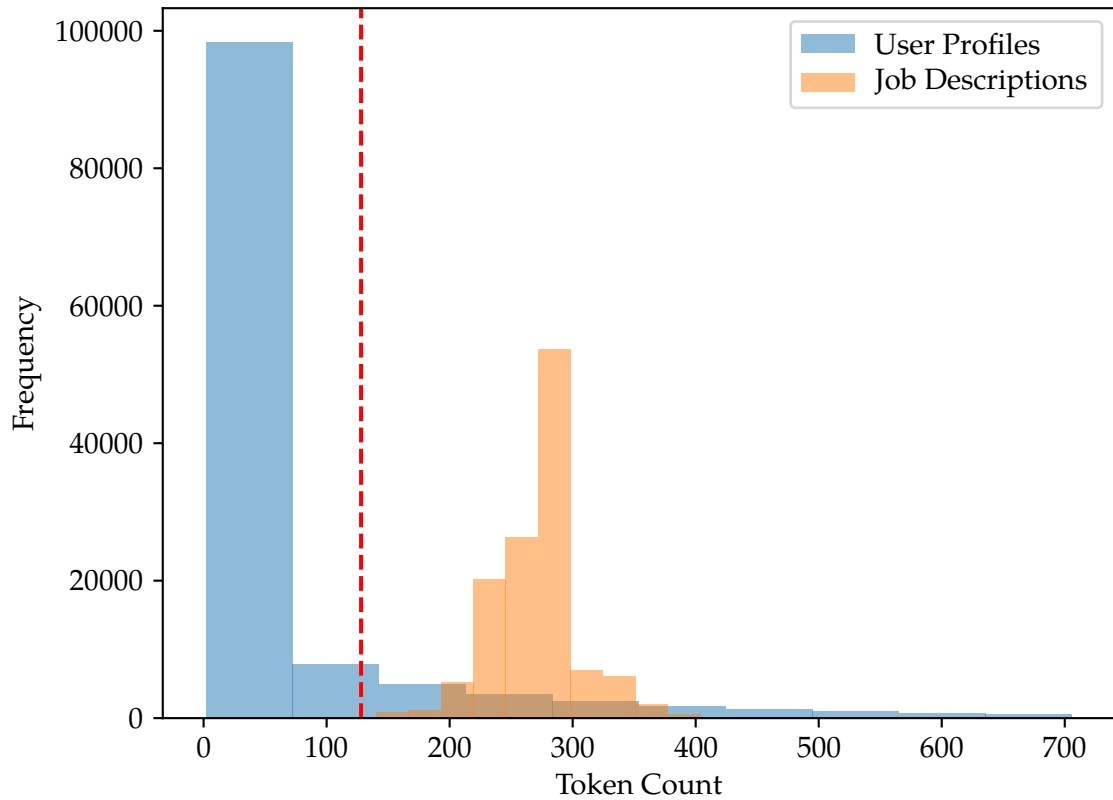


Figure 5.3: Distribution of Token Counts for User Profiles and Job Descriptions in the Tribepad data. The input sequence threshold of 128 tokens is shown in red.

or qualifications) is unlikely to be distributed evenly across the document, and it is more likely to feature spans of rich semantic content in certain parts of the document. However, since the structure of job descriptions is non-uniform, it is not clear whether these areas of semantic content are generally situated at the front or rear end of the documents, or somewhere in-between.

To investigate this, we compare four truncation methods and their subsequent effect on model performance: truncation from the front, truncation from the back, random selection, and in descending order of *IDF* and *TF-IDF* values.

FRONT

In this setting, we select the first 128 tokens in the input sequence. This is the default truncation method in many implementations of short-form models, and was used for the experiments described in §5.5.3.3. *Front* truncation is ideal if documents tend to be front-loaded with semantic content; in the context of job descriptions, if the information necessary for effective job recommendation is generally at the start of the document (for example, the list of attributes pertaining to an ideal candidate), then *front* truncation will be more effective than other truncation methods at capturing important semantic content, which will ultimately result in a better performing model.

BACK

In this setting, we select the last 128 tokens in the input sequence. This is the reverse of the *front* method, and will outperform similar truncation methods if job descriptions tend to feature important semantic content at the rear end of the document, such as the list of required skills.

RANDOM

In this setting, we randomly sample 128 tokens without replacement from the input sequence. We include this setting as a benchmark to compare to other methods of truncation; if the resultant model using *random* truncation performs comparably with all other methods of truncation, then it is unlikely that there is any advantage to manipulating the truncation process of applying short-form models to long-form data.

The process of random sampling is performed once, and models were trained on the same truncated corpus.

IDF & TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) (Sparck Jones, 1988) is a measure that evaluates the relevancy of a word relative to a document in a corpus. It is the product of the *term frequency* and the *inverse document frequency*. A more detailed explanation of TF-IDF and its components can be found in §2.4.1.

TF-IDF is a common metric in NLP for roughly evaluating the relative importance of terms in documents. In this setting, we calculate the TF-IDF metric for each word in the corpus and select the 128 tokens with the highest value for each document. To avoid data leakage we do not include the test or validation data in TF-IDF calculation for the training data.

Similarly to the *random* truncation method, the TF-IDF method loses all sequential attributes of the input text, although capturing the most ‘important’ terms in the input sequence may compensate for this. Methods that do not leverage positional information of tokens in the input sequences should not be affected.

In addition to standard TF-IDF, we include a setting using *boolean term frequency* which sets the TF to 1 if the term is present in the document. This approach tends to be useful in contexts where it is important to capture the presence or absence of terms across a collection of documents, and where the frequency of those terms within documents is of less importance. Since this setting removes the variability of the TF component, it is referred to as IDF.

5.5.4.1 Results

Results for the experiments are shown in Tables 5.11 and 5.12 using the {Hired; Rejected} and {Interview; No Interview} dichotomies respectively.

5.5.4.2 Discussion

The results show a common trend of TF-IDF truncation method leading to marginally greater model performance for the BiLSTM and Decomposable Attention models relative to other truncation methods, and less consistent results for the various truncation

{Hired; Rejected}	Truncation Method				
	Front	Back	Random	IDF	TF-IDF
Average Embeddings	0.7106	0.7159	0.7038	0.6882	0.7131
BiGRU	0.7225	0.7126	0.6920	0.6885	0.7122
BiLSTM	0.7218	0.7100	0.6929	0.7005	0.7255
Decomposable Attention	0.7120	0.7062	0.6838	0.7045	0.7389
Sum Embeddings	0.7201	0.7132	0.7017	0.6924	0.7117

Table 5.11: F_1 scores for truncation method experiments using the Tribepad corpus with {Hired; Rejected} status dichotomy. The best performing truncation method for each model is shown in bold.

{Interviewed; Not Interviewed}	Truncation Method				
	Front	Back	Random	IDF	TF-IDF
Average Embeddings	0.7082	0.6908	0.7019	0.6824	0.7067
BiGRU	0.7166	0.7045	0.6880	0.6826	0.7043
BiLSTM	0.7108	0.7060	0.6804	0.6945	0.7137
Decomposable Attention	0.7102	0.6443	0.6838	0.6985	0.7313
Sum Embeddings	0.7082	0.6943	0.7135	0.6863	0.7022

Table 5.12: F_1 scores for truncation method experiments using the Tribepad corpus with {Interview; No Interview} status dichotomy. The best performing truncation method for each model is shown in bold.

methods on the balance of models. This trend suggests that the BiLSTM and Decomposable Attention models are better able to represent job descriptions when the most relevant terms are used, which leads to greater model performance.

The relatively poor performance of models trained on data using the IDF truncation method is particularly notable. In several cases, this method is outperformed by the *random* method, which would suggest that the terms selected through IDF appear to be of *lesser* predictive importance than average. IDF is equivalent to *boolean TF-IDF*, in that the presence of a term in a document, no matter how many times it appears, yields a term frequency value of 1. This seems more conceptually appropriate in job descriptions, where the reference of a particular skill term or domain, even if it is of particular importance to the job, may only be included once. However, our experimental setting of the job description included up to four concatenated components: the *external job summary*; the *internal job summary*; the *ideal candidate description*; and the *description of job responsibilities*. If a term is of particular importance to a job, it is likely that it will appear in multiple of these components, which may explain the relative improvement of the *TF-IDF* setting over the *IDF* setting, since there is valuable information that can be inferred from the frequency of terms in the combined document.

In conclusion, when we are unable to fit all tokens in the input sequence into the model, selecting the most appropriate terms for model input leads to greater model performance. In our experiments, selecting the terms with the greatest *TF-IDF* weighting as input to the Decomposable Attention model leads to the greatest model performance. However, there are two aspects of this process that may be improved. Firstly, in our experiments, tokens were treated independently and in isolation as input to models that did not make use of the position of words in sentences, whereas in job descriptions this is not always the case, as important skill terms may be multi-word phrases (for example, *data science*; *machine learning*; *technical product management*), and treating these tokens independently neglects the important semantic content of the combined term. Models may be improved by the incorporation of methods that capture multi-word entities within the text. Secondly, letting *TF-IDF* (or any other truncation method, for that matter) effectively decide which tokens the model receives as input results in an overall loss of information. Ideally, the terms that are not selected are those that have no use as part of the decision-making process for the matching problem, but there is no guarantee that this is the case, and uncommon terms or those that appear once

within a job description but are of particular importance will be ignored. Models may be improved by a learned process that selects the most appropriate terms as input for the model, as opposed to simple truncation of input sequences.

Section 5.6 describes experiments that were constructed to address these particular issues, combining the ER process detailed in Chapter 4 for feature selection with the Decomposable Attention model developed in §5.5.3.

5.6 MATCHING PIPELINE

5.6.1 Matching Pipeline Overview

The second research question this thesis addresses is as follows:

RQ2 *How can deeper understanding of the candidate and job be used to influence a candidate profile-job description matching solution?*

To effectively address this question, we have conducted thorough research into each of the components that comprise an applicant profile-job description matching solution. In this section we bring together the individual components in a novel *pipeline* that accepts raw data as input and produces recommendations or predictions as an output. Figure 5.4 visualises the pipeline architecture.

The combination of the ER and matching algorithm components into a cohesive framework represents a novel scientific contribution of this research. While entities within user profiles and job descriptions have been utilised to evaluate the strength of an application (specifically the *skillset* as described in §3.4.2), to our knowledge there has not yet been work dedicated to integrating a trained ER system into a matching solution. Additionally, adapting these extracted entities for an NLI task to predict the outcome status associated with the application is a novel contribution, and the composition of the two components in a *matching pipeline* enhances the efficiency of job recommendation and lays the foundation for a more informed and streamlined online recruitment system.

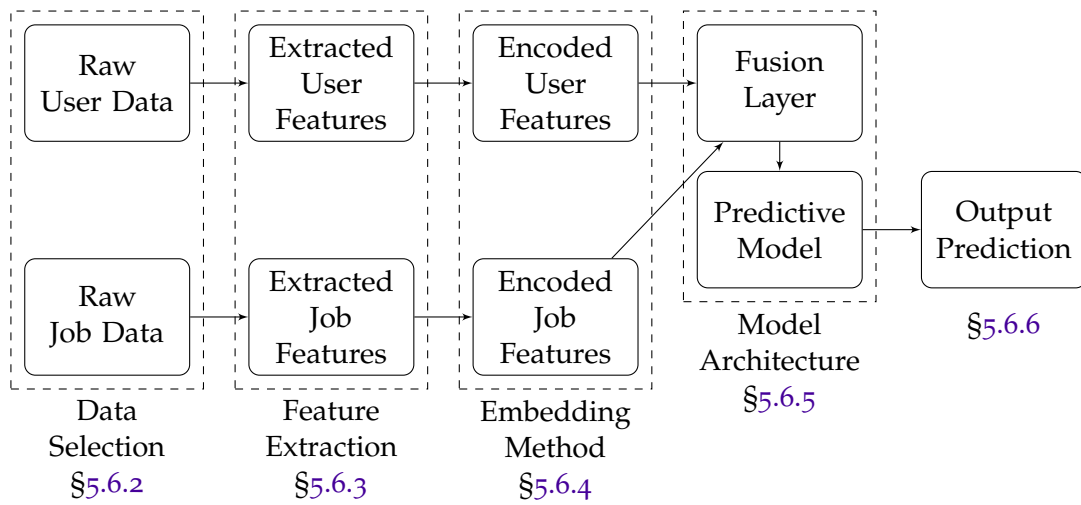


Figure 5.4: Architecture of the proposed job matching pipeline. Relevant sections are shown in parentheses.

5.6.2 Data Selection

Data Selection refers to the process by which certain fields in the available data are included or excluded from processing. Over-inclusion of available data fields will, at best, increase model training and evaluation times, increase model size, and increase complexity for little or no increase in model performance. At worst, over-inclusion may result in a predictive model that draws inference from unintended features, and consequently risks propagating systematic and institutional biases (Bogen and Rieke, 2018). Conversely, under-inclusion of available data may result in a model with little predictive power, or a system that disadvantages certain candidates or jobs.

Data eligible for selection as part of this process required legitimate reasoning supporting its potential role in a matching process. Essentially, the following question was considered for each attribute of the job and candidate profile data: *would a human agent consider this attribute when deciding if the candidate is a good fit for this job, or vice versa?* Ultimately, attributes such as the candidate’s listed skills, education and employment history were included, but those related to the candidate’s demographics, such as current location and names of educational institutions, were omitted. Although we have demonstrated that location is an important feature for consideration (see §5.4),

User Features	Job Features
Skills	External Job Summary
Education History	Internal Job Summary
Employment History	Ideal Candidate Description
	Description of Job Responsibilities

Table 5.13: Data Selection for the Job Matching Pipeline.

this is only relevant when predicting *which jobs a candidate will apply to*, since closer proximity between user and job has been shown to be associated with a higher rate of application. The focus of the current matching pipeline does not include application prediction, but rather: *given that an application has been made, predict the outcome status associated with the application*. Furthermore, the inclusion of demographic features may result in a system that leverages unintended latent features to aid predictions, reflecting patterns of biases observed from live training data. For example, candidates with listed home locations that are less socio-economically developed or feature high density of minority groups may be unfairly disadvantaged.

Selected data pertaining to jobs includes information that is available to the candidate at the point of application as well as information withheld from the candidate. Notably, both forms of the job summary are included: the *external* job summary, which typically resembles an advertisement, intended to entice candidates to apply, and the *internal* job summary, which is a more technical and specific document describing the job.

Selected features of the candidate profiles and job descriptions as part of this process are listed in Figure 5.13.

To simplify the pipeline architecture, all features pertaining to each unique user and job were concatenated into a single document before being passed to the next component of the pipeline.

