Automatic Detection of Visually Descriptive Language and Scene Boundaries in Narrative Text

Tarfah Alrashid

Department of Computer Science

University of Sheffield

This dissertation is submitted for the degree of

Doctor of Philosophy

August 2024

I would like to dedicate this thesis to my beloved father in heaven, who did not have the chance to continue this journey with me. I was always blessed with his endless love, support and encouragement. He always believed in me, supported me, had my back in every step of the way. I am grateful for the happy and joyful life I lived with you, for our little secrets, for every exciting conversation, and for all our adventures that I'm so glad we had the chance to have. May god bless your beautiful, kind, and loving soul. Rest in peace ...

Fig. 1 My father Talal Abdulaziz Alrashid, 1960-2022.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Tarfah Alrashid

August 2024

Acknowledgements

I would like to seize this moment to convey my heartfelt appreciation to my supervisor, Professor Robert Gaizauskas, for his consistent support throughout my Ph.D. journey. His patience, encouragement, and profound knowledge were instrumental in guiding me during the entire process of researching and writing this thesis. I'm truly honoured to be supervised by him, I learned a lot from him during these years.

I extend my sincere thanks to my beloved mother Fawziah for her prayers, encouragements, and for all the love. Her support has been my constant source of strength. I would also like to thank my sister Taif and my brothers Abdulaziz, Ahmad, and Mohammad for their love and support. Their presence in my life has been a constant source of encouragement.

My deepest gratitude goes to my best friend Hesah Aldihan. You are the greatest gift I gained from this journey, the family I chose. Your presence made the challenges, obstacles, and moments of grief much easier and bearable. I cannot imagine completing this journey without your love and support. Thank you for being there for me every step of the way.

I would also like to thank my long-time friend Ablah Alamri. Your continuous support, love, and kindness have been a blessing. Your words always had a way of calming me down. You are a friend for life, and I am incredibly grateful to have you by my side on this journey.

To Noura Alotaibi and Shaha Altammar, your constant check-ins, prayers, and unwavering support have meant the world to me. Your kind messages and heartfelt wishes have provided strength and comfort. I am deeply grateful to have you so close to my heart.

To my little friend Naif, oh dear, if you could only know what your smiles did for me every single day. You lightened up my world and gave me a reason to smile when everything

else was telling me not to. Thank you for just being you. I would also like to thank my dear friend and neighbour Wafa, for your kind heart, support and lovely smile.

I would also like to express my gratitude to my friends, Shurooq, Omaima, Ashwaq, Noura, Maha, Amal, Areej and Rehab. Thank you for your unwavering support, and for the countless moments of laughter and comfort. Your companionship has been invaluable, and I am so fortunate to have you all in my life.

I want to thank my colleagues Reem, Varvara, and Sara. Working with you in the same research group and lab has been a wonderful experience. Your support, and friendship have made our work environment enjoyable.

I would like to extend my thanks to my sponsor, the University of Jeddah, for providing me with this great opportunity.

Last but not least, I would like to thank myself for completing this PhD, for not quitting when things were hard, for the hard work, and for the late nights. I am proud of my determination and resilience, and for believing in myself even when the journey was challenging.

Abstract

The last ten years have seen an explosion of research interest in the cross-over area between computer vision and natural language processing, driven by interest in potential applications such as automatic image description and captioning, image generation from text and automatic text illustration. Such research integrating vision and language often tries to solve image based tasks, by relying on images aligned with texts. However, such aligned data is both noisy and limited in volume. The research reported here starts from the observation that there is a vast amount of visually descriptive language in text not aligned with images that could potentially be exploited in image-text related applications. To use such language requires us to be able to identify it and to organise it, which leads us to the two exploratory research questions pursed here. First, can we identify visually descriptive language in text? And, second, given that human activity in the world takes places in various types of settings, can we identify such settings in narrative descriptions and can we identify when these settings change?

We pursue these questions through two specific studies which address novel language processing tasks. In our first study, we focus on the task of automatically classifying text into three classes (visually descriptive, partially descriptive and not visually descriptive) based on the proposed definition of VDL given in Gaizauskas et al. (2015). We perform this task on two levels: sentence level and segment level We use several linguistic and statistical features both separately and in different combinations to observe which perform better in the classification tasks. Our findings show that sentence level classification can be performed at around 79% accuracy and segment level classification can be achieved at around 81% accuracy.

In our second study, we explore how scenes in narrative text can be automatically identified. We contribute with a small corpus of scene annotated narrative text (ScANT) which to the best of our knowledge, is the first publicly available corpus of English narrative text annotated with scene boundaries. In this study, we develop guidelines to manually annotate the dataset with scene boundary information following SceneML framework (Gaizauskas and Alrashid, 2019). We also develop automatic scene segmentation models using both feature engineering and deep learning approaches. In the feature engineering approach we test the extent to which VDL helps in automatic scene segmentation. Our results show this is a hard task, with our best performing model, a pre-trained language model, achieving just 58% balanced accuracy on the automatic scene segmentation task.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Context and Motivation

In recent years, the fields of computer vision and natural language processing (NLP) have undergone unprecedented growth, driven by advancements in artificial intelligence leading to the integration of these two fields. The collaborative efforts between computer vision and NLP communities have contributed to solving many visual language tasks. There has been a considerable amount of work done on *image captioning*; the task of generating natural language descriptions for a given image (Chen et al., 2015; Ghandi et al., 2023; Ming et al., 2022). Figure 1.1 shows an example of captions that have been automatically generated for the given photo by (Li et al., 2020). Other work has been done on *text-to-image generation* in which images are generated from textual descriptions. Figure 1.3 shows an example of an image generated for the text scene below it (Coyne and Sproat, 2001). Recent advancements in *text-to-image generation* have led to groundbreaking platforms of generative AI such as Stable Diffusion[1], DALL-E (Ramesh et al., 2021) and Midjourney (https://www.midjourney.com). Figure 1.4 shows an example of two images generated for the text scene below them using Stable Diffusion tool. *Visual question answering* has also been a trending topic in which systems are trained to understand and answer questions about images (Antol et al., 2015; Goyal et al., 2017; Malinowski et al., 2015; Wu et al., 2017).