In summary, the Data Selection component of the pipeline yields the following datasets:

- a dataset \mathcal{D} represented as a set of triples $(\mathcal{U}_i, \mathcal{J}_i, \mathcal{Y}_i)$, where \mathcal{U}_i is the user profile, \mathcal{J}_i is the job description, and \mathcal{Y}_i is the outcome label associated with that application (described in §5.6.6)

- a dataset containing documents of concatenated data fields associated with each user profile
- a dataset containing documents of concatenated data fields associated with each job description

5.6.3 Feature Extraction

Feature Extraction refers to the process by which salient entities are automatically extracted from data. Here, we employ the findings from Chapter 4; using the entity schema of Skills, Qualifications, Occupations, and Domains, we select our best-performing model (BERT) as the ER method for feature extraction.

Formally, this component of the pipeline associates a variable length set of salient entities with each user id and job id. For example, the user with id i would be associated with entity set $\{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_y\}$, where y is the number of entities extracted for user u_i .

The text preprocessing method performed before applying ER was identical to that used when performing ER experiments in Chapter 4 (specifically, §4.8.1); HTML tags, whitespace, and accented characters were removed, contractions were expanded, and text was converted to lower case.

The Feature Extraction component of the pipeline takes two datasets as input: the dataset containing documents of concatenated data fields associated with each user profile, and a dataset containing documents of concatenated data fields associated with each job description. The method applied to both datasets was consistent. Text fields passed through the ER model were first split into sentences, using periods, question marks, and exclamation marks as sentence delimiters. Although theoretically this is unnecessary, given that subdividing the text into smaller documents does not affect the entities contained therein, this process was performed in order to keep input data consistent with the form and structure of data observed by the ER models during training and evaluation.

Although our trained ER methods identify and extract five entity classes from text data (*Skill, Experience, Qualification, Occupation, Domain*), we exclude any *Experience* entities from passing to the next component of the pipeline. Since these are periods of time (for example, *two years experience*), they have little value in the absence of explicit information indicating which entity or entities they refer to.

In summary, the Feature Extraction component of the pipeline yields the following artefacts:

- a set of key-value pairs associating each entity id with the original entity text
- a memory-mapped file containing the set of entity ids corresponding to each user and job id
- a set of key-value pairs associating each user id and job id with the corresponding index in the memory-mapped file

This process is visualised in Figure 5.5.

5.6.4 *Embedding Method*

Embedding refers to the creation of mathematical representations of word meanings (see §2.4.2). The embedding method of the pipeline encodes the extracted features passed from the previous component of the pipeline to yield an n -dimensional vector for each entity. Entity embeddings created as part of this process will be closer in vector-space if they are similar in meaning.

This component of the pipeline is separate from the model architecture component (§5.6.5) in order to reduce the operational cost of the overall process. By encoding all extracted entities in the data in a single dedicated process, the computational cost is minimised since encoding is performed once for each unique entity extracted as part of the feature extraction component. An alternative method would be to encode entities as they are processed during model training, but this method necessitates either encoding entities as required (which will result in the same entity undergoing the encoding process multiple times) or including a logical check each time an entity requires encoding to check if it has been seen before. Performing the embedding process once, and constructing a dictionary of entities and their corresponding embeddings, reduces the total cost of computing resources. Furthermore, keeping this component modular enables us to experiment with different encoding methods which form a critical part of the pipeline.

We select four different encoding models for generating entity embeddings for this component of the pipeline, which are shown in Table 5.14. MiniLM, MPNet, and BERT

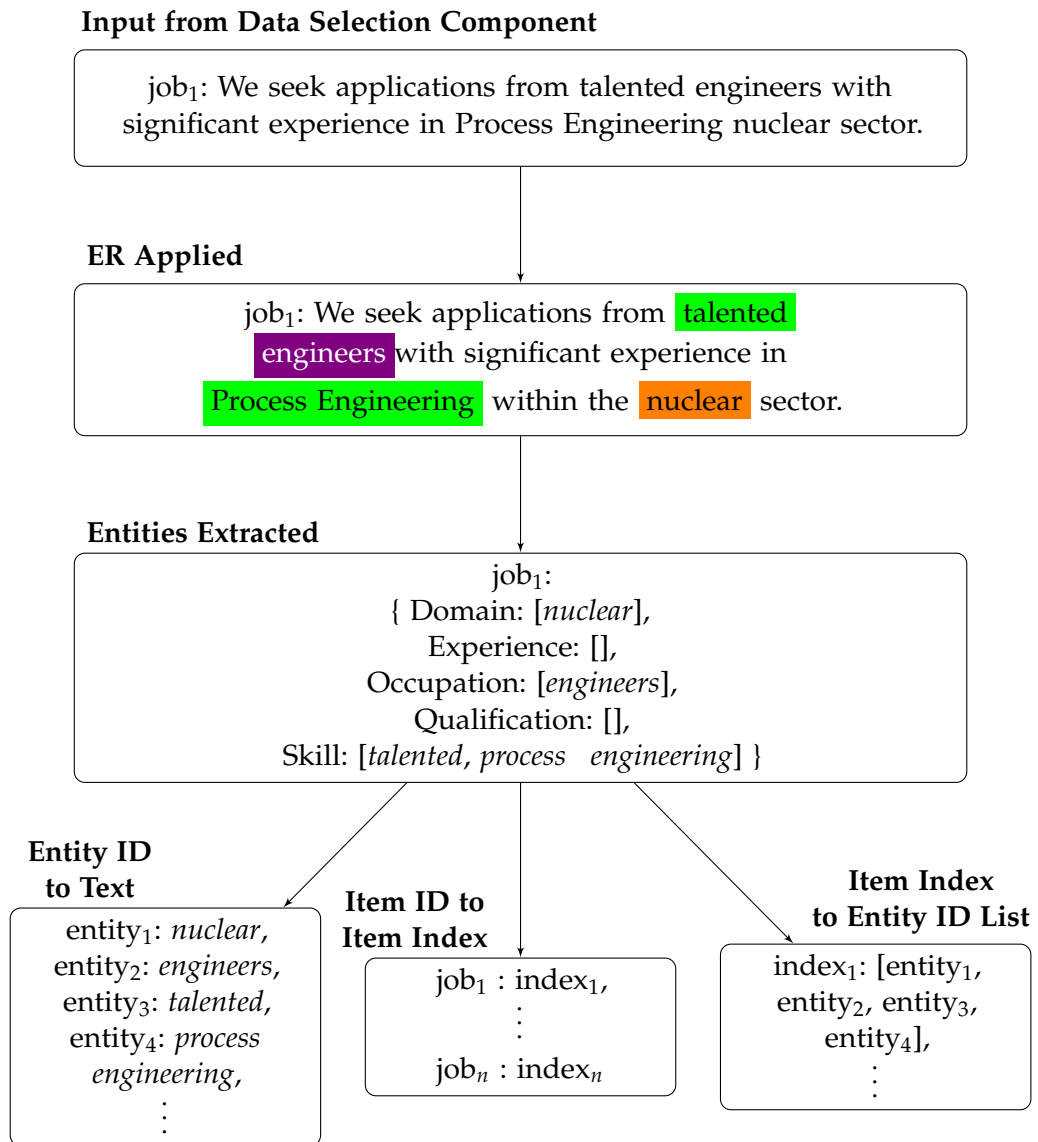


Figure 5.5: Visualisation of the Feature Extraction Pipeline Component.

Model Name	Embedding Method	Dim.	Max Seq. Length	Encoding Speed (sent/sec)
GloVe Wiki Gigaword	Token	100	-	-
MiniLM L6 V2	Span	384	256	14200
MPNet V2	Span	768	384	2800
BERT NLI Mean Tokens	Span	768	128	34000

Table 5.14: Encoder Model Attributes for the Job Matching Pipeline.

NLI Mean Tokens (Reimers and Gurevych, 2019) are *sentence embedding* methods, which convert sentences and paragraphs into n -dimensional vector space and are hosted on the HuggingFace Model Hub¹⁵. In addition, we include token-level GloVe (Pennington, Socher, and Christopher D Manning, 2014) embeddings as a setting to compare the effectiveness of sentence embedding methods versus simple token-by-token encoding. In this setting, rather than generating a vector representation for each extracted entity span, we split each entity span by token and encode each token separately.

With the exception of GloVe, all methods are pre-trained on NLI data and have been extensively evaluated for their quality to embedded sentences, queries, and paragraphs¹⁶.

5.6.5 Model Architecture

We use the Decomposable Attention model as described in §5.5.3.3 for two reasons. Firstly, it is the most conceptually suitable to the matching task of our proposed models, since it aligns and compares subphrases in the user profile and job description to make predictions. Secondly, after using a more effective truncation method for the input sequences, it is the best performing of all our models on the status prediction task ($F_1 = 0.73$).

¹⁵ <https://huggingface.co/sentence-transformers>

¹⁶ https://www.sbert.net/docs/pretrained_models.html

5.6.6 Output Prediction

§5.5.3.1 discusses the classification types for the status prediction task. We retain and compare both paradigms here. The selection of classification type alters the outcome label \mathcal{Y}_i , which is associated with application triple $(\mathcal{U}_i, \mathcal{J}_i, \mathcal{Y}_i)$, where \mathcal{U}_i is the user profile, \mathcal{J}_i is the job description, and \mathcal{Y}_i is the outcome label associated with that application (described in §5.6.6).

{HIRED; REJECTED}

$$\mathcal{Y}_i = \begin{cases} 1, & \text{if status transition sequence terminates with } \textit{Hired} \\ 0, & \text{if status transition sequence terminates with } \textit{Rejected} \end{cases} \quad (5.4)$$

{INTERVIEWED; NOT INTERVIEWED}

$$\mathcal{Y}_i = \begin{cases} 1, & \text{if } \textit{Interview} \text{ or } \textit{Hired} \text{ in status transition sequence} \\ 0, & \text{if otherwise, and } \textit{Rejected} \text{ in status transition sequence} \end{cases} \quad (5.5)$$

5.6.7 Evaluation

Due to data protection concerns given the personal nature of the user features, experiments were conducted on a secure, remote virtual machine with the following specifications:

- Operating System: Linux v5.15
- CPU count: 2
- GPU count: 1
- GPU type: GRID A100-4-20C
- RAM: 16GB
- VRAM: 20GB

Encoder Model	Embedding Method	Micro F_1 Score	
		{H; R}	{I; I'}
GloVe Wiki Gigaword	Token	0.6903	0.6624
MiniLM L6 V2	Span	0.6192	0.5880
MPNet V2	Span	0.6059	0.5870
BERT NLI Mean Tokens	Span	0.6058	0.5870

Table 5.15: Results for the Matching Pipeline on the Tribepad corpus. Micro F_1 -Scores are presented for each of the encoder models and embedding methods on each of the dichotomies: {Hired; Rejected} and {Interviewed; Not Interviewed}.

Hyperparameter tuning of batch size, dropout rate, and learning rate was performed separately for each setting. Results for the Matching Pipeline experiments are shown in Table 5.15.

Overall, results for the Matching Pipeline are notably poorer than those of the truncated settings described in §5.5.3.4 and §5.5.4 ($\max F_1 = 0.73$). There are two components of the matching pipeline that may be contributing to this result: the ER process for feature selection; or the embedding process of extracted features.

The ER process for feature selection identifies and extracts the salient entities from the input sequences according to the developed schema described in §4.4 and was trained on a human-labelled corpus. In theory, this process should outperform simple truncation in terms of selecting suitable features for the model, but the results do not appear to reflect this. It is possible that the ER model is unable to properly extract these features from the input sequences due to their asyntactic property resulting from concatenating many separate fields on the user profile or job description. It may also be that this feature extraction process is neglecting certain important features in the input sequences, which would necessitate greater training and optimisation of the ER model developed to identify and extract them.

It is also possible that the embedding process of extracted features is a limiting factor of the matching pipeline in that the selected methods are not effectively encoding and representing the semantic properties of the extracted features. The discrepancy in status prediction results across embedding method shows that this component has an impact on the overall performance of the pipeline. It is notable that the embedding

method that led to the highest performance, MiniLM, had the lowest dimensionality of the implemented methods (384 vs 768, see Table 5.14). Word embeddings of 300 dimensions are common in NLP research (Mikolov et al. (2013); Pennington, Socher, and Christopher D Manning (2014); Bojanowski et al. (2017)), but there is no clear trend for sentence embeddings, which typically range from 100 to 1000 dimensions. In a study of fine-grained analysis of sentence embeddings on various prediction tasks, Adi et al. (2017) found that higher dimensionality did not always result in better performance, and sentence embeddings of 300 and 750 dimensions were shown to be ideal for different tasks. This suggests that higher dimensionality does not necessarily equate to richer semantic representations, and that lower-dimensional sentence embeddings may capture the most salient semantic relationships, resulting in more meaningful sentence representations that the model can interpret.

It may also be possible that the chosen model was unable to leverage the high dimensionality of sentence embeddings, and performance suffered as a result. In model architectures where this is the case, sentence embeddings can be projected into a lower dimensional space using processes such as Uniform Manifold Approximation and Projection (UMAP) (McInnes, Healy, and Melville, 2020), which has been shown to retain semantic properties of sentences for NLP tasks (Z. Zhang et al., 2022). To address this, we conducted experiments whereby MiniLM sentence embeddings were projected onto a lower dimensionality space of values between 2 and 300 using UMAP. Table 5.16 shows the results of these experiments, which appear to show uniformly poor results across projected dimensionalities that marginally exceed chance level. The results seem to indicate that the encoded representation of extracted entities in the Tribepad corpus lose important semantic content when projected to a lower dimensionality via UMAP, which suggests this is not an effective solution for addressing the issue of high dimensionality of sentence embeddings.

5.7 CONCLUSION

In this chapter, we have presented our proposed methods of addressing the matching problem. First, we frame the matching problem as a *recommendation* problem, where the task is to recommend n jobs to a user. We investigate various recommender systems

Dimensions	Micro F_1 Score
2	0.5045
5	0.5395
20	0.5319
100	0.5063
200	0.5147
300	0.5288
384	0.6192

Table 5.16: Results for the UMAP Experiments on the Matching Pipeline on the {Hired; Rejected} Tribepad corpus. Micro F_1 -Scores are presented for each of the dimensionalities projected by UMAP. MiniLM L6 V2 was used as the sentence embedding method. The original dimensionality, 384d, is shown for reference.

in line with related work to exceed baseline performance on the [KJRC](#) corpus and apply the models to the live Tribepad corpus.

We suggest that framing the matching problem as a *recommendation* problem is flawed in that it is tangential to its fundamental purpose, and that *predicting which jobs a user will apply to* is disparate from *recommending suitable jobs to a user*.

We propose that framing the matching problem as a *text classification* or [NLI](#) problem is much more closely aligned with the fundamental purpose of the matching problem, and suggest that learning a function to predict the *outcome status* of a given application of a user to a job is of more value than a function that predicts that the application was made. We detail our investigation into the *status prediction* task and show performance far exceeding baseline over two dichotomies of application success with a variety of [TML](#) and [DL](#) models. We observe that job descriptions in particular are composed of long sequences, exceeding the typical sequence limit of attention-based model architectures, and investigate alternative truncation methods for selecting the most salient tokens in the input sequences. We show that [TF-IDF](#) truncation yields the strongest model performance of those implemented.

We combine the [ER](#) models described in Chapter 4 with the Decomposable Attention model attuned to the status prediction task in a *matching pipeline* and experiment with methods of encoding the entities extracted as part of feature selection. We observe

poorer results relative to the more simplistic model architectures that use truncation to reduce the input sequences to the maximum sequence limit, and further observe a poorer result when sentence encoding methods are used to encode each entity span into vector representation as opposed to encoding each token in the entity span separately.

An important quality of an implemented job recommendation system is *transparency*, which refers to the ability for a non-expert to query any recommendation that has been generated by the system and understand why that decision has been made (discussed in more detail in section 2.8). The models described in this chapter feature varying degrees of transparency. For example, feature importance in Logistic Regression models is relatively straightforward, and if this were to be implemented as part of an online portal for job recommendation then it would be relatively trivial to convert feature weights to an interpretable and user-friendly interface for interactive analysis to facilitate understanding. Conversely, model interpretability for NNs and large attention-based models such as BERT is considerably more complex, although tools have been developed to facilitate human understanding (Kokhlikyan et al. (2020); Chefer, Gur, and Wolf (2021)). In any case, systems that employ job recommendation algorithms outlined in this chapter should be implemented with transparency capabilities available in the associated user interface; without this intervention, the system would risk propagating *automation bias*, leading users to give undue weight to information that may be erroneous or ill-conceived purely because it has been perceived to come from an algorithm.

In conclusion, the development of automatic matching solutions described in this chapter represents a novel contribution to the field of NLP in recruitment. The developed systems and methodologies may be integrated in online portals, benefitting jobseekers and recruiting agents alike with increased accuracy of job recommendations, saving time and resources in the pursuit of suitable jobs and candidates.

CONCLUDING REMARKS

This thesis addresses the *matching problem* which is defined as the task of developing an automatic system that recommends jobs to jobseekers. This research was motivated by the joint issues of the *information overload* problem (Dhameliya and Desai, 2019) and *filter failure* problem (Shirky, 2008), whereby the amount of data available to jobseekers in the form of job listings is too large to review and that it is difficult to effectively filter the data to find appropriate matches.

In Chapters 2 and 3 we explained preliminary concepts necessary to understand the methods and contributions of this novel research, and described related work in the fields of job recommendation and NLP to show how our research fits into the landscape of established literature.

In Chapter 4 we described the process by which we iteratively developed and refined the annotation schema and accompanying task for creating a human-labelled dataset of job descriptions with annotated salient entities, with which we trained and evaluated ER systems to identify and extract these entities automatically as part of a matching solution. We found that, by performing artificial noise induction experiments on a similarly constructed ER dataset with models that were able to learn directly from noisy annotations, we were able to propose an accuracy threshold at which we could confidently accept or reject Worker contributions on a qualification task before admitting the top-performing Workers to contribute to the live corpus. We analysed and commented on the corpus and the tagged entities within, and observed substantial agreement between annotators.

We developed, trained, and evaluated a baseline CRF model for ER on the constructed dataset and investigated its consistent errors in classification. We used this insight to develop a number of competitive models on the ER task, including BiLSTM, transformer architectures such as BERT, and CNN with the inclusion of a Crowd Layer (Rodrigues and Francisco Pereira, 2017) that enables the model to learn directly from noisy annotations without the need for label aggregation methods. The strongest performing model was

shown to be pretrained BERT incorporating text case information which achieved an F_1 score of 0.73.

In Chapter 5 we described our research into the matching problem. We considered three ways the matching problem can be framed: as a *recommendation* problem, in line with related work on the subject, whereby the task is to recommend the top n jobs for a user, and can be used to develop a job recommender system that is able to predict which jobs a user will apply to; as a *text classification* problem, whereby the task is to predict the status associated with a given application, and can be used to develop a job recommender system that is able to predict the outcome of the application; and as a *NLI* problem, which is similar in task and purpose to the *text classification* framing but treats the user features and job features as *premise* and *conclusion* and considers whether the latter *entails* the former or *contradicts* it. We described the KJRC and how it frames the matching problem as a *recommendation* problem, and developed models that were able to marginally exceed baseline performance. We observed that *location* (or rather, *proximity*) was a strong indicator of whether an application has been made by a user for a job.

We described the Tribepad corpus and showed that the data pertaining to the outcome status of each job application affords an opportunity to develop a matching solution more closely aligned with the ultimate goal of the problem, in that we can use this information to train a model that is able to predict whether a user will be successful in their application or not, which could be extended to give a more useful set of job recommendations to a user. We showed that poor results were obtained when applying application prediction models to the Tribepad corpus, most likely due to the insufficient location information contained in the data.

We presented two dichotomies for defining application success as part of the status prediction problem: {Hired; Rejected}, whereby an application is successful if the candidate is *Hired* and unsuccessful if *Rejected*; and {Interviewed; Not Interviewed}, whereby an application is successful if the candidate was invited for interview, and unsuccessful otherwise. Using both dichotomies, we developed, trained, and evaluated models for status prediction, including TML models such as Logistic Regression, BiLSTM, and BiGRU, and attention-based DL model architectures including Decomposable Attention and BERT. We observed that Logistic Regression with TF-IDF features of the concatenated input sequences, although relatively simplistic in design,

performed comparably with the more complex DL models. We hypothesised that the truncation of the input sequence length in the more complex models may be limiting predictive power. We showed support for this hypothesis in the investigation of alternative truncation methods, which showed that selecting the top n tokens by descending TF-IDF weight yielded the strongest model results.

We then combined our strongest status prediction model (Decomposable Attention) with our strongest ER model (BERT) in an end-to-end *matching pipeline* to predict the outcome status of a given application. We observed poorer results for the matching pipeline relative to our status prediction experiments and considered two reasons for this: the suboptimal performance of the ER component for feature selection; and the suboptimal encoding process whereby fixed-length vector representations for extracted entities were generated and used as input to the predictive model.

6.1 ASSESSMENT OF CONTRIBUTIONS

6.1.1 *Extracting Salient Entities from Job Descriptions*

The first research question this thesis aims to address is as follows:

RQ1 *How can salient entities in applicant profiles and job descriptions be identified and extracted for use in an applicant profile-job description matching solution?*

The scientific contributions of addressing this research question are:

- A list of entity classifications and their definitions in the form of an annotation schema
- A public, labelled dataset for the development and evaluation of entity extraction systems
- A state-of-the-art system for extracting salient entities from applicant profiles and job descriptions

Our published dataset has been used in contemporary NLP research to develop skill extraction systems (M. Zhang, Goot, and Plank, 2023), and has been cited several times in the field of skill extraction and job recommendation (M. Zhang, Kristian Nørgaard

Jensen, et al. (2022a); T. Yu et al. (2023); N. Li, Kang, and De Bie (2023); Naik, Patel, and Kannan (2023)).

6.1.2 *Matching Candidate Profiles and Job Descriptions*

The second research question this thesis aims to address is as follows:

RQ2 *How can deeper understanding of the candidate and job be used to influence an candidate profile-job description matching solution?*

The scientific contributions of addressing this research question are:

- A novel approach to applicant profile-job description matching that leverages deep insight of applicant profiles and job descriptions

6.2 LIMITATIONS

Given that the systems described and developed in this thesis are to the benefit of jobseekers and recruiting agents in reducing the time and effort of the initial stages of the hiring process, it would have been ideal to include human evaluation of these systems. For example, the matching solutions described in Chapter 5 could be implemented in an online portal that evaluates the suitability of jobs and shows a ranked list to the user in descending likelihood of application success. We would then be able to receive feedback from live users of the system in terms of how confident they were with the recommendations and whether they seemed sensible and appropriate. Furthermore, it may be possible to infer further information from the *length of engagement* of users on the platform, indicating the time taken to find an appropriate match, or perhaps a more direct metric of system quality would be the average number of applications made in a certain time frame. In the absence of human evaluation we are not able to conclude that the developed models reduce the time and effort associated with the initial stages of the hiring process, although we provide evidence in terms of F_1 that the models we develop outperform more simple models for identifying suitable matches.

In Chapter 4 we described our efforts in developing ER systems that are able to identify and extract salient entities in job descriptions. These systems were trained and

evaluated on our developed corpus of job descriptions with human-labelled entities. As such, the models are calibrated to the distinctive syntax (or lack thereof) of job descriptions, including those written formally, in the style of technical documents, or informally, in the style of advertisements. However, for data protection issues, it was not feasible to include CVs and resumes in the corpus to be annotated. As a result of their omission from the corpus, trained models may be unsuited to the identification of salient entities from candidate CVs. This issue was circumnavigated in our investigation into the development of matching solutions described in Chapter 5, since the Tribepad corpus included features from candidate profiles that had been extracted through other means; by the in-house ER systems employed by Tribepad, or by users populating distinct fields during the sign-up process on the Tribepad platform. The implementation of matching solutions in live settings may require a similar process, or alternatively an ER system trained on a corpus of candidate CVs with salient entities identified.

A further limitation of this research is the difficulty in comparing our developed systems with others. We were able to frame the matching problem as a status prediction problem (that is, where the task was to predict whether a given application would be successful or not) because of the corpus that Tribepad were able to provide, which showed the transition of status codes associated with each user-job application. As far as we are aware, there are no similar corpora available for academic use nor described by related work in this field. In the absence of similar datasets, it is difficult to compare our systems to other job recommendation systems.

6.3 FUTURE WORK

6.3.1 *Extending the Salient Entity Extraction Systems*

In Chapter 4 we developed models for ER in job descriptions, identifying spans as *Skill*, *Qualification*, *Experience*, *Occupation*, *Domain*, or *None*. Our best-performing system was used for feature selection in a developed matching solution, described in Chapter 5. However, the ER system could be extended to provide utility beyond feature selection. For example, the ER model could be used to provide information about the broader landscape of job descriptions, identifying emerging domains or new skills or the prevalence of certain qualification requirements. This insight could be leveraged to

inform workforce development strategies, and to influence new courses that develop certain skills or award certain qualifications. Additionally, in industries with specific regulatory or certification requirements, an extension of the ER system could assist organisations in ensuring that listed job descriptions comply and align with legal and industry standards. Future work regarding the extraction of salient entities in job descriptions may investigate other ways in which insights from extracted entities can be applied.

When applied as feature selection, the ER systems described in Chapter 4 convert the text of a job description (more specifically, a concatenation of all text fields pertaining to the job) into a hierarchical data structure composed of each of the defined entity classes. An extension of this process would be a system that performs this process *in reverse*; a system that converts a list of key entities into a job description document. The advantage of this process is that it may help to avoid bias in the generation of the job description document. Tools such as *textio*¹ have been developed to aid recruiting agents avoid biased and gendered terms in the creation of job documentation to ensure inclusivity. A natural language *generation* tool that converts a list of salient entities into a job description may be preferable to the current human-generated process, as the content in the document can be generated to minimise bias while including the necessary terms.

6.3.2 Extending the Matching Pipeline

In section 5.3 we described three ways in which the matching problem can be framed. By framing the matching problem as a *recommendation problem* (§5.3.1) we generate a list of n jobs that a user is most likely to apply to. In order to achieve this, we essentially evaluate the suitability of each eligible job for recommendation (that is, those within the same time window as assigned by the KJRC) and return the top n jobs of highest suitability. By framing the matching problem as a *text classification problem* (5.3.2) or *NLI problem* (5.3.3) we evaluate the suitability of each job and predict a label of 1 or 0, indicating a likely successful application (that is, *Hired* or *Interviewed*) or an unsuccessful application (*Rejected* or *Not Interviewed*). There is a slight modification that is required to this process before it could be implemented in a live system: before selecting the

¹ <https://textio.com>

more likely of the two binary classes, the models calculate the probability that a given application belongs to either class. This probability can be interpreted as a *confidence* of classification suitability, and therefore jobs can be ranked in decreasing confidence of application success. With this modification, the top n jobs of highest suitability can be shown to the user. Future work may investigate the effectiveness of this application of the matching pipeline.

Of the salient entity classifications extracted in feature selection of the matching pipeline by ER systems, *Experience* spans were the only spans that were withheld from the model. Since *Experience* spans are measures of time (for example, *at least two years* in the sentence *at least two years experience as Project Manager*) In the absence of clear information as to which *Experience* entity refers to which *Skill*, *Occupation* or *Domain* entity, they are of little value to a matching system. Future work may focus on how these extracted *Experience* entities can be used to better inform a matching solution, similar to the work conducted by José-García et al. (2022) (§3.2.5) which incorporated self-evaluated skill proficiency levels into the user-job similarity calculation.

The advantage of TML and Decomposable Attention models when applied to the matching problem is that they can be queried. Future work may focus on developing a clear interface for users to query any decision or recommendation given by the matching system. This may take the form of a visual tool that shows the extracted salient entities from both the job description and the user profile and visualises the relative importance of matches, displaying which are driving the decision to classify the job as a good or poor recommendation.

6.3.3 The Skills Delta

The *skills delta*, a concept suggested by the industry sponsor of this project, Tribepad², refers to the change in skillset a candidate would need in order to qualify for (or become an ideal candidate for) a particular job position.

Investigation of the skills delta could enrich a candidate's understanding of their current position relative to another position and could be used to create a training system to suggest courses for skill development. This process could be extended beyond immediate career progression to suggest several steps consisting of professional

² <https://tribepad.com>

development programmes and *stepping-stone* roles which could be concatenated to create a *career transition graph*, showing the route a candidate could take in order to qualify for a desired high-level position. As far as we are aware, this topic has not been studied in existing literature, although previous research into applicant profile-job description matching solutions has suggested that this may be a promising avenue for future research (Gugnani and Misra, 2020), and puts forward the use of *skill graphs* of an applicant profile to provide additional recommendations to the user, including the cost of acquiring a new skill and suggesting novel skills for development.