[1] https://stablediffusionweb.com

Furthermore, *automatic text illustration*; the task of automatically finding a suitable image for a given segment of text has attracted attention in the research field. Joshi et al. (2006) built a system that automatically selects images from story texts.

Such research integrating vision and language generally tries to solve image based tasks, by relying on images aligned with texts (Chen et al., 2015; Mogadala et al., 2021). The problem is twofold: (1) Many captions associated with images do not just describe the image. For example, a caption for an image could be "First photo Taken with my Canon AE-1" as in Figure 1.2. (2) The volume of images with associated captions overall is not great. For example, the most commonly used datasets are Flickr8K dataset which has 8092 images and Flickr30k dataset which has 31783 images in which each image has a caption containing 5 sentences.

Our research focuses primarily on the second of these two questions. Is there some way of exploiting text *not* directly associated with images? We can find such text in narratives and drama, because we get descriptions of setting in fiction. We may get visual descriptions in other genres also, for example in biography. So we propose to investigate ways of utilising these other purely linguistic image-related resources to help in tasks relating to images. These tasks could include:

- annotating images with captions — when describing objects or events in images, understanding how these objects or events are typically described in non-illustrated text can provide valuable insights to suggest what attributes to include in captions. For instance, describing a car may involve attributes such as colour, size, make, and age, while describing a wedding may include details like the number of guests, the bride's dress, and the cake;

- aligning images and movies with text — non-illustrated text, particularly in narrative forms like novels, is often organised into scenes. Understanding how scenes are formed and transitioned in text is essential for precise alignment with images/movie clips. It helps systems identify suitable breakpoints where images or movies can be aligned;

- selecting or generating images to include in illustrations in text — understanding how text is divided into scenes helps in generating images that accurately reflect the narrative structure. Each scene has distinct visual elements, settings, or characters. Learning to extract them from text can be helpful to be captured in the generated images.

Angle 1

The research reported in this thesis has explored two angles. One angle is building on the notion of visually descriptive language to see whether there is a way of automatically identifying language which is visually descriptive in text. The challenge is that there is a lot of visually descriptive text in other forms of text besides image captions and we need to see how to identify and extract that and utilise it for vision tasks. In order to do so, we need a definition of "visual language". Here we propose to use the definition provided by (Gaizauskas et al., 2015): visually descriptive language (VDL) is language which asserts propositions whose truth can be confirmed through visual sense alone.

Angle 2

The other angle is to explore the idea that narrative scenes are divisions found in all narratives, which amongst other things have to do with physical setting. Usually there are descriptions that enable the reader to note scenes have changed. We conjecture that as a first step towards identifying portions of a narrative that describe the physical setting of a scene, it would be useful to automatically detect scene boundaries in the narrative. Automatic scene segmentation could be useful for purposes of illustration and alignment. It can aid in the process of aligning images with story segments. It can also help in aligning movie clips with their corresponding textual scene segments by segmenting scenes from running narrative texts associated with the movies.

This thesis proposes the idea of automatic segmentation of narrative text (stories) into scenes. In order to carry out this task, we first provide a definition of what a scene is. Then, we define guidelines for scene annotation that help in identifying scene change and scene boundaries. Automatic segmentation of text into scenes could help in building a text corpus of scenes that could be used by systems converting scenes into images. In addition, scenes

could be clustered and common visual elements identified for scenes of the same type, i.e. scenes which are clustered together.

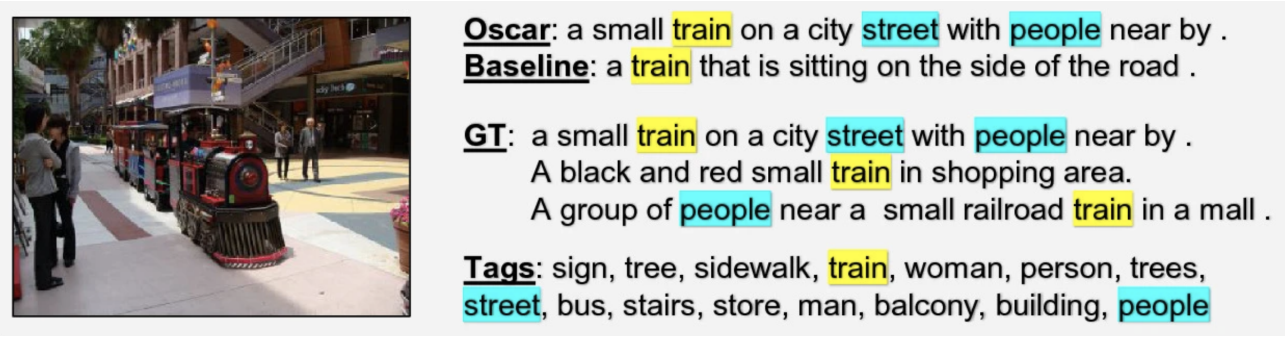


Fig. 1.1 An example of an image with captions that has been automatically generated - The captions are generated by Li et al. (2020)

Fig. 1.2 An example of an image with caption that is not descriptive - taken from https://www.baty.net/2013/canon-ae-1-program-as-nostalgia/

In our approach to automatic segmentation of narrative texts into scenes, we will investigate the use of visually descriptive language (VDL) as mentioned above. Using VDL is motivated by the idea that one indicator of scene change could be the change of physical setting, and description of setting is conveyed by VDL then the presence of VDL may suggest that this is where a scene change is taking place.