Bañeres and Conesa (2017) developed a recommender system that uses NLP techniques to infer the knowledge a user should acquire to be able to perform a given job. Additionally, the system is able to identify academic programmes and subjects where the user can acquire the missing skills. The methodology is simplistic, and involves performing keyword matching on the user profile (extracted from the user's LinkedIn profile through an API) and a given job's requirements to find the *acquired skills* and *lacking skills*. Then, the user would be shown the list of *lacking skills*, and recommended courses at the Universitat Oberta de Catalunya that were known to teach these skills. However, this system is unable to quantify the *level* at which a skill is required; for example, a user who states they are a *novice* at the programming language *Python* may be shown they have the necessary skillset to qualify for a Senior Software Development position that lists *expertise in Python* as a requirement for the job. Furthermore, recommendations were ranked by the number of matched terms between the user profile and the job offer. This method is too simplistic to be effective, as it assumes that jobs with the same number of *lacking skills* are equally recommendable, whereas in fact the difference in acquisition cost of the specific skills may be vastly different, for example, a user may lack the easy-to-acquire skill *teamworking* for one job, and the more difficult-to-acquire skill *financial software development* for another, but both jobs are recommended with the same rank. This system could have benefited from a vector-space approach, where user profiles and job descriptions are represented in vector space, and insight could be drawn from the skills delta to rank job recommendations and suggest courses for skill development.

The work described in this thesis could be extended to address the skills delta. In order to identify the change in skillset a candidate would need in order to qualify for a particular job position, it is first necessary to identify the skillset the candidate currently

has, and the level at which those skills can be exercised. This may be accomplished as an extension of the research described in Chapter 4, which focused on the identification and extraction of salient entities in job descriptions. Similar to the work conducted by José-García et al. (2022), it is first necessary to quantify the level at which skills in the candidate skillset are embodied by the candidate. Although José-García et al. (2022) considered user-manipulable radar charts in a visual interface to gauge the level at which users embodied skills, this approach appears to rely on both the honesty and realistic self-inference of users rather than supporting evidence in the form of experience spans or qualifications. Our research includes the analysis and extraction of these experiences and qualifications, and future work may consider linking these to the corresponding skills as a measure of the level at which that skill has been attained.

Additionally, the research described in Chapter 5 could be extended to address the skills delta. In order to evaluate the skills and experiences associated with an ‘ideal’ candidate for a given role, it may be necessary to consider previous hiring data to identify the particular attributes embodied by the successful candidate(s) in order to develop a function that can generate an *ideal candidate profile* given a job description. The matching solutions developed in Chapter 5 represent the first step toward this goal as they learn the features of the user profile and job descriptions that are most indicative of a successful application. Given the user profile and the artificially generated ‘ideal’ profile, the positive distance (that is, where the ideal profile exceeds the user profile) on each dimension could be conceptualised in terms of a skills delta. This could be integrated with an online portal in an extension of the work conducted by Bañeres and Conesa (2017) to recommend appropriate academic courses in which certain skills or qualifications could be obtained, or intermediary job roles to accrue experience in a certain occupation or domain.

Recommendations for addressing the skills delta could be concatenated from position to position to map job transition graphs, illustrated in Figure 6.1. Novel transition graphs could be created for candidates in the early stages of their career; given a desired ‘dream job’, stepping-stone jobs could be found that create a path from a user’s current position to their desired position, and training courses and qualifications could be shown that satisfy the skills delta between each step and the next.

Previous work has focused on skills gaps and career advice, particularly with a view to enhancing resilience to labour market disruption and supporting career transitions.

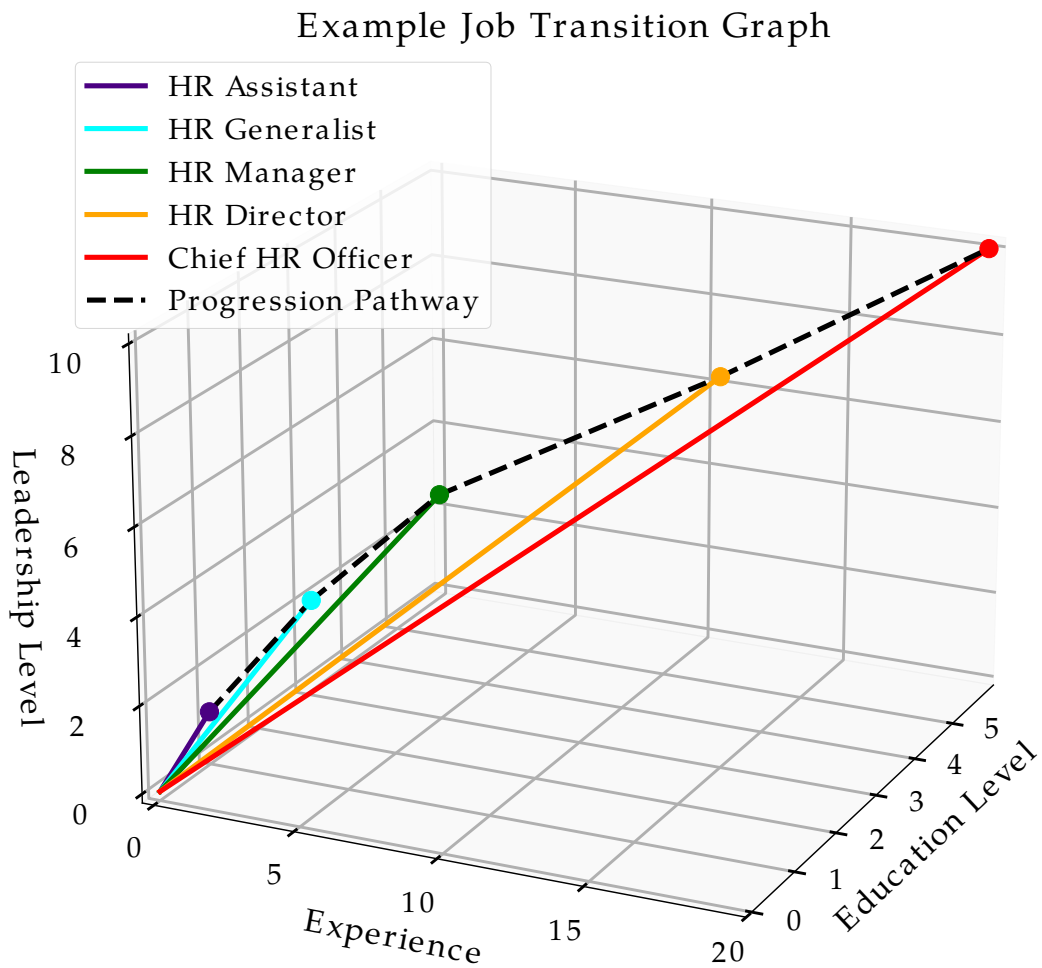


Figure 6.1: An illustration of a job transition graph. Jobs are represented in three-dimensional vector space. *Stepping-stone* jobs are found between the entry-level *HR Assistant* role and the end goal of the desired *Chief HR Officer* job. The dotted line represents the *progression pathway* which is the concatenation of skills deltas between each transition.

Kanders et al. (2020) note that AI-driven labour-displacing technology disproportionately and drastically affects certain occupations, places, and demographic groups, and emphasise the importance of developing practical strategies to help those impacted by automation. The authors generate feature vectors of jobs that capture the relative intensity of work activities in occupations (based on the ESCO skills categories), and compare representations to measure occupation similarity. The underlying assumption of this approach is that similar feature vectors represent viable job transitions, which can be used to recommend career paths for those in job sectors at risk of disruption as a result of AI. Additionally, the authors propose an approach for identifying workers' *skills gaps*, conceptually similar to the *skills delta*, which may need to be filled as they move from one occupation to another. Poor matching scores between skill pairs across the origin and destination occupations highlight large skill gaps, which can be used to focus re-skilling efforts. Although the *skills gap* is a potentially informative and useful metric for jobseekers, the method proposed does not quantify the *extent to which* a skill is required, nor does it consider the cost associated with acquiring that skill. The authors conclude that the insights from this research may guide policy making and can be used to develop strategies to help people and places build resilience to uncertainty and change in the labour market.

6.4 IMPACT OF THESIS CONTRIBUTIONS

The research described in this thesis has an impact on both academic knowledge and practical applications. Firstly, the iterative processes described in Chapter 4 by which our ER corpus and annotation schema were developed lay the foundation for future academic work collecting annotation data of good quality. Repeatedly improving task clarity and delivery to maximise annotator agreement ensures comprehensibility of task instructions and minimises systematic noise resulting from human error.

Additionally, through noise induction experiments described in Chapter 4, this research considers the relationship between AMT Worker performance and resultant ML model performance. The novel process by which we defined acceptable AMT Worker thresholds is an original contribution that enhances the reliability of annotations and has practical implications for quality control in crowdsourced data.

The development of ER systems developed for the identification and extraction of salient entities in job descriptions described in Chapter 4 represents a contribution to the field of NLP in recruitment, and has an additional impact on online portals. These systems can be implemented into internal systems with relative ease and will improve upon more simplistic *keyword extraction* systems.

Finally, the novel matching systems described in Chapter 5 have an impact on real-world applications of NLP in recruitment. With the integration of these systems and methodologies in online portals, jobseekers and recruiting agents will benefit from the increased accuracy of job recommendations, saving time and resources in the pursuit of suitable jobs and candidates.

In conclusion, the impact of this thesis extends beyond academic research and directly impacts applications of NLP in recruitment. By introducing novel methodologies of corpora construction, ER in job descriptions, and automatic matching solutions, the research described in this thesis facilitates the collection of good and unbiased data, more efficient and successful recruitment through online portals, and contributes to the academic advancement of NLP in recruitment.



APPENDIX

A.1 ENTITY RECOGNITION CORPUS

A.1.1 *Annotation Materials*

Annotation Task

Classify all relevant entities mentioned in a job description.

Note: Please read this document carefully. The definitions that we offer here for terms such as 'Skill' and 'Domain' may conflict with your own definitions. In this case, we ask that you use our definitions to complete this task.

In this task, you will be shown job descriptions from a recruitment website. For each of the job descriptions, please identify all the words in the text that belong to one of the 'Classifications' shown below.

Sometimes, a classification can contain multiple words – for example, the words 'Teaching Assistant' should together be classified as an 'Occupation'.

Classifications

- Skill
- Qualification
- Experience
- Occupation
- Domain

Classification Definitions

- **Skills**

- Examples: *'computer programming', 'French', 'data analysis', 'Microsoft Word', 'leadership', 'unloading cargo', 'problem solving', 'honesty', 'graduate recruitment strategy'*
- They are tasks that can be performed, or the attributes and abilities that enable people to perform tasks
- Includes descriptions of tasks (e.g. 'unloading cargo')
- Includes both domain-specific 'hard skills' and domain-general 'soft skills'
- Includes specific knowledge (e.g. 'understanding of marketing strategies')
- May be validated with a qualification or experience

- **Qualifications**

- Examples: *'Bachelor's Degree', 'chartership', 'National Pool Lifeguard Qualification', 'three A-levels'*
- They are official certifications obtained through taking a course or passing an exam or an appraisal
- Includes driving licenses and security clearance

- **Experience**

- Examples: *'2 years experience', 'minimum of 5 years experience'*
- They are quantified by length of time
- Does not include what the experience is of or in - for example, in the sentence *'this job requires at least 10 years of experience as a CEO'*, only the words *'at least 10 years of experience'* are the Experience

- **Occupations**

- Examples: *'Teaching Assistant', 'CEO', 'Data Analyst', 'Chef de partie'*
- These are job titles
- Includes abbreviations - for example, both *'Chief Executive Officer'* and *'CEO'* are Occupations

- **Domains**

- Examples: *'aerospace', 'oil industry', 'education', 'human resources'*
- These are areas of industry in which someone might have knowledge or experience

Clarification Questions

- If a skill refers to the organisation rather than the candidate (e.g. '*our client is a company that specialises in 'software development'*'), should this be classified as a Skill?
 - Yes. Skills should be classified as Skills whether they refer to the candidate the job is looking for or the organisation that is offering the job. In this example, the term 'software development' should be classified as a Skill.

- Can a Job Title contain a Skill? (e.g. 'Data Analysis Specialist')
 - No, these are NOT classified as Skills. The job title in its entirety is classified as an Occupation.

- How should 'background in...' sentences be classified?
 - When the background is in a skill, the Skill should be classified (e.g. '*background in teaching*' - '*teaching*' is classified as a Skill).
 - When the background is in a domain, the Domain should be classified (e.g. '*background in education*' - '*education*' is classified as a Domain).
 - The same is true for 'experience in...' sentences - if the experience is of a skill, the Skill should be classified.

- When a skill is applied to a particular task, should the details of the task be contained in the skill-term?
 - Only if it is a specific application (e.g. '*creating technical documentation*') and not a general application (e.g. '*cleaning kitchens*', where only '*cleaning*' should be classified as a Skill).

- How do you classify a sentence where Skills are split by a conjunction? (e.g. 'construction and maintenance of databases')
 - In these cases, the classification extends across all skills, and includes the application if it is relevant (in the above case, '*construction and maintenance of databases*' would be classified as a Skill in its entirety).

- If a skill is modified by a word like 'good' or 'excellent', is this part of the skill classification? (e.g. 'excellent computing skills', 'good time management')
 - No, the modifier (in the above examples, 'excellent', and 'good') are NOT part of the classification. In the phrase 'excellent computing skills', only 'computing' is the skill.

Job Description Annotation Task

Worked Examples

Example #1

These are **Skills** because they are tasks that can be performed or abilities that help people perform tasks. Although they are not separated with punctuation, capital letters mark the boundaries.

Key responsibilities include: **Develop demand plans at product, location and channel levels** **Develop monthly demand consensus and integrated Business Planning information packs** **Manage monthly demand meetings** **Analysis and review of demand plans** with **Sales**, **Marketing** and **Finance** **Analysis of historical data / trends / seasonality using forecasting models** **Collaboration** with **Finance**, **Sales** and **Marketing** to **develop long range and strategic plans**.

These are **Domains** because they are areas of industry in which one might have skills or experience.

Example #2

These are **Occupations** because they are job titles. The forward slash separates two separate classifications.

This is a **Skill** because it is an attribute that helps people perform tasks.

Our client is recruiting for an additional **experienced** **Microsoft** **Analyst/Programmer** to work on a variety of interesting **business software** **development** and **support contracts** for **corporate** and **public sector** customers **working within existing skilled teams**.

These are **Skills** because they are tasks to be performed or abilities that help people perform tasks. The conjunction 'and' separates two separate classifications.

These are **Domains** because they are areas of industry in which one might have skills or experience.

Example #3

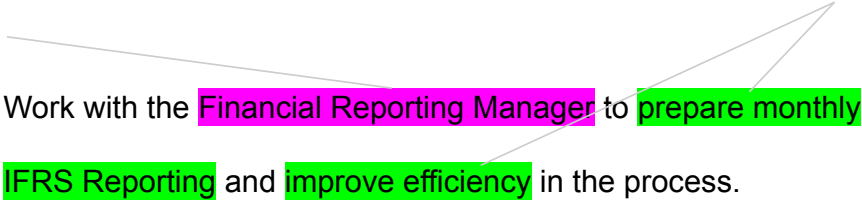
This role is based at our clients' North London office, with flexibility to work at customer sites in London and surrounding counties as required.