Fig. 1.3 An example of a scene generated from the following text (Coyne and Sproat, 2001) “John uses the crossbow. He rides the horse by the store. The store is under the large willow. The small allosaurus is in front of the horse. The dinosaur faces John. A gigantic teacup is in front of the store. The dinosaur is in front of the horse. The gigantic mushroom is in the teacup. The castle is to the right of the store".

1.2 Contributions

This thesis presents novel work in research related to visually descriptive language and automatic scene segmentation. The contributions are as follows:

- Extends work done on sentence level classification according to whether or not they contain visually descriptive language by adding additional features (see research paper 1) and presents the first attempt for the segment level classification task.
- Introduces guidelines based on sceneML to annotate scenes in narrative text.
- Introduces a publicly available dataset of annotated scenes (ScANT), the corpus and annotation guidelines are available at https://doi.org/10.15131/shef.data.21517908 and are made available under the CC By-NC 4.0 licence [2] (see research paper 4).

[2]https://creativecommons.org/licenses/by-nc/4.0/.

(a) “John uses the crossbow. He rides the horse by the store. The store is under the large willow. The small allosaurus is in front of the horse. The dinosaur faces John. A gigantic teacup is in front of the store. The dinosaur is in front of the horse. The gigantic mushroom is in the teacup. The castle is to the right of the store” – the text scene is taken from Coyne and Sproat (2001).

(b) Sara is working on her PhD Thesis anxious and worried till late at night on a cold rainy day. She is at the department lab in her desk drinking chamomile tea to calm her down.

Fig. 1.4 An example of two images generated by Stable Diffusion for the text below them.

- Introduces the novel task of automatic segmentation of narrative text into scenes based on ScANT dataset.
- Finally, it explores how useful VDL is in the task of automatic scene segmentation.

1.3 Overview of the Thesis

This thesis is structured as follows:

- **Chapter 2** is a literature survey of background information relevant to the project. In this chapter, section 2.1 covers the notion of visually descriptive language and gives details, both of the work of Gaizauskas et al. (2015) and other works on visual language. Section 2.3 discusses some definitions of scene in literature. Section 2.4 is about scene segmentation. Section 2.5 provides brief description of named entity recognition. Section 2.6 is about text segmentation; its approaches and evaluation.
- **Chapter 3** describes the research questions and research methodology and the data that has been collected and used for this research.
- **Chapter 4** presents the topic of Visually Descriptive Language. It describes the details of experiments done for the tasks of sentence level classification and segment level classification of VDL.
- **Chapter 5** introduces the idea of scene annotation in narrative text. It describes the framework established for annotating scenes in narrative text (sceneML). It also gives results of a pilot study on the annotation task and then gives a detailed explanation of the actual annotation task.
- **Chapter 6** presents work on the automatic scene segmentation task. It presents the automatic scene segmentation system that was built using three approaches (CRF based approach, sentence pair classification approach, and BERT based approach). It provides performance results of the three models and discusses these results.

- **Chapter 7** provides a comprehensive summary of the main ideas of this thesis, and presents a conclusion and future work.

1.4 Published Work

This section lists papers published in peer-reviewed international conferences as part of the research work carried out on this thesis.

1 The work described in Section 2.3.2 of chapter 2 has been published in the proceedings of the 15th Joint ACL - ISO Workshop on Interoperable Semantic Annotation (ISA-15) that was held in conjunction with The 13th International Conference on Computational Semantics (IWCS 2019) (Gaizauskas and Alrashid, 2019).

2 The work described in section 4.1 of chapter 4 has been published in the proceedings of the 15th Joint ACL - ISO Workshop on Interoperable Semantic Annotation (ISA-15) that was held in conjunction with The 13th International Conference on Computational Semantics (IWCS 2019) (Alrashid et al., 2019).

3 The work described in section 5.2 of chapter 5 has been published in the proceedings of the Fourth International Workshop on Narrative Extraction from Texts (Text2Story) that was held in conjunction with the 43rd European Conference on Information Retrieval (ECIR 2021) (Alrashid and Gaizauskas, 2021).

4 The work described in section 5.3 of chapter 5 has been published in the proceedings of the Sixth International Workshop on Narrative Extraction from Texts (Text2Story) held in conjunction with the 45th European Conference on Information Retrieval (ECIR 2021) (Alrashid and Gaizauskas, 2023).

Chapter 2

Background

The overall aims of this thesis are to investigate how visually descriptive language may be automatically extracted from text and how scenes may be automatically segmented in narrative text. In this chapter we review previous work on these two topics and also describe the principal natural language processing (NLP) techniques we use in the experimental work carried out in a subsequent chapter. In section 2.1, a description of Visually Descriptive Language (VDL) and examples of it are given. An overview of other approaches to visual language is also presented. In section 2.3, work related to scene segmentation is reviewed, and a brief summary of some definitions of scene in the literature are given. In section 2.4, a brief description of supervised classification methods is given. As discussed in the introduction, the task of segment level classification will use some of named entity recognition approaches, so the description of it is given in section 2.5. In addition, in section 2.6 a discussion about text segmentation, approaches and evaluation methods is provided. Finally, section 2.7 summarises the content discussed in this chapter.

2.1 Visually Descriptive Language

Gaizauskas et al. (2015) introduced the concept of visually descriptive language (VDL) in their research "Defining Visually Descriptive Language"[1]. VDL is proposed to be of

[1]This section has been adapted from my MSc (Alrashid, 2017)

potential benefit to research integrating vision and language (e.g. image captioning, story illustration, text/image retrieval). The authors try to answer the questions: (i) "how much text out there without associated images is 'visually descriptive' and thus potentially useful for such image-based tasks" (ii) "can these 'visually descriptive' text segments be identified automatically within documents which may consist of predominantly 'non-visual' text?". The goal of defining VDL is to be able to automatically mine VDL segments from a corpus of text that can help computational models of image based tasks. They provide a definition of VDL which relies on the notion of text segments whose truth can be confirmed by visual sense and this type of VDL need not be associated or aligned with an image, unlike the definition provided by (Dodge et al., 2012) (see below) identify noun phrases that are 'visually concrete' and often found in image captions. This distinction between the two definitions gives the following implications:

1. Visually Descriptive Language (VDL) can be extracted from any text, not limited to content linked with images. This broadens the scope of obtainable data for mining purposes.

2. text segments larger than noun phrases can be extracted and collected.

3. and finally, text can be collected from any genre, it is not restricted to the language commonly found in captions of images or news.

The Definition:

Gaizauskas et al. (2015) define visual descriptive language as follows:

> "A text segment is visually descriptive if it asserts one or more propositions about either (a) a specific scene or entity whose truth can be confirmed or disconfirmed through direct visual perception (e.g. (1)), or (b) a class of scenes or entities whose truth with respect to any instance of the class of scenes or entities can be confirmed or disconfirmed through direct visual perception (e.g. (2)). "
>
> 1. *"John carried the bowl of pasta across the kitchen and placed it on the counter"*.

2. *"Tigers have a pattern of dark vertical stripes on reddish-orange fur with a lighter underside"*.
3. *"Maria is thinking about what the future holds for her"*. (Not VDL)

The authors go on to clarify that by "direct visual perception" they mean that (1) for any given statement, it's truth can be confirmed (visually) by only looking at the image describing it without any intervention in the scene to gain more visual information. As an example mentioned by the authors, the statement 'John weighs 65kg' can only be confirmed if the scene contains john standing on a scale that shows 65kg reading on, otherwise this sentence is not VDL. (2) any inference from the scene that needs to be made to confirm or disconfirm the truth of the proposition in question would be made by any typical observer in the position of the narrator; and (3) in assessing the truth of a candidate VDL text segment the identity of any NE's is known to any observer putting themselves in the position of the narrator. Furthermore, in the context of the definition the statement "asserting a proposition" implies that any text segment deemed VDL should express an assertion that can be true or false.

A text segment here might consist of a phrase, a sentence, or a sequence of sentences. A sub-sentential structure such as a noun phrase can still be VDL even though it is does not assert a proposition directly because for any sentence in which it occurs, the truth of that sentence may entail the truth of another proposition which is implicit for the noun phrase to successfully refer. E.g, "The tall man standing by the bar is VP of Enron". In this sentence "The tall man standing by the bar" does not directly assert a proposition. But if the sentence "The tall man standing by the bar is VP of Enron" is true then the sentence "There is a tall man standing by the bar" must be true and this can be confirmed through direct visual perception.

Further notion of *impure visually descriptive language* (IVDL) is also defined, which is a text segment that contains a mixture of both VDL segments and non VDL segments (Gaizauskas et al., 2015). Here are examples of VDL sentences, sentences that are not VDL, and sentences classified as IVDL:

- VDL — *John carried the bowl of pasta across the kitchen and placed in on the counter*

- non VDL —*Maria is thinking about what the future holds for her*
- IVDL — *{the tall} , well-educated {man}*

Please note that this thesis does not involve the use of IVDL.

Based on the definition of VDL, a text may be annotated according to that definition in various ways, two possible ways are: sentence level annotation and segment level annotation. In sentence level annotation, each sentence is assigned **0** if it does contain any VDL, **1** if the whole sentence is VDL, and **2** if it has one or more segments that are considered VDL but the whole sentence is not a VDL. Guidelines for annotating challenging cases were provided by the authors, including:

- **Metaphors** (e.g. *he panted like {a big dog that has been running too long}*)
- **Words with mixed visual and aural or visual and experiential meanings** (e.g., *shuffle, shudder*)
- **Temporal adverbials of frequency** (e.g., *always, usually, sometimes*)
- **Multiple visual perspectives** (e.g., *{Billy climbed the tree wearing his back- pack}, {which contained his slingshot, some pebbles and a magnifying glass}*)
- **Intentional contexts** *(e.g., John believed that Mary was playing in the garden)*
- **Hypotheticals, modals, counterfactuals and subjunctives** (e.g., *If Jack sets the table then Will serves dinner*)
- **Location information** (e.g., VDL- *The Episcopal Church stood across the street*; not VDL — *The Episcopal Church was one block down Sussex Street*)
- **Statements of purpose** (e.g., *Billy climbed to the rooftop to shoot at crows*)
- **Imperative and interrogative sentences** (e.g., imperative - *Come out to the field and call us*, and interrogative - *How did {you escape from the beast}?*)

- **Participial phrases;** (e.g., *{Walking slowly across the ice, John} thought about his mother.*)
- **Dialogues** (e.g., *Dorothy said, "{Toto is running away}"*)

In the segment-level annotation task, the words constituting a VDL segment are annotated using swipe and click annotation tools.

Other approaches to visual language

There has been little work on this topic and we are only aware of two other relevant studies, Dodge et al. (2012) and Winn et al. (2016).