There are no entities in this example.

Example #4

This is an **Occupation** because it is a job title. It doesn't matter that it is not the job title that this job advert is for.

These are **Skills** because they are tasks to be performed or abilities that help people perform tasks.



Work with the **Financial Reporting Manager** to **prepare monthly IFRS Reporting** and **improve efficiency** in the process.

Example #5

This is a Skill because it is a task to be performed.

This position will be working within a fast paced environment and the successful candidate will have at least 6 months experience within a similar role.

This is an Experience because it is quantified by a length of time. It does not include the subject of the experience (here, 'within a similar role').

Example #6

These are Skills because they are abilities or attributes that help people perform tasks.

Are you a talented, ambitious, experienced and high performing sales professional?

This is an Occupation because it is a job title. If the term 'Sales' were on its own, then this might be classified as a Domain, but here it is an Occupation as it paired with the term 'Professional'.

Example #7

The term 'blue chip' may seem like a Domain, but it is not an area of industry in which someone might have knowledge or experience. It is therefore left unclassified.

We work with a number of market leading and blue chip organisations to help shape and deliver their graduate recruitment strategy.

This is a Skill because it is a task that can be performed. The term 'Graduate' is included in this classification because it is a specific application (see FAQ document for more clarification).

Example #8

These are Skills because they are tasks that can be performed.

Key responsibilities will include: Identifying opportunities, implementing and embedding change to deliver best practice.

This is not part of the Skill classification because it is a general application of the skill (see FAQ document for more clarification).

Label	P	R	F_1	Support
B-Skill	0.71	0.52	0.60	796
I-Skill	0.60	0.77	0.67	1439
B-Qualification	0.75	0.58	0.65	31
I-Qualification	0.52	0.59	0.56	37
B-Occupation	0.80	0.68	0.74	142
I-Occupation	0.85	0.75	0.80	159
B-Experience	0.88	0.54	0.67	13
I-Experience	0.57	1.00	0.72	17
B-Domain	0.35	0.55	0.43	60
I-Domain	0.26	0.26	0.26	39
micro avg	0.63	0.67	0.65	2733
macro avg	0.63	0.62	0.61	2733
weighted avg	0.65	0.67	0.65	2733

Table A.1: Results for BERT base ‘uncased’ model (fine-tuned on preprocessed data). Precision, Recall, and F_1 -Score are presented.

A.2 ER MODEL RESULTS

Tables A.1 through A.4 show the full class breakdown of ER model results.

A.3 TRIBEPAD MATCHED USER PROFILE - JOB DESCRIPTION CORPUS

Label	P	R	F_1	Support
B-Skill	0.62	0.45	0.52	796
I-Skill	0.54	0.65	0.59	1439
B-Qualification	0.57	0.55	0.56	31
I-Qualification	0.38	0.32	0.35	37
B-Occupation	0.57	0.42	0.48	142
I-Occupation	0.62	0.52	0.56	159
B-Experience	0.75	0.23	0.35	13
I-Experience	0.50	0.59	0.54	17
B-Domain	0.31	0.32	0.31	60
I-Domain	0.25	0.23	0.24	39
micro avg	0.55	0.55	0.55	2733
macro avg	0.51	0.43	0.45	2733
weighted avg	0.56	0.55	0.55	2733

Table A.2: Results for DistilBERT base ‘uncased’ model (fine-tuned on preprocessed data). Precision, Recall, and F_1 -Score are presented.

Label	P	R	F_1	Support
B-Skill	0.80	0.60	0.69	971
I-Skill	0.64	0.84	0.73	1595
B-Qualification	0.82	0.63	0.71	49
I-Qualification	0.51	0.47	0.49	40
B-Occupation	0.93	0.86	0.90	216
I-Occupation	0.93	0.89	0.91	241
B-Experience	0.60	0.46	0.52	13
I-Experience	0.64	0.78	0.70	18
B-Domain	0.51	0.59	0.55	93
I-Domain	0.45	0.55	0.50	56
micro avg	0.70	0.75	0.73	3292
macro avg	0.68	0.67	0.67	3292
weighted avg	0.72	0.75	0.73	3292

Table A.3: Results for BERT base ‘cased’ model (fine-tuned on preprocessed data). Precision, Recall, and F_1 -Score are presented.

Label	P	R	F_1	Support
B-Skill	0.79	0.59	0.68	1046
I-Skill	0.66	0.80	0.72	1660
B-Qualification	0.77	0.61	0.68	54
I-Qualification	0.61	0.58	0.60	43
B-Occupation	0.92	0.84	0.88	220
I-Occupation	0.95	0.82	0.88	254
B-Experience	0.78	0.54	0.64	13
I-Experience	0.70	0.78	0.74	18
B-Domain	0.50	0.60	0.55	99
I-Domain	0.52	0.53	0.52	57
micro avg	0.72	0.72	0.72	3464
macro avg	0.72	0.67	0.69	3464
weighted avg	0.73	0.72	0.72	3464

Table A.4: Results for BERT base ‘multilingual cased’ model (fine-tuned on preprocessed data). Precision, Recall, and F_1 -Score are presented.

A.3.1 Status Codes

Status Code	Short Description	Frequency	Proportion	Description
0	Incomplete	8,682,446	37.5%	Candidate started an application but did not submit it. May be a complete application bar submission or entirely incomplete.
1	Shortlisted	3,015,556	13.0%	The Recruiter or System has shortlisted the candidate because they passed pre-screening assessments.
2	Rejected	5,841,138	25.2%	The Recruiter or System rejected the candidate as unsuitable.
3	Hired	221,321	1.0%	Candidate was offered the position and accepted.
4	Interview	462,491	2.0%	Candidate was selected for interview, or has been interviewed and awaiting decision.
5	Offered	114,599	0.5%	Candidate offered the role but not yet accepted.
6	Assessment	0	0.0%	Candidate required to pass bespoke assessment for position.
7	Accepted	48,335	0.2%	Candidate offered the role and accepted.
8	In Review	4,597,105	19.8%	No single definition. Generally means the candidate passed pre-screening but the Recruiter has not yet made a decision or full review. Pre-interview but may be pre or post shortlist.
9	Withdrawn	194,012	0.8%	The Candidate or Recruiter has withdrawn the Candidate's application.

Table A.5: A summary of data fields in the apps dataset.

Field	Description
user_id	The id of the user that the career item refers to.
jobtitle	The job title of the career item.
description	The description of the career item.
employer	The name of the company for the career item.
startdate	The date the career item started.
enddate	The date the career item ended.

Table A.6: A summary of data fields in the career dataset.

A.3.2 *career data*

Field	Description
user_id	The id of the user that the education item refers to.
qualification	The short description of education item, if applicable.
description	The description of the education item.
educationlevel	The level of the education item (e.g. Bachelors, Secondary).
institution	The institution at which the education item was attained.
startdate	The date the education item started.
enddate	The date the education item ended.

Table A.7: A summary of data fields in the education dataset.

A.3.3 *education data*

A.3.4 *job data*

Field	Description
job_id	The unique id of the job.
job_title	The title of the job.
summary_external	The summary of the job intended for Candidates.
summary_internal	The summary of the job intended for internal use.
main_responsibilities	A description of the job responsibilities.
ideal_candidate	A description of the ideal candidate for the job.
location_city	The city in which the job is located.
location_country	The country in which the job is located.
location_country	The country in which the job is located.
no_of_positions	The number of open positions for the job.
open_date	The date from which the job started collected applications.
closed_date	The date at which the job ceased accepting applications.
salary_frequency	The code denoting frequency of pay, e.g. 'per hour', 'per annum'.
salary_from	The lower bound of salary. Used in conjunction with salary_frequency.
salary_to	The upper bound of salary. Used in conjunction with salary_frequency.
shift_hours	The number of hours per shift, if applicable.

Table A.8: A summary of data fields in the job dataset.

Field	Description
user_id	The id of the user associated with the skill. Duplicate skills across multiple users are contained in unique skill items.
skill	The skill term extracted from the user profile.

Table A.9: A summary of data fields in the skills dataset.

A.3.5 *skills data*

Field	Description
user_id	The id of the user.
created_time	The date the user profile was created.
address_city	The city in which the user is currently located.
address_county	The county in which the user is currently located.
address_country	The country in which the user is currently located.

Table A.10: A summary of data fields in the user dataset.

A.3.6 *user data*

A.4 ETHICAL APPROVAL

The following documents are included here:

- **Ethics Application Form** - includes all details of the submitted ethics application for this research project.
- **Ethics Approval Letter** - confirmation of the approval of the ethics application form.
- **Ethics Post-Approval Edit** - an email chain describing requested amendments to the approved ethics application as well as approval from the ethics committee.

A.4.0.1 *Ethics Application Form*

Application 036039

Section A: Applicant details

Date application started:
Fri 17 July 2020 at 09:51

First name:
Thomas

Last name:
Green

Email:
tafgreen1@sheffield.ac.uk

Programme name:
PhD Project

Module name:
PhD Project
Last updated:
24/08/2020

Department:
Computer Science

Applying as:
Postgraduate research

Research project title:
Using NLP to Resolve Mismatches Between Jobseekers and Positions in Recruitment

Has your research project undergone academic review, in accordance with the appropriate process?
Yes

Similar applications:
- *not entered* -

Section B: Basic information

Supervisor

Name

Email

Diana Maynard

d.maynard@sheffield.ac.uk

Proposed project duration

Start date (of data collection):
Thu 1 October 2020

Anticipated end date (of project)
Sun 1 October 2023

3: Project code (where applicable)

Project code
- *not entered* -

Suitability

Takes place outside UK?

No

Involves NHS?

No

Health and/or social care human-interventional study?

No

ESRC funded?

No

Likely to lead to publication in a peer-reviewed journal?

Yes

Led by another UK institution?

No

Involves human tissue?

No

Clinical trial or a medical device study?

No

Involves social care services provided by a local authority?

No

Is social care research requiring review via the University Research Ethics Procedure

No

Involves adults who lack the capacity to consent?

No

Involves research on groups that are on the Home Office list of 'Proscribed terrorist groups or organisations'?

No

Indicators of risk

Involves potentially vulnerable participants?

No

Involves potentially highly sensitive topics?

No

Section C: Summary of research

1. Aims & Objectives

Nearly everyone has a CV (or at least a career and/or education history).

Nearly every non-sole trader business recruits people.

Most people want to progress from the job they are in - over 60% of people are always at least passively looking for a job.

CV-to-Job Spec matching technologies are generally underperforming right now, but there is plenty of data available for machine learning solutions to improve performance. This would improve the quality and objectivity of recruitment, reduce the number of irrelevant applications recruiters receive, reduce the likelihood that a good candidate is overlooked, save time and lead to faster filling of open positions, and enhance the candidate experience. One challenge is that job seekers are reluctant to provide structured data about themselves, and prefer to "throw" a CV over to the recruiter and then "throw" their CV over to the next recruiter at the next company and so on. Because of this, the data from a job seeker is often not in structured form. Similarly, a lot of the data the recruiter gives about the job is also unstructured - provided in a number of paragraphs of text. Yet all the key information is available to match someone. The job description specifies key skills in the text, the ideal candidate, the soft skills, the must-haves, the skills that are beneficial but not essential, and

so on.

This project will involve researching a 'matching solution' that takes all of the above into consideration - the unstructured data needs to be parsed using Natural Language Processing (NLP) techniques, the key components extracted, and fed into a machine learning algorithm to determine how close someone is for a job and what the key differences are. This could help people see what skills they need to acquire in order to progress (their 'skills delta'), and that data could eventually feed into a training system to suggest courses (college, university or other).

2. Methodology

A wide range of Information Extraction methods (including knowledge-engineering and supervised learning methods) will be explored to tackle parsing of unstructured CV and job specification data, and both shallow-learning and deep-learning methods will be considered for the 'matching' component of the CV-job spec matching solution.

3. Personal Safety

Have you completed your departmental risk assessment procedures, if appropriate?

Yes

Raises personal safety issues?

No

It is not necessary for researchers to work outside normal hours, conduct activities off University premises, work with potentially threatening people, nor conduct activities in a potentially dangerous environment.

All contributions to this project can be made on a desktop PC in a safe office environment on University premises.

Section D: About the participants

1. Potential Participants

Rather than identifying potential participants, this project will require obtaining a large number of job descriptions from recruitment sites and a large number of candidate CVs. Some of this data will be supplied by the industry partner, TribePad. Permissions for the sharing of data have already been given by candidates to the respective recruitment sites and explicit permission will be sought from recruitment sites before any personal data is obtained.

2. Recruiting Potential Participants

We will seek explicit, specific permission from each provider to use data obtained from their platform, and adhere to their policies regarding the use of data for research in existing agreements with their customers.

2.1. Advertising methods

Will the study be advertised using the volunteer lists for staff or students maintained by CiCS? No

- *not entered* -

3. Consent

Will informed consent be obtained from the participants? (i.e. the proposed process) No

Since data is acquired from a provider and not from the individual themselves (for instance, in the context of candidate CVs, these are obtained from a recruitment site API rather than the CV authors themselves), we will not be seeking informed consent from the individuals. We will, however, seek explicit, specific permission from each provider to use data obtained from their platform, and adhere to their policies regarding the use of data for research in existing agreements with their customers.

4. Payment

Will financial/in kind payments be offered to participants? No

5. Potential Harm to Participants

What is the potential for physical and/or psychological harm/distress to the participants?

None. Data will be anonymised by the researchers prior to any analysis to eliminate the risk of individuals being personally identified.

How will this be managed to ensure appropriate protection and well-being of the participants?

Data will be anonymised by the researchers prior to any analysis to eliminate the risk of individuals being personally identified. However, we acknowledge that during this process, researchers will have access to personal information such as name and DOB. There is, therefore, a risk of data breach associated with handling this sensitive data. Encryption and storage of sensitive data is detailed in a later section.

In order to address potential biases in the data such as EDI issues, institutional bias, and systemic biases, special attention will be paid during the course of this project to implementing active measures within a CV-Job Spec matching system, and to ensure that this system promotes diversity and equity goals.

Section E: About the data

1. Data Processing

Will you be processing (i.e. collecting, recording, storing, or otherwise using) personal data as part of this project? (Personal data is any information relating to an identified or identifiable living person).

Yes

Which organisation(s) will act as Data Controller?