Dodge et al. (2012) who studied the topic of visual text, state, "A piece of text is visual (with respect to a corresponding image) if you can cut out a part of that image, paste it into any other image, and a third party could describe that cutout part in the same way". They believe that automatically mining visual text from corpora will be helpful in a number of tasks: to automatically mine image caption data that is used in training object detection systems, image retrieval systems, and in developing better language models to train models for image captioning systems. Images and their corresponding descriptions taken from SBU Captioned Photo Dataset (Ordonez et al., 2011) were used in Dodge et al.'s study as a dataset, the original dataset consists of 1 million images (selected from Flickr platform) accompanied by user-generated captions. Dodge et al.'s have two annotated datasets SMALL and LARGE, the SMALL contain 803 images with 2339 instances (each instance is composed of an image, an associated caption, and a highlighted noun phrase within that caption) where the LARGE dataset has 84k images with 40 instances. Each instance in the SMALL dataset is annotated by three annotators and annotated by one in the LARGE dataset. In this dataset, only noun phrases were annotated as *visual, not visual,* or *error* (for the cases where the noun phrase segmentation were incorrect). The difference between this work and Visually Descriptive Language is that the definition proposed by Dodge et al. relies on identifying noun phrases that are visually concrete, while VDL is not restricted to noun phrases; any segment of text can be visually confirmed. In addition, Dodge et al.'s dataset relies on texts that are associated

with images (i.e. image captions), which limits the amount of text that can be mined, whereas VDL can be mined from any text.

Winn et al. (2016) were interested in detecting visually relevant language to use in the task of fine-grained image classification. The question that they were trying to answer is "Can we automatically identify visually descriptive sentences relevant to a particular object from documents that may contain predominantly non-visual text?" Their work only focused on data regarding birds because that topic is one of the most well-studied domains in the task of fine-grained image classification. The researchers used the definition of VDL with some modifications (Gaizauskas et al., 2015) for defining the visually relevant sentences. Both definitions are similar in that they both detect visually confirmed segments of text rather than visually concrete segments. This is because their descriptions concentrated on a given class (birds) and not on an image. However, Winn et al. (2016) differs from Gaizauskas et al. (2015) in that their definition only considered visually descriptive segments with respect to the objects (birds); these sentences were defined to contain visually relevant sentences. A total of 1,150 Wikipedia articles on the topic of bird species were examined; this dataset was used to detect visually relevant sentences. The sentences were annotated as either **1** (VRL), which was assigned when the entire sentence was visually relevant, or **0** (non VRL), which was assigned when the sentence contained a bird object but was not visually descriptive.

2.2 Initial Work on Feature Extraction for Sentence Level VDL Classification

The following three subsections briefly explain the work that was done on feature extraction methods during my MSc dissertation (Alrashid, 2017). Three feature extraction methods were developed and explored; VerbNet, WordNet and tf-idf. The three feature extraction methods were tested using three different machine learning algorithms: Multinomial Bayes classifier (mnb), K Nearest Neighbour classifier (knn) and Support Vector Machine classifier (svm).

2.2.1 Approach 1: Using VerbNet

"VerbNet is a hierarchical domain-independent, broad-coverage verb lexicon with mappings to several widely used verb resources, including WordNet, Xtag, and FrameNet" (Schuler, 2005). VerbNet groups English verbs into classes based on Levin's classification (Levin, 1993), offering syntactic information. Levin's classification has been refined and new subclasses have been introduced. VerbNet has 237 first-level classes and 4526 verb senses (VerbNet, 2006). The NLTK interface to the VerbNet extended version was used in the implementation of this approach.

This approach utilises VerbNet to classify verbs. The Stanford Dependency Parser (Chen and Manning, 2014) is used to parse the sentence to be classified. From the resulting parse tree we first extract the main verb – it can be identified as the root of the sentence. Sometimes, the are no verbs in some sentences, or the parser may fail to find a verb, so a binary feature is added to the feature list (the input to the supervised learning algorithm) to identify the existence of a verb, 1 if a verb found in the sentence and 0 otherwise (Alrashid, 2017). This is an admittedly simply approach but we wanted to see how it would work.

Then, the algorithm searches VerbNet with the main verb to extract the verb class. VerbNet uses numbers to represents classes. The class numbers that have the potential to represent visual verbs are manually recorded in a list. To create lists that store visual verbs and not visual verbs, the VerbNet main classes were manually inspected and their ids were manually assigned to either the visual verbs list or the not visual list. To do that, we tried to answer the question *can the verb be represented in an image?* if yest then the verb id is stored in the visual verb list and in the not visual verb list otherwise. For example, the word class *eat-39* can be considered as a visual verb so its id *39* is saved in the visual verbs list. After the verb class is searched in VerbNet, whether the class number exists in the manually created list is then checked. If the class number is found, a binary visual verb feature is added with a value of 1; otherwise 0 is assigned to this feature (Alrashid, 2017).

2.2.2 Approach 2: Using WordNet

WordNet serves as a comprehensive lexical resource for English, categorising verbs, nouns, adjectives, and adverbs into distinct sets of synonyms known as "synsets". With 117,000 synsets, WordNet links between them through conceptual relations (University., 2010). A gloss is added for every synset that defines the word. Noun synsets have three types of relations between them: (1) hyperonomy (superclass), (2) hyponymy (subclass) and (3) meronomy (is a part of). The hierarchical structure of WordNet is formed by linking those synsets by these relations, and there is a root node (entity) for all synsets. This root node is called 'physical entity' or 'physical object' in the case of physical entities. Verb synsets also form a hierarchical structure but they do not share a root node. However, verbs in WordNet are organised into fifteen files based on semantic classes, each file representing a specific semantic domain, such as "verbs of bodily care and functions, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social interaction, and weather verbs" (Miller et al., 1990).

This approach is like Approach 1 but uses WordNet to extract verb classes, rather than VerbNet. First, it uses the Stanford parser to parse each sentence to obtain the main verb. Then it looks for the verb class in WordNet, and takes the first sysnet from the list of synsets that WordNet returns as synsets for a word are meant to be ordered by frequency of occurrence in a corpus. To do so, two strategies are explored. The first strategy searches the main verb root's hypernym and considers this the class name. Then, it assigns the class name as as a feature value. If there is no verb in the sentence or the class name is not found, then the feature is assigned a value of 0. So the feature takes as values either the string which is the synset name of main verbs' hypernym or 0. This approach was designed because there is no single node that all verbs share when going up the verb hierarchy.

In the second strategy, the names of the 15 WordNet database files that store verbs are treated as verb classes. Each class is then categorised manually into two groups: visual verbs and non-visual verbs. The visual verbs include verb classes such as *verb consumption, verb creation, verb body, verb motion, verb change, verb communication, verb competition,* and *verb contact*. On the other hand, the non-visual verbs encompass verb classes like *verb*

cognition, verb emotion, verb perception, verb possession, verb social, verb stative, and *verb weather*. This categorisation serves as a key feature extraction step, providing insights into the visual or non-visual nature of the verbs within the WordNet database files. The algorithm then determines which verb class the main verb occurs in. If that verb class is one that has been determined to be visual, a feature of value 1 is added to the feature list or 0, otherwise (Alrashid, 2017).

Subsequently, the arguments of the main verb in the parsed sentence are extracted from the parse – the subject and object of the sentence. These arguments are also checked to see if they are visual. WordNet is searched for the head word of the object and subject. The algorithm examines the hypernyms of these words, ascending the hypernym hierarchy. At each step, it checks whether the hypernym contains the word 'physical' until it either discovers the word 'physical' or reaches the root hypernym. If the term 'physical' is identified, indicating that the argument is visual, a feature with a value of 1 is included in the feature set. Conversely, a value of 0 is appended if no objects or subjects are detected in the sentence(Alrashid, 2017).

2.2.3 Approach 3 - Using Tf-idf

In this approach a document d_j from a document collection D is represented as a term weight vector, $\overrightarrow{d}_j = (w_{1j}, ..., w_{|T|j})$, T represents the set of terms or words that appeared at least once in the document D (Nigam et al., 2000). Term weights may be determined by treating a document as a bag of words, so that the term weights in the document vector are the number of occurrences of the term in the document. This is known as term frequency (tf) term weighting. " (Manning et al., 2008). Other methods to measure the term weight of a term t are (Salton and Buckley, 1988) (Manning et al., 2008):

- **Document frequency(*df*)**: is the number of documents in the document collection D that have term t.
- **Inverse document frequency (*idf*)**: is computed as $idf = log\frac{N}{df_t}$, N represents the number of documents in the document collection D and df_t is the document frequency of t is the term.

- **Tf-idf**: is computed by integrating the term frequency (*tf*) with inverse document frequency (*idf*) as in $tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t$.

In the tf-idf approach, the text is represented as a bag of words in which the order of the words is not considered (Manning et al., 2008).

In this feature extraction approach, each sentence in the given data is tokenised into words and processed using the term frequency-inverse document frequency (tf-idf) method. The TfidfVectorizer from scikit-learn is employed to handle this process. In addition, the inverse document frequency (IDF) values are calculated globally across the entire dataset, ensuring consistency in term weighting throughout all sentences. The document collection for calculating IDF includes both the training and test sets. Each sentence is transformed into a sparse document vector of term weightings, and these vectors collectively form a new list representing the features extracted from the text. This process, which includes the global calculation of IDF, is consistently applied to both the training and test sets. The resulting feature vectors, composed of term weightings, are then used to train a classifier.

2.3 Scene Recognition in Narrative Text

One of the aims of this thesis is to investigate how to automatically segment narrative text into scenes. To achieve this objective, we need a clear definition of what constitutes a scene, along with a set of guidelines and a framework for annotating data. This section provides an overview of definitions of scenes from different perspectives in section 2.3.1. Section 2.3.2 provides a brief description of sceneML, which is a framework to annotate scenes in narrative text. Furthermore, section 2.3.3 provides a brief review of previous work related to the tasks of scene annotation and scene segmentation.

2.3.1 Scenes

The definition of *scene* varies a lot depending on the context in which it is being mentioned and the area it is being used in. Some authors use the definition in models of computer

generative narrative perspective while others use it from the perspective of literary analysis. Here we give an overview of some definitions of a scene.

Scenes in Text

The first group of researchers used the definition of a scene in drama or narrative, both in computational and literary studies. Callaway and Lester (2002) in their work on “Narrative Prose Generation”, state that a segment of narrative text is considered to be a scene if it is contiguous in time, character, and place. Changes in any of these three elements constitute a scene change, i.e. when characters change their location or time, or when events change to focus on other characters. The authors proposed a model for narrative generation system that produces narratives. Kozima and Furugori (1994) in their paper “Segmenting Narrative Text into Coherent Scenes” proposed an algorithm to automatically segment a text into scenes. They defined the scene as a piece of text that has common characteristics as in a movie scene. They both have characters and objects in a certain situation with a specific time, place and background. From the point of view of drama, Polking (1990) defined scene as “a division within an act of a play, indicated by a change of locale, abrupt shift in time, or the entrance or exit of a major character".

In addition, Cutting (2014) stated that a scene is a chunk of narrative that is sometimes synonymous with the concept of event and often found in narrative arts. A scene has three parameters location, character, and time. He used that definition in a study of scene/event segmentation in movies, that investigates the shifts in location, characters, and time. Changes in these three elements create seven types of shifts. Dunne (2017) states that the physical life of a *setting* gives the truth about the characters in the story or in the scene. The setting is defined by the physical life, the time, and the physical objects of the setting, it is about knowing where and when the scene is taking place. Actions happening at night differ from these happening during the day, outdoor settings are different from indoor setting. Physical life helps the reader to imagine the image, and brings visual power to the scene.