University of Sheffield only

2. Legal basis for processing of personal data

The University considers that for the vast majority of research, 'a task in the public interest' (6(1)(e)) will be the most appropriate legal basis. If, following discussion with the UREC, you wish to use an alternative legal basis, please provide details of the legal basis, and the reasons for applying it, below:

- *not entered* -

Will you be processing (i.e. collecting, recording, storing, or otherwise using) 'Special Category' personal data?

No

3. Data Confidentiality

What measures will be put in place to ensure confidentiality of personal data, where appropriate?

Data supplied by the industry partner will be anonymised prior to data handover, and all data acquired from other sources will be anonymised by the researcher.

4. Data Storage and Security

In general terms, who will have access to the data generated at each stage of the research, and in what form

Data supplied by the industry partner will be anonymised prior to data handover, and so the PhD candidate and their supervisory team will have access to this data in its anonymised form.

Data acquired through other sources will be anonymised by the PhD candidate and this will be stored in anonymised form for access by the supervisory team and industry partner.

What steps will be taken to ensure the security of data processed during the project, including any identifiable personal data, other than those already described earlier in this form?

Data will be stored on an encrypted laptop and backed up to two locations:

1) the University of Sheffield Google Drive, and

2) University of Sheffield research data storage,

both of which can only be accessed by authorised members of the project.

Any sensitive data (as defined by the GDPR) that is stored on portable electronic devices will be protected by encryption software to ISO IEC 27001 standard. Any sensitive data that needs to be transmitted electronically will first be encrypted.

The University's Information Security Policies as well as the industry partner's Data Protection and Retention Policies will be abided by at all times.

An incremental copy of data is automatically taken every night (and kept for 28 days) and a full copy is taken every month. Backup integrity will be tested every month by restoring the data to a virtual machine and running a spot check and macro test to match against the production database.

Will all identifiable personal data be destroyed once the project has ended?

Yes

Please outline when this will take place (this should take into account regulatory and funder requirements).

Data supplied by our industry partner will be destroyed upon completion of the project to ensure that any data under their ownership is not accessed by external parties. The industry partner will be free to retain any of this data in accordance with their internal policies.

Data obtained freely through other methods will be prepared for future sharing and potential secondary analysis, and made available for at least 10 years as per UKRI guidelines. Data will be deposited for archiving and re-use with the UKRI data service provider, UKDA, at the end of the project. Data will be available on request only.

Section F: Supporting documentation

Information & Consent

Participant information sheets relevant to project?

No

Consent forms relevant to project?

No

Additional Documentation

[Document 1081827 \(Version 1\)](#)

[All versions](#)

This is the data management plan for the project which describes how data will be acquired, processed, stored, and archived in greater detail.

External Documentation

- not entered -

Section G: Declaration

Signed by:

Thomas AF Green

Date signed:

Thu 13 August 2020 at 14:13

Official notes

- not entered -

A.4.0.2 *Ethics Approval Letter*



Downloaded: 21/01/2021
Approved: 24/08/2020

Thomas Green
Registration number: 190185826
Computer Science
Programme: PhD Project

Dear Thomas

PROJECT TITLE: Using NLP to Resolve Mismatches Between Jobseekers and Positions in Recruitment
APPLICATION: Reference Number 036039

On behalf of the University ethics reviewers who reviewed your project, I am pleased to inform you that on 24/08/2020 the above-named project was **approved** on ethics grounds, on the basis that you will adhere to the following documentation that you submitted for ethics review:

- University research ethics application form 036039 (form submission date: 13/08/2020); (expected project end date: 01/10/2023).

If during the course of the project you need to [deviate significantly from the above-approved documentation](#) please inform me since written approval will be required.

Your responsibilities in delivering this research project are set out at the end of this letter.

Yours sincerely

Com Ethics
Ethics Administrator
Computer Science

Please note the following responsibilities of the researcher in delivering the research project:

- The project must abide by the University's Research Ethics Policy:
<https://www.sheffield.ac.uk/rs/ethicsandintegrity/ethicspolicy/approval-procedure>
- The project must abide by the University's Good Research & Innovation Practices Policy:
https://www.sheffield.ac.uk/polopoly_fs/1.671066!/file/GRIPPpolicy.pdf
- The researcher must inform their supervisor (in the case of a student) or Ethics Administrator (in the case of a member of staff) of any significant changes to the project or the approved documentation.
- The researcher must comply with the requirements of the law and relevant guidelines relating to security and confidentiality of personal data.
- The researcher is responsible for effectively managing the data collected both during and after the end of the project in line with best practice, and any relevant legislative, regulatory or contractual requirements.

A.4.0.3 *Ethics Post-Approval Edit*

The following changes to the ethics application form were made after its approval, on 30th November 2020. These changes were acknowledged and approved on 2nd December 2020.

SECTION C: SUMMARY OF RESEARCH

2. Methodology (to be added)

This research will include the construction of a labelled dataset of job descriptions, which will involve human annotation of job descriptions.

SECTION D: ABOUT THE PARTICIPANTS

1. Potential Participants (this has been rewritten)

This research will include human annotation of job descriptions for the construction of a labelled dataset. For this task, participants will be registered Workers on the crowdsourcing website Amazon Mechanical Turk (MTurk; <https://www.mturk.com>). Additionally, this research will require obtaining a large number of job descriptions from recruitment sites and a large number of candidate CVs. Some of this data will be supplied by the industry partner, TribePad. Permissions for the sharing of data have already been given by candidates to the respective recruitment sites and explicit permission will be sought from recruitment sites before any personal data is obtained.

3. Consent (this has been rewritten)

Answer: Yes.

For annotation data sourced through Amazon Mechanical Turk, registered Workers will be required to give their consent before participating. Workers who click on this job through the AMT platform will be immediately taken to the standard Amazon Mechanical Turk Informed Consent Form which must be read and accepted before they are permitted to work on the task. For data acquired through other means, we will not

be seeking informed consent from the individuals themselves (for instance, in the context of candidate CVs, this data is obtained from recruitment site APIs rather than the CV authors themselves). We will, however, seek explicit, specific permission from each provider to use data obtained from their platform, and adhere to their policies regarding the use of data for research in existing agreements with their customers.

4. Payment (this has been rewritten)

Answer: Yes.

For annotation data sourced through Amazon Mechanical Turk, Workers will be remunerated according to the standard pricing structure using the UK minimum wage (£8.72 per hour at time of writing) as a guideline.

5. Potential Harm to Participants (to be added)

For annotation data sourced through Amazon Mechanical Turk, there is very little risk of harm or discomfort to participants in the annotation task. The user interface for this task will be clear and simple, instructions will be easy to follow and unambiguous, and participants can withdraw at any point and will be compensated for their contribution.

SECTION E: ABOUT THE DATA

3. Data Confidentiality (to be added)

Human annotated data acquired through Amazon Mechanical Turk is anonymised by Amazon prior to data handover; each worker is assigned a unique worker ID which ensures that no personally identifiable information is accessible to the requester.

In answer to the Lead Ethics Reviewer's questions:

- **Question:** What kind of annotations are expected from the human annotators? What is exactly what they will be asked to do?
 - **Answer:** Human annotators will be provided with sentences from job descriptions sourced from a public dataset of job descriptions published under a CCo license. Through the MTurk platform, annotators will be asked to

read a short task instructions sheet and highlight entities within a sentence, assigning each of them a unique classification (Hard Skill, Soft Skill, Qualification, Experience, Occupation, Domain) or state that no entities exist within the sentence. Annotators will be compensated for each annotated sentence, regardless of their performance or number of annotated sentences.

- **Question:** If a Public dataset is going to be built from this data, what will be the conditions for the Public to have access to the dataset?
 - **Answer:** The public dataset will be published under a CCo license and will be available to everyone to use, copy, modify, or distribute without having to seek explicit permission. Publishing under a Creative Commons license is standard procedure for similar corpora published on data science community site Kaggle (<https://www.kaggle.com>).

BIBLIOGRAPHY

- Adi, Yossi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg (2017). "Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks." In: arXiv: [1608.04207 \[cs.CL\]](https://arxiv.org/abs/1608.04207).
- Agarap, Abien Fred (2019). *Deep Learning using Rectified Linear Units (ReLU)*. arXiv: [1803.08375 \[cs.NE\]](https://arxiv.org/abs/1803.08375).
- Aggarwal, Charu, Alexander Hinneburg, and Daniel A. Keim (2001). "On the surprising behavior of distance metrics in high dimensional space." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1973, pp. 420–434. ISSN: 16113349. DOI: [10.1007/3-540-44503-x_27](https://doi.org/10.1007/3-540-44503-x_27).
- Ahmed, Shabbir, Mahamudul Hasan, Md. Nazmul Hoq Salim, and Muhammad Abdullah Adnan (Oct. 2016). "User interaction analysis to recommend suitable jobs in career-oriented social networking sites." In: pp. 1–6. DOI: [10.1109/ICODSE.2016.7936143](https://doi.org/10.1109/ICODSE.2016.7936143).
- Alabdulkareem, Ahmad, Morgan R. Frank, Lijun Sun, Bedoor AlShebli, César Hidalgo, and Iyad Rahwan (2018). "Unpacking the polarization of workplace skills." In: *Science Advances* 4.7, eaa06030. DOI: [10.1126/sciadv.aao6030](https://doi.org/10.1126/sciadv.aao6030).
- Almalis, Nikolaos D., George A. Tsihrintzis, and Nikolaos Karagiannis (July 2014). "A content based approach for recommending personnel for job positions." In: IEEE, pp. 45–49. ISBN: 978-1-4799-6171-9. DOI: [10.1109/IISA.2014.6878720](https://doi.org/10.1109/IISA.2014.6878720). URL: <http://ieeexplore.ieee.org/document/6878720/>.
- Almalis, Nikolaos D., George A. Tsihrintzis, Nikolaos Karagiannis, and Aggeliki D. Strati (2016). "FoDRA - A new content-based job recommendation algorithm for job seeking and recruiting." In: *IISA 2015 - 6th International Conference on Information, Intelligence, Systems and Applications*. DOI: [10.1109/IISA.2015.7388018](https://doi.org/10.1109/IISA.2015.7388018).
- Balog, Krisztian, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si (2012). "Expertise Retrieval." In: *Foundations and Trends® in Information Retrieval* 6.2–3, pp. 127–256. ISSN: 1554-0669. DOI: [10.1561/15000000024](https://doi.org/10.1561/15000000024). URL: <http://dx.doi.org/10.1561/15000000024>.

- Bañeres, David and Jordi Conesa (2017). "A Life-long learning recommender system to Promote Employability." In: *International Journal of Emerging Technologies in Learning* 12 (6), pp. 77–93. ISSN: 18630383. DOI: [10.3991/ijet.v12i06.7166](https://doi.org/10.3991/ijet.v12i06.7166).
- Bansal, Shivam, Aman Srivastava, and Anuja Arora (Jan. 2017). "Topic Modeling Driven Content Based Jobs Recommendation Engine for Recruitment Industry." In: *Procedia Computer Science* 122, pp. 865–872. ISSN: 1877-0509. DOI: [10.1016/J.PROCS.2017.11.448](https://doi.org/10.1016/J.PROCS.2017.11.448).
- Bastian, Mathieu, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, and Hyungjin Kim (2014). "Linked in skills: Large-scale topic extraction and inference." In: *RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems* (October), pp. 1–8. DOI: [10.1145/2645710.2645729](https://doi.org/10.1145/2645710.2645729).
- Bennett, James and Stan Lanning (2007). "The Netflix Prize." In: *KDD Cup and Workshop*, pp. 3–6. ISSN: 1554351X.
- Bogen, Miranda and Aaron Rieke (2018). "Help Wanted: An Examination of Hiring Algorithms, Equity, and Bias." In: *Upturn* (December), pp. 40–73.
- Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2017). "Enriching Word Vectors with Subword Information." In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. ISSN: 2307-387X. DOI: [10.1162/tacl_a_00051](https://doi.org/10.1162/tacl_a_00051).
- Bowman, Samuel R., Gabor Angeli, Christopher Potts, and Christopher D. Manning (2015). "A large annotated corpus for learning natural language inference." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.
- Burke, Robin (2002). "Hybrid Recommender Systems: Survey and Experiments." In: *User Modeling and User-Adapted Interaction* 12. ISSN: 02545330. DOI: [10.1023/A:1021240730564](https://doi.org/10.1023/A:1021240730564).
- Chefer, Hila, Shir Gur, and Lior Wolf (2021). *Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers*. arXiv: [2103.15679 \[cs.CV\]](https://arxiv.org/abs/2103.15679).
- Choudhary, Savita, Siddanth Koul, Shridhar Mishra, Anunay Thakur, and Rishabh Jain (2016). "Collaborative job prediction based on Naïve Bayes Classifier using python platform." In: *2016 International Conference on Computation System and Information Technology for Sustainable Solutions, CSITSS 2016*, pp. 302–306. DOI: [10.1109/CSITSS.2016.7779375](https://doi.org/10.1109/CSITSS.2016.7779375).