Scenes in Images

On the other hand, the second group of researchers are interested in scene understating in the computer vision field. For example, Xiao et al. (2010) define a scene as any place or location a person can take actions in or a place a person can navigate in. Scenes are often related or associated with specific actions, e.g. "sleeping in a bedroom or reading in a library" and these actions are related to the space's visual features. The environment is defined by its size and shape, e.g. "a narrow corridor is for walking", by its material e.g. "snow, grass, wood", by its objects e.g. "table and chairs". Scenes in their project are categorised into a into a tree whose first level has three divisions: indoor, outdoor, and outdoor man-made. They built a database of 131,072 images called the SUN database, categorised into 908 scenes (Xiao et al., 2016) using the WordNet database. They select 70,000 concrete nouns and manually select any word completing the phrase "I am in a *place*" or "Let's go to the *place*". The definition of a *scene* here is similar to the definition of a *setting* in research about narrative that consists of only two elements: location and time.

2.3.2 SceneML

In the work presented below in Chapters 5 and 6 we make use of the SceneML framework for annotating scenes in narrative text introduced in (Gaizauskas and Alrashid, 2019). In this section we first give an overview of the overall SceneML annotation framework and then briefly describe each of the elements in this framework. A few changes were made to the annotation scheme when building the corpus. These changes are described in Chapter 5 and in Appendix A. Finally, we include an example of annotation scheme applied to a specific text. Unless otherwise indicated all quotes and details in the following are taken from (Gaizauskas and Alrashid, 2019).

The Annotation Framework

Taking into consideration the literature related to scene definition presented in section 2.3.1, SceneML defines a scene as a unit of a story in which the elements: time, location, and main

```
<scene id="s1" loc="pod" time="base">
    <character>I</character>
    <character>Skip</character>
    <character>Pockets</character>
</scene>
<sds id = "sds1" scene = "s1">
```

As we approached our destination, Skip started to issue instructions to the pod about approach vectors ... I was about to say something ... when Skip opened the door and

```
</sds>
<scene id="s2" loc="bubble shaped large room" time="base">
    <character>I</character>
    <character>Skip</character>
    <character>Pockets</character>
    <character>Trouble</character>
    <character>Methusaleh</character>
</scene>
<sds id = "sds2" scene_id="s2">
```

I stumbled out into a blaze of light and noise ... I was in a large room ... It was bubble shaped ... All around me in the bubble were bunnies ... " ... Right now, we have to take him to Methuselah." ... I was just about to ask if Skip was coming in with me, but the door had already opened and they were manoeuvring me through.

```
</sds>
<scene id="s3"  loc="cylindrical room"  time="base">
     <character>I</character>
    <character>Methusaleh</character>
</scene>
<sds  id = "sds3" scene_id="s3">
```

I found myself in a room that was cylindrical, like the pod only bigger ... There, in front of, or above, me (zero gravity is so confusing) was the oldest rabbit I had ever seen ... Methuselah then told me about how the Bunnies from the Future first came into being.

```
</sds>
<scene id="s4" loc="planet earth"  time="a long time ago">
    <character>humans</character>
    <character>bunnies</character>
</scene>
<sds  id = "sds4" scene_id="s4">
```

It was after the plants had turned nasty – a long time ago, even for this old bunny – and the story sounded more like a legend than real history ... The war was lost, but the bunnies never stopped searching for a way to achieve victory and to reclaim the planet.

```
</sds>
<sds id = "sds5" scene_id="s3">
```

“What happened to people? Where are they now?” I interrupted, rather rudely ...

```
<istlink id = "isl1" type = "sequence" scene1 = "s1" scene2 = "s2"/>
<istlink id = "isl2" type = "sequence" scene1 = "s2" scene2 = "s3"/>
<istlink id = "isl3" type = "analepsis" scene1 = "s3" scene2 = "s4"/>
```

Fig. 2.1 Example SceneML Annotations for Chapter 2 of Corcoran (2016)

characters are constant. Any change in these elements indicates a change of scene. A scene is an abstract discourse element, not a specific span of text. It consists of a location or setting, a time and characters who are involved in the events that take place in the scene. However, these elements that constitute a scene exist in the real or fictive world (*storyworld* as per narrative theory Schmid (2010)) that the narrative revolves around. "The scene itself is an abstraction away from the potentially infinitely complex detail of that real or fictive world, a cognitive construct that makes finite, focused narrative possible".

According to SceneML a scene in a textual narrative is realised through one or more *scene description segments* (SDS). An SDS is "a contiguous span of text that, possibly together with other SDSs, expresses a scene" (Gaizauskas and Alrashid, 2019). Typically a scene consists of one SDS, unless that scene involves other SDSs of other scenes such as scenes about past (memories) or future events. Furthermore, SDSs can be in the form of "spatially distinct locations that are topologically contained within or connected to the embedding SDS, or if the author is employing the narrative device of rotating between multiple concurrent scenes each of which is advancing a distinct storyline (a common technique in action movies)" (Gaizauskas and Alrashid, 2019).

It is important to define what each element of a scene (*characters*, *time*, *location*) is to facilitate the annotation process and to make it easier to detect scene changes if any of the elements changed. As shown in Section 2.3.3, scene changes have been studied by researchers such as (Cutting, 2014) and others. Studying which of the scene elements (*characters*, *time*, *location*) changes in a scene can be considered a contribution to previous studies. Here we propose to adopt the definitions, annotation standards, for the elements (*time*, *location*, and *spatial entities*) from Iso-TimeML (Pustejovsky et al., 2010) and Iso-Space (Pustejovsky et al., 2011). As for characters, the definition and annotation standards for named entities of type person from the ACE program will be adopted which has been recently used in the TAC 2018 entity discovery and linking task[2].