- Commission, European, Social Affairs Directorate-General for Employment, and Inclusion (2019). *ESCO handbook – European skills, competences, qualifications and occupations*. Publications Office. DOI: [doi/10.2767/451182](https://doi.org/10.2767/451182).
- Covington, Paul, Jay Adams, and Emre Sargin (2016). “Deep neural networks for youtube recommendations.” In: *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198.
- Cunningham, Pádraig and Sarah Jane Delany (2020). “k-Nearest neighbour classifiers 2nd edition (with python examples).” In: *arXiv* (1), pp. 1–22. ISSN: 23318422.
- Dastin, Jeffrey (Oct. 11, 2018). “Amazon scraps secret AI recruiting tool that showed bias against women.” In: *Thomson Reuters*. URL: <https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G> (visited on 09/06/2023).
- Deleger, Louise, Q. Li, Todd Lingren, Megan Kaiser, Katalin Molnar, Laura Stoutenborough, Michal Kouril, Keith Marsolo, and Imre Solti (2012). “Building gold standard corpora for medical natural language processing tasks.” In: *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium 2012*, pp. 144–153. ISSN: 1942597X.
- Devlin, Jacob, Ming Wei Chang, Kenton Lee, and Kristina Toutanova (2019). “BERT: Pre-training of deep bidirectional transformers for language understanding.” In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference 1* (Mlm), pp. 4171–4186.
- Dhameliya, Juhi and Nikita Desai (2019). “Job Recommender Systems: A Survey.” In: *2019 Innovations in Power and Advanced Computing Technologies, i-PACT 2019*, pp. 1–5. DOI: [10.1109/i-PACT44901.2019.8960231](https://doi.org/10.1109/i-PACT44901.2019.8960231).
- Fareri, S., G. Fantoni, F. Chiarello, E. Coli, and A. Binda (2020). “Estimating Industry 4.0 impact on job profiles and skills using text mining.” In: *Computers in Industry* 118, p. 103222. ISSN: 0166-3615. DOI: <https://doi.org/10.1016/j.compind.2020.103222>.
- Fournier, Quentin, Gaé tan Marceau Caron, and Daniel Aloise (July 2023). “A Practical Survey on Faster and Lighter Transformers.” In: *ACM Computing Surveys* 55.14s, pp. 1–40. DOI: [10.1145/3586074](https://doi.org/10.1145/3586074). URL: <https://doi.org/10.11452F3586074>.
- Freedman, David, Robert Pisani, and Roger Purves (2007). “Statistics (international student edition).” In: *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

- Galron, Daniel A., Yuri M. Brovman, Jin Chung, Michal Wieja, and Paul Wang (2018). "Deep Item-based Collaborative Filtering for Sparse Implicit Feedback." In: *CoRR* abs/1812.10546. arXiv: 1812.10546. URL: <http://arxiv.org/abs/1812.10546>.
- Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé, and Kate Crawford (2018). "Datasheets for Datasets." In: ISSN: 2331-8422. URL: <http://arxiv.org/abs/1803.09010>.
- Gershgorn, Dave (Oct. 22, 2018). "Companies are on the hook if their hiring algorithms are biased." In: *Quartz*. URL: <https://qz.com/1427621/companies-are-on-the-hook-if-their-hiring-algorithms-are-biased> (visited on 09/06/2023).
- Gholamy, Afshin, Vladik Kreinovich, and Olga Kosheleva (2018). "Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation." In: URL: <https://api.semanticscholar.org/CorpusID:7467506>.
- Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. Adaptive computation and machine learning. MIT Press. ISBN: 9780262035613. URL: <https://books.google.co.in/books?id=Np9SDQAAQBAJ>.
- Gugnani, Akshay, Vinay Kumar Reddy Kasireddy, and Karthikeyan Ponnalagu (2019). "Generating unified candidate skill graph for career path recommendation." In: *IEEE International Conference on Data Mining Workshops, ICDMW 2018-Novem*, pp. 328–333. ISSN: 23759259. DOI: 10.1109/ICDMW.2018.00054.
- Gugnani, Akshay and Hemant Misra (Apr. 2020). "Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation." In: *Proceedings of the AAAI Conference on Artificial Intelligence 34 (08)*, pp. 13286–13293. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i08.7038. URL: <https://aaai.org/ojs/index.php/AAAI/article/view/7038>.
- Hawkes, Alan G. (1971). "Spectra of Some Self-Exciting and Mutually Exciting Point Processes." In: *Biometrika* 58.1, pp. 83–90. ISSN: 00063444. URL: <http://www.jstor.org/stable/2334319>.
- Hayes, Andrew F. and Klaus Krippendorff (2007). "Answering the Call for a Standard Reliability Measure for Coding Data." In: *Communication Methods and Measures* 1.1, pp. 77–89. DOI: 10.1080/19312450709336664.
- Hoang, Phuong, Thomas Mahoney, Faizan Javed, and Matt McNair (2018). "Large-scale occupational skills normalization for online recruitment." In: *AI Magazine* 39 (1), pp. 5–14. ISSN: 07384602. DOI: 10.1609/aimag.v39i1.2775.

- Hovy, Dirk, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy (2013). "Learning Whom to Trust with MACE." In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1120–1130. ISSN: 00021369.
- Hripcsak, George and Adam S. Rothschild (2005). "Agreement, the F-measure, and reliability in information retrieval." In: *Journal of the American Medical Informatics Association* 12 (3), pp. 296–298. ISSN: 10675027. DOI: [10.1197/jamia.M1733](https://doi.org/10.1197/jamia.M1733).
- Huang, Zhiheng, Wei Xu, and Kai Yu (2015). *Bidirectional LSTM-CRF Models for Sequence Tagging*. arXiv: [1508.01991](https://arxiv.org/abs/1508.01991) [cs.CL].
- Iacovou, Charalambos L and Vermont Ronald L Thompson (2002). "Job Selection Preferences Of Business Students." In: *Journal of Applied Business Research* 20 (1), pp. 87–98.
- Jiang, Ming, Jennifer D'Souza, Sören Auer, and J. Stephen Downie (2020). *Improving Scholarly Knowledge Representation: Evaluating BERT-based Models for Scientific Relation Classification*. arXiv: [2004.06153](https://arxiv.org/abs/2004.06153) [cs.DL].
- José-García, Adán, Alison Sneyd, Ana Melro, Ollagnier Anaïs, Georgina Tarling, Haiyang Zhang, Mark Stevenson, Richard Everson, and Rudy Arthur (Dec. 2022). "C3-IoC: A Career Guidance System for Assessing Student Skills using Machine Learning and Network Visualisation." In: *International Journal of Artificial Intelligence in Education*. DOI: [10.1007/s40593-022-00317-y](https://doi.org/10.1007/s40593-022-00317-y).
- Kanders, K, J Djumalieva, C Sleeman, and J Orlik (2020). *Mapping Career Causeways: Supporting Workers at Risk*. Tech. rep. Nesta.
- Karakatsanis, Ioannis, Wala AlKhader, Frank MacCrorry, Armin Alibasic, Mohammad Atif Omar, Zeyar Aung, and Wei Lee Woon (Apr. 2017). "Data mining approach to monitoring the requirements of the job market: A case study." In: *Information Systems* 65, pp. 1–6. ISSN: 03064379. DOI: [10.1016/j.is.2016.10.009](https://doi.org/10.1016/j.is.2016.10.009).
- Khobreh, Marjan, Fazel Ansari, Madjid Fathi, Reka Vas, Stefan T. Mol, Hannah A. Berkers, and Krisztian Varga (2016). *An Ontology-Based Approach for the Semantic Representation of Job Knowledge*. DOI: [10.1109/TETC.2015.2449662](https://doi.org/10.1109/TETC.2015.2449662).
- Kivimäki, Ilkka, Alexander Panchenko, Adrien Dessy, Dries Verdegem, Pascal Francq, Cédric Fairon, Hugues Bersini, and Marco Saerens (2020). "A graph-based approach to skill extraction from text." In: *Proceedings of TextGraphs@EMNLP 2013: The 8th Workshop on Graph-Based Methods for Natural Language Processing* (October), pp. 79–87.

- Kokhlikyan, Narine et al. (2020). *Captum: A unified and generic model interpretability library for PyTorch*. arXiv: [2009.07896](https://arxiv.org/abs/2009.07896) [cs.LG].
- Kolen, John F. and Stefan C. Kremer (2001). "Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies." In: *A Field Guide to Dynamical Recurrent Networks*, pp. 237–243. DOI: [10.1109/9780470544037.ch14](https://doi.org/10.1109/9780470544037.ch14).
- Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix Factorization Techniques for Recommender Systems." In: *IEEE COMPUTER*. DOI: [10.1.1.147.8295](https://doi.org/10.1.1.147.8295).
- Lafferty, John, Andrew McCallum, and Fernando Pereira (1999). "Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data Abstract." In: 2001 (June), pp. 282–289.
- Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut (2020). *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. arXiv: [1909.11942](https://arxiv.org/abs/1909.11942) [cs.CL].
- Landis, J. Richard and Gary G. Koch (1977). "The Measurement of Observer Agreement for Categorical Data." In: *Biometrics* 33 (1), p. 159. ISSN: 0006341X. DOI: [10.2307/2529310](https://doi.org/10.2307/2529310).
- Le, Quoc and Tomas Mikolov (2014). "Distributed representations of sentences and documents." In: *31st International Conference on Machine Learning, ICML 2014 4*, pp. 2931–2939.
- LeCun, Yann, Y. Bengio, and Geoffrey Hinton (May 2015). "Deep Learning." In: *Nature* 521, pp. 436–44. DOI: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- Lee, Danielle H. and Peter Brusilovsky (2007). "Fighting information overflow with personalized comprehensive information access: A proactive job recommender." In: *3rd International Conference on Autonomic and Autonomous Systems, ICAS'07* (May 2014). DOI: [10.1109/CONIELECOMP.2007.76](https://doi.org/10.1109/CONIELECOMP.2007.76).
- Levesque, Hector J., Ernest Davis, and Leora Morgenstern (2012). "The Winograd Schema Challenge." In: *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*. KR'12. Rome, Italy: AAAI Press, pp. 552–561. ISBN: 9781577355601.
- Li, Nan, Bo Kang, and Tijn De Bie (2023). "SkillGPT: a RESTful API service for skill extraction and standardization using a Large Language Model." In: *arXiv preprint arXiv:2304.11060*.

- Li, Qian, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip Yu, and Lifang He (Apr. 2022). "A Survey on Text Classification: From Traditional to Deep Learning." In: *ACM Transactions on Intelligent Systems and Technology* 13, pp. 1–41. DOI: [10.1145/3495162](https://doi.org/10.1145/3495162).
- Linden, G., B. Smith, and J. York (Jan. 2003). "Amazon.com recommendations: item-to-item collaborative filtering." In: *IEEE Internet Computing* 7 (1), pp. 76–80. ISSN: 1089-7801. DOI: [10.1109/MIC.2003.1167344](https://doi.org/10.1109/MIC.2003.1167344). URL: <http://ieeexplore.ieee.org/document/1167344/>.
- Liu, Junhua, Yung Chuen Ng, Zitong Gui, Trisha Singhal, Lucienne T M Blessing, Kristin L Wood, and Kwan Hui Lim (2022). "Title2Vec: a contextual job title embedding for occupational named entity recognition and other applications." In: DOI: [10.1186/s40537-022-00649-5](https://doi.org/10.1186/s40537-022-00649-5). URL: <https://doi.org/10.1186/s40537-022-00649-5>.
- Liu, Junhua, Yung Chuen Ng, Kristin L. Wood, and Kwan Hui Lim (2020). "IPOD: A large-scale industrial and professional occupation dataset." In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, pp. 323–328. DOI: [10.1145/3406865.3418329](https://doi.org/10.1145/3406865.3418329).
- Luoma, Jouni and Sampo Pyysalo (2020). "Exploring Cross-sentence Contexts for Named Entity Recognition with BERT." In.
- Lyu, Wenjing and Jin Liu (2021). "Soft skills, hard skills: What matters most? Evidence from job postings." In: *Applied Energy* 300, p. 117307. ISSN: 0306-2619. DOI: <https://doi.org/10.1016/j.apenergy.2021.117307>.
- MacCartney, Bill and Christopher D. Manning (Aug. 2008). "Modeling Semantic Containment and Exclusion in Natural Language Inference." In: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Ed. by Donia Scott and Hans Uszkoreit. Manchester, UK: Coling 2008 Organizing Committee, pp. 521–528. URL: <https://aclanthology.org/C08-1066>.
- Maheshwari, Sumit, Sainani Abhishek, and Polepalli Krishna Reddy (2010). "An Approach to Extract Special Skills to Improve the Performance of Resume Selection." In: *Conference: Databases in Networked Information Systems, 6th International Workshop, DNIS 2010* (March 2010). ISSN: 03029743. DOI: [10.1007/978-3-642-12038-1](https://doi.org/10.1007/978-3-642-12038-1).
- Maheshwary, Saket and Hemant Misra (2018). "Matching Resumes to Jobs via Deep Siamese Network." In: pp. 87–88. DOI: [10.1145/3184558.3186942](https://doi.org/10.1145/3184558.3186942).

- Malherbe, Emmanuel, Mario Cataldi, and Andrea Ballatore (2015). "Bringing Order to the Job Market : Efficient Job Offer Categorization in E-Recruitment." In: (Sirip).
- McCullagh, P. and J.A. Nelder (1989). *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall. ISBN: 9780412317606.
- McInnes, Leland, John Healy, and James Melville (2020). "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." In: arXiv: [1802.03426 \[stat.ML\]](https://arxiv.org/abs/1802.03426).
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient estimation of word representations in vector space." In: *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pp. 1–12.
- Mohamed, Asmaa, Ali Eldesoky, and Hesham Ali (Feb. 2015). "Exploiting Semantic Annotations and Q-Learning for Constructing an Efficient Hierarchy/Graph Texts Organization." In: *TheScientificWorldJournal* 2015, p. 136172. DOI: [10.1155/2015/136172](https://doi.org/10.1155/2015/136172).
- Mooney, Raymond J. and Loriene Roy (2000). "Content-based book recommending using learning for text categorization." In: ACM Press, pp. 195–204. ISBN: 158113231X. DOI: [10.1145/336597.336662](https://doi.org/10.1145/336597.336662). URL: <http://portal.acm.org/citation.cfm?doid=336597.336662>.
- Nagarhalli, Tatwadarshi P., Vinod Vaze, and N. K. Rana (2021). "Impact of Machine Learning in Natural Language Processing: A Review." In: *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 1529–1534. DOI: [10.1109/ICICV50876.2021.9388380](https://doi.org/10.1109/ICICV50876.2021.9388380).
- Naik, Varsha, Purvang Patel, and Rajeswari Kannan (2023). "Legal Entity Extraction: An Experimental Study of NER Approach for Legal Documents." In: *International Journal of Advanced Computer Science and Applications* 14.3.
- Nigam, Amber, Aakash Roy, Hartaran Singh, and Aabhas Tonwer (2019). "Job recommendation through progression of job selection." In: *arXiv*, pp. 212–216.
- Nwankpa, Chigozie, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. arXiv: [1811.03378 \[cs.LG\]](https://arxiv.org/abs/1811.03378).

- Al-Otaibi, Shaha T. and Mourad Ykhlef (2012). "A survey of job recommender systems." In: *International Journal of the Physical Sciences* 7 (29). ISSN: 1992-1950. DOI: [10.5897/ijps12.482](https://doi.org/10.5897/ijps12.482).
- Özcan, Gözde and Sule Günduz Öguducu (2017). "Applying different classification techniques in reciprocal job recommender system for considering job candidate preferences." In: *2016 11th International Conference for Internet Technology and Secured Transactions, ICITST 2016*, pp. 235–240. DOI: [10.1109/ICITST.2016.7856703](https://doi.org/10.1109/ICITST.2016.7856703).
- Pacuk, Andrzej, Piotr Sankowski, Karol Węgrzycki, Adam Witkowski, and Piotr Wygocki (Sept. 2016). "RecSys Challenge 2016." In: *Proceedings of the Recommender Systems Challenge*. ACM. DOI: [10.1145/2987538.2987544](https://doi.org/10.1145/2987538.2987544).
- Parhi, Prateek, Ashish Pal, and Manuj Aggarwal (2017). "A survey of methods of collaborative filtering techniques." In: *Proceedings of the International Conference on Inventive Systems and Control, ICISC 2017*, pp. 1–7. DOI: [10.1109/ICISC.2017.8068603](https://doi.org/10.1109/ICISC.2017.8068603).
- Parikh, Ankur P., Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit (2016). *A Decomposable Attention Model for Natural Language Inference*. arXiv: [1606.01933 \[cs.CL\]](https://arxiv.org/abs/1606.01933).
- Peer, Eyal, Joachim Vosgerau, and Alessandro Acquisti (Dec. 2014). "Reputation as a sufficient condition for data quality on Amazon Mechanical Turk." In: *Behavior Research Methods* 46 (4), pp. 1023–1031. ISSN: 15543528. DOI: [10.3758/s13428-013-0434-y](https://doi.org/10.3758/s13428-013-0434-y).
- Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). *GloVe: Global Vectors for Word Representation*.
- Petre, A., C. Osoian, and M. Zaharie (2016). *Applicants' Perceptions On Online Recruitment*. English. Copyright - Copyright Babes Bolyai University, Faculty of Economics and Business Administration 2016.
- Piróg, Danuta and Adam Hibszer (2022). "Do employers really require experience? An analysis of online job adverts and the implications for HE policy." In: *Studies in Higher Education* 47.11, pp. 2138–2160. DOI: [10.1080/03075079.2021.2020747](https://doi.org/10.1080/03075079.2021.2020747).
- Poll, Glassdoor Harris (2020). *30+ HR and Recruiting Stats for 2020*. URL: <https://www.glassdoor.co.uk/employers/resources/40-hr-and-recruiting-stats-for-2020/>.
- Qin, Chuan, Hengshu Zhu, Tong Xu, Chen Zhu, Chao Ma, Enhong Chen, and Hui Xiong (Feb. 2020). "An Enhanced Neural Network Approach to Person-Job Fit in

- Talent Recruitment." In: *ACM Transactions on Information Systems* 38 (2). ISSN: 15582868. DOI: [10.1145/3376927](https://doi.org/10.1145/3376927).
- Quillian, Lincoln, Devah Pager, Ole Hexel, and Arnfinn H. Midtbøen (Oct. 2017). "Meta-analysis of field experiments shows no change in racial discrimination in hiring over time." In: *Proceedings of the National Academy of Sciences of the United States of America* 114 (41), pp. 10870–10875. ISSN: 10916490. DOI: [10.1073/pnas.1706255114](https://doi.org/10.1073/pnas.1706255114).
- Radford, Alec and Karthik Narasimhan (2018). "Improving Language Understanding by Generative Pre-Training." In: URL: <https://api.semanticscholar.org/CorpusID:49313245>.
- Rafter, Rachael, Keith Bradley, and Barry Smyth (2000). "Personalised Retrieval for Online Recruitment Services." In: *Proceedings of the 22nd Annual Colloquium on Information Retrieval (IRSG 2000)* (April 2000), pp. 151–163. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1619&rep=rep1&type=pdf>.
- Ramezani, Maryam, Lawrence Bergman, Rich Thompson, Robin Burke, and Bamshad Mobasher (2008). "Selecting and applying recommendation technology." In: *IUI-08 Workshop on Recommendation and Collaboration (ReColl2008)*, pp. 1–9. URL: <http://maya.cs.depaul.edu/~mobasher/papers/Recommender-technology-ReColl08.pdf>.
- Ratinov, Lev and Dan Roth (2009). "Design Challenges and Misconceptions in NER." In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)* (June), pp. 1–66. URL: http://www.usaidbest.org/docs/Burundi%5C_2013%5C_Report%5C_Final%5C_508.pdf.
- Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. URL: <http://arxiv.org/abs/1908.10084>.
- Reinhart, Alex (2018). "A Review of Self-Exciting Spatio-Temporal Point Processes and Their Applications." In: *Statistical Science* 33.3. DOI: [10.1214/17-sts629](https://doi.org/10.1214/17-sts629). URL: <https://doi.org/10.1214%2F17-sts629>.
- Ricci, Francesco, Lior Rokach, Shapira Bracha, and Paul B. Kantor (2010). *Recommender Systems Handbook*. 1st. Springer-Verlag. ISBN: 0387858199. DOI: [10.5555/1941884](https://doi.org/10.5555/1941884).
- Rodrigues, Filipe and Francisco Pereira (Sept. 2017). "Deep learning from crowds." In: (July). URL: <http://arxiv.org/abs/1709.01779>.

- Sabou, Marta, Kalina Bontcheva, Leon Derczynski, and Arno Scharl (2014). "Corpus annotation through crowdsourcing: Towards best practice guidelines." In: *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014* (2010), pp. 859–866.
- Sang, Erik F. Tjong Kim and Fien De Meulder (2003). "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition." In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147. URL: <https://aclanthology.org/W03-0419>.
- Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf (2020). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. arXiv: [1910.01108 \[cs.CL\]](https://arxiv.org/abs/1910.01108).
- Şapcı, Ali Osman Berk, Hasan Kemik, Reyyan Yeniterzi, and Oznur Tastan (June 2023). "Focusing on potential named entities during active label acquisition." In: *Natural Language Engineering*, pp. 1–23. ISSN: 1351-3249. DOI: [10.1017/S1351324923000165](https://doi.org/10.1017/S1351324923000165). URL: https://www.cambridge.org/core/product/identifier/S1351324923000165/type/journal_article.
- Saranya, A. and R. Subhashini (2023). "A systematic review of Explainable Artificial Intelligence models and applications: Recent developments and future trends." In: *Decision Analytics Journal* 7, p. 100230. ISSN: 2772-6622. DOI: <https://doi.org/10.1016/j.dajour.2023.100230>. URL: <https://www.sciencedirect.com/science/article/pii/S277266222300070X>.
- Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl (Aug. 2000). "Application of Dimensionality Reduction in Recommender System – A Case Study." In: Schmitt, Thomas, Philippe Caillou, and Michèle Sebag (Sept. 2016). "Matching Jobs and Resumes: a Deep Collaborative Filtering Task." In: *GCAI 2016 - 2nd Global Conference on Artificial Intelligence*. Vol. 41. GCAI 2016. 2nd Global Conference on Artificial Intelligence. Berlin, Germany. URL: <https://hal.inria.fr/hal-01378589>.
- Shanno, D F (1985). "An example of numerical nonconvergence of a variable-metric method." In: *Journal of optimization theory and applications* 46.1, pp. 87–94. ISSN: 0022-3239.
- Shi, Baoxu, Jaewon Yang, Feng Guo, and Qi He (Aug. 2020). "Saliency and Market-aware Skill Extraction for Job Targeting." In: Association for Computing Machinery, pp. 2871–2879. ISBN: 9781450379984. DOI: [10.1145/3394486.3403338](https://doi.org/10.1145/3394486.3403338).

- Shirky, Clay (2008). *It's Not Information Overload. It's Filter Failure*. URL: <https://youtu.be/LabqeJE0QyI>.
- Singhal, Amit (2001). "Modern Information Retrieval: A Brief Overview." In: *IEEE Data Engineering Bulletin* 24. ISSN: 1461-3557.
- Siting, Zheng, Hong Wenxing, Zhang Ning, and Yang Fan (July 2012). "Job recommender systems: A survey." In: IEEE, pp. 920–924. ISBN: 978-1-4673-0242-5. DOI: [10.1109/ICCSE.2012.6295216](https://doi.org/10.1109/ICCSE.2012.6295216).
- Smith, Ellery, Andreas Weiler, and Martin Braschler (2021). "Skill Extraction for Domain-Specific Text Retrieval in a Job-Matching Platform." In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Ed. by K. Selçuk Candan, Bogdan Ionescu, Lorraine Goeriot, Birger Larsen, Henning Müller, Alexis Joly, Maria Maistro, Florina Piroi, Guglielmo Faggioli, and Nicola Ferro. Cham: Springer International Publishing, pp. 116–128. ISBN: 978-3-030-85251-1.
- Sparck Jones, Karen (1988). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval." In: *Document Retrieval Systems*. GBR: Taylor Graham Publishing, pp. 132–142. ISBN: 0947568212.
- Storks, Shane, Qiaozhi Gao, and Joyce Y. Chai (2020). *Recent Advances in Natural Language Inference: A Survey of Benchmarks, Resources, and Approaches*. arXiv: [1904.01172](https://arxiv.org/abs/1904.01172) [cs.CL].
- Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer (Oct. 2003). "Feature-rich part-of-speech tagging with a cyclic dependency network." In: vol. 1. Association for Computational Linguistics, pp. 173–180. DOI: [10.3115/1073445.1073478](https://doi.org/10.3115/1073445.1073478).
- Tripathi, Pooja, Ruchi Agarwal, and Tanushi Vashishtha (2016). "Review of job recommender system using big data analytics." In: *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016*, pp. 3773–3777.
- Turney, Peter D. and Patrick Pantel (2010). "From frequency to meaning: Vector space models of semantics." In: *Journal of Artificial Intelligence Research* 37, pp. 141–188. ISSN: 10769757. DOI: [10.1613/jair.2934](https://doi.org/10.1613/jair.2934).
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In:

- Advances in Neural Information Processing Systems* 2017-Decem (Nips), pp. 5999–6009. ISSN: 10495258.
- Wall, Michael E., Andreas Rechtsteiner, and Luis M. Rocha (2003). *Singular Value Decomposition and Principal Component Analysis*. Kluwer: Norwell, MA, pp. 91–109.
- Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman (Nov. 2018). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, pp. 353–355. DOI: [10.18653/v1/W18-5446](https://doi.org/10.18653/v1/W18-5446). URL: <https://aclanthology.org/W18-5446>.
- Wang, Xinyu, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu (2021). “Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning.” In: URL: <https://github.com/Alibaba-NLP/CLNER..>
- Warstadt, Alex, Amanpreet Singh, and Samuel R Bowman (2018). “Neural Network Acceptability Judgments.” In: *arXiv preprint arXiv:1805.12471*.
- Wilson, D. Randall and Tony R. Martinez (2003). “The general inefficiency of batch training for gradient descent learning.” In: *Neural Networks* 16 (10), pp. 1429–1451. ISSN: 08936080. DOI: [10.1016/S0893-6080\(03\)00138-2](https://doi.org/10.1016/S0893-6080(03)00138-2).
- Xiao, Wenming, Xiao Xu, Kang Liang, Junkang Mao, and Jun Wang (2016). “Job Recommendation with Hawkes Process: An Effective Solution for RecSys Challenge 2016.” In: *Proceedings of the Recommender Systems Challenge*. RecSys Challenge '16. Boston, Massachusetts, USA: Association for Computing Machinery. ISBN: 9781450348010. DOI: [10.1145/2987538.2987543](https://doi.org/10.1145/2987538.2987543). URL: <https://doi.org/10.1145/2987538.2987543>.
- Yadav, Vikas and Steven Bethard (2019). “A Survey on Recent Advances in Named Entity Recognition from Deep Learning models.” In: arXiv: [1910.11470 \[cs.CL\]](https://arxiv.org/abs/1910.11470).
- Yang, Runqi, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen (2019). “Simple and Effective Text Matching with Richer Alignment Features.” In: *Association for Computational Linguistics (ACL)*.
- Yang, Shuo, Mohammed Korayem, Khalifeh AlJadda, Trey Grainger, and Sriraam Natarajan (2017). “Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational Learning

- approach." In: *Knowledge-Based Systems* 136, pp. 37–45. ISSN: 09507051. DOI: [10.1016/j.knosys.2017.08.017](https://doi.org/10.1016/j.knosys.2017.08.017).
- Yang, Tianshan, Pengyuan Li, and Xiaoliang Wang (Aug. 2020). "Convergence Analysis of an Improved BFGS Method and Its Application in the Muskingum Model." In: *Mathematical Problems in Engineering* 2020, pp. 1–9. DOI: [10.1155/2020/4519274](https://doi.org/10.1155/2020/4519274).
- Yu, Shuiyuan, Jin Cong, Junying Liang, and Haitao Liu (2016). "The distribution of information content in English sentences." In: arXiv: [1609.07681](https://arxiv.org/abs/1609.07681) [cs.CL].
- Yu, Tianyu, Chengyue Jiang, Chao Lou, Shen Huang, Xiaobin Wang, Wei Liu, Jiong Cai, Yangning Li, Yinghui Li, Kewei Tu, et al. (2023). "SeqGPT: An Out-of-the-box Large Language Model for Open Domain Sequence Understanding." In: *arXiv preprint arXiv:2308.10529*.
- Yuan, Jianbo, Walid Shalaby, Mohammed Korayem, David Lin, Khalifeh AlJadda, and Jiebo Luo (Nov. 2016). "Solving Cold-Start Problem in Large-scale Recommendation Engines: A Deep Learning Approach." In: *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pp. 1901–1910. DOI: [10.1109/BigData.2016.7840810](https://doi.org/10.1109/BigData.2016.7840810). URL: <http://arxiv.org/abs/1611.05480>.
- Yuret, Deniz and Henry Winston (Mar. 1994). "From Genetic Algorithms To Efficient Optimization." PhD thesis. Massachusetts Institute of Technology.
- Zhang, Mike, Rob van der Goot, and Barbara Plank (2023). *ESCOXLM-R: Multilingual Taxonomy-driven Pre-training for the Job Market Domain*. arXiv: [2305.12092](https://arxiv.org/abs/2305.12092) [cs.CL].
- Zhang, Mike, Kristian Nørgaard Jensen, Rob van der Goot, and Barbara Plank (2022a). "Skill extraction from job postings using weak supervision." In: *arXiv preprint arXiv:2209.08071*.
- Zhang, Mike, Kristian Nørgaard Jensen, Sif Dam Sonniks, and Barbara Plank (Apr. 2022b). "SkillSpan: Hard and Soft Skill Extraction from English Job Postings." In: URL: <http://arxiv.org/abs/2204.12811>.
- Zhang, Xiang and Yann LeCun (2016). *Text Understanding from Scratch*. arXiv: [1502.01710](https://arxiv.org/abs/1502.01710) [cs.LG].
- Zhang, Zihan, Meng Fang, Ling Chen, and Mohammad Reza Namazi Rad (July 2022). "Is Neural Topic Modelling Better than Clustering? An Empirical Study on Clustering with Contextual Embeddings for Topics." In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational

Linguistics, pp. 3886–3893. DOI: [10 . 18653 / v1 / 2022 . naacl - main . 285](https://doi.org/10.18653/v1/2022.naacl-main.285). URL: <https://aclanthology.org/2022.naacl-main.285>.

Zhelezniak, Vitalii, Aleksandar Savkov, April Shen, and Nils Y. Hammerla (2019). *Correlation Coefficients and Semantic Textual Similarity*. arXiv: [1905.07790](https://arxiv.org/abs/1905.07790) [cs.CL].

DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University of Sheffield's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Sheffield, United Kingdom, September 2023

Thomas AF Green