The previous standards will facilitate the annotation process for all mentions of times, locations/spatial entities and persons represented in the text. However, we are particularly

[2]See http://nlp.cs.rpi.edu/kbp/2018/ and https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-entities-guidelines-v5.6.6.pdf.

interested in the relations between the specific entities that construct a specific scene (i.e., the *characters*, *time*, *location* that construct a specific scene). For example, we want to know which SDS's are part of which scene and which characters, times and places are the characters, times and places are part of the scene, as opposed to merely being mentioned by characters in the course of a scene.

SceneML Elements

SceneML elements are categorised into two main categories:

1. **Entities:** entities consist of scenes, SDSs, characters, times, and locations.
2. **Relations:** "scene-scene narrative progression links", other relational links, e.g. the time of a scene, are represented as attributes of the scene, while character-scene links that indicate which characters participate in a scene, are represented as sub-elements of the scene entity to which they belong.

Scenes Scenes are the main element in SceneML. A scene has attributes: `id, time` and `location`. The triple of these attributes is unique for each scene. Each scene also includes a list of character sub-elements as there may be more than one character for each scene.

SDSs Scene description segments (SDSs) are the actual strings of text that compose the scene. One SDS cannot belong to more than one scene, but a scene can be composed of multiple SDSs. They include the following attributes: `id` and `scene_id` that is the id of the scene that the SDS belongs to, these attributes are unique to each SDS.

Time Time elements used here are the ones developed by ISO-TimeML, they include an `id` attribute and a text segment. Time could also include the time of the storyworld and it is represented by the attribute `base`.

Location Like time, we use locations element developed from ISO-Space. They also include an `id` attribute that is unique for each location and a text span.

Character Here we use the named entity type `person` from the ACE English Annotation Guidelines for Entities. The only difference here is that animals and non humans can be considered as characters. Characters have an `id` attribute that is uniquely assigned for each character and a text segment.

Narrative Progression Links Narrative progression links (`nplinks`) link between any two adjacent scenes. These links have different types depending on the type of transition between the two scenes .SceneML includes four types of nplinks:

- `sequence` links are assigned when one scene follows on from another, this could be because a character moves to another location; new characters arrive or old characters leave the setting; or just because time passes (e.g. two characters talk all afternoon and now it is nighttime)
- `analepsis` occurs when there is a flashback in the scene e.g. memory of the past
- `prolepsis` (or flashforward) occurs when then we are taken forward in time
- `concurrent links` are assigned between two scenes when the transition happens because there is another thread of the story happening at the same time so the transition take us to different characters and different place but the same time.

Example Annotation and Analysis

Figure 2.1 shows an example of annotated text from the children story "Bunnies From the Future" by Corcoran (2016). For readability reasons, the entities `time, location` and `character` are not annotated by their id, instead their text span is used as their attribute value (Gaizauskas and Alrashid, 2019).

The authors discuss two concerns that arise after annotating the story using XML: (1) how relations are represented in XML, and (2) how transition signals can be represented. For the first one, we have two choices; either the relations is represented a separate entity, or it is embedded as an attribute to either of the entities. In the example provided here, relations are represented as an attribute added to the entities e.g. `sceneid` is an attribute

added to the `sds`. Second, there are some sentences that signal a transition from one scene to another, such sentences could be considered to belong to the first scene, the second scene, both scenes or neither scenes. On the other hand, it could be treated as a separate element `scene_transition_signal`, just as ISO-TimeML and ISO-Space annotate signals (e.g. “in front of"), we prefer to represent them as separate signals (Gaizauskas and Alrashid, 2019).

Additional details about SceneML and scene annotation can be found in Chapter 5 and Appendix A.

2.3.3 Previous Work on Scene Annotation and Segmentation

The task of automatically segmenting narrative text into scenes has not been widely investigated by researchers. To approach this task from a supervised learning perspective requires one to first provide a clear definition of what a “scene” is as discussed in section 2.3.1, then propose annotator guidelines to manually annotate scenes in narrative text to train models that automatically segment narrative text into scenes. A few authors have addressed the task of automatically segmenting text into scenes but that text is not narrative, others have addressed the more general task of segmenting narrative text into topically coherent segments. In this section we consider the work of four authors relevant to this topic and finish with a summary of how we see their work relating to the work we report below in Chapter 6.

Kozima and Furugori (1994) introduce a method of segmenting narrative text into coherent scenes, suggesting that it could be used to resolve anaphora and ellipsis inside a scene. They define a scene as a set of continuous sentences that have coherence between them. A scene in a text is like a movie scene in that it describes objects, such as characters or properties, in a certain time, place and background. Scene segmentation is done using a Lexical Cohesion Profile (LCP), which was first proposed by Kozima (1993), to indicate boundaries of scenes in narrative text. It relies on the idea that coherent text is lexically cohesive (Halliday and Hasan, 2014; Morris and Hirst, 1991), so that local cohesiveness leads to local coherence. An LCP keeps a record of lexical cohesion between words inside a window, moving a window of a certain size (i.e. 50 words) word by word and measuring the

lexical cohesion between the words each time the window is moved. There is a relationship between LCP and change of scene; when a window is inside a scene, the LCP value is typically high and words tend to be lexically cohesive; however, when a window crosses a scene boundary, the LCP value decreases and the words vary lexically. Figure 2.2 shows how an LCP identifies the alternation of a scene by detecting the valleys that are the minimum points.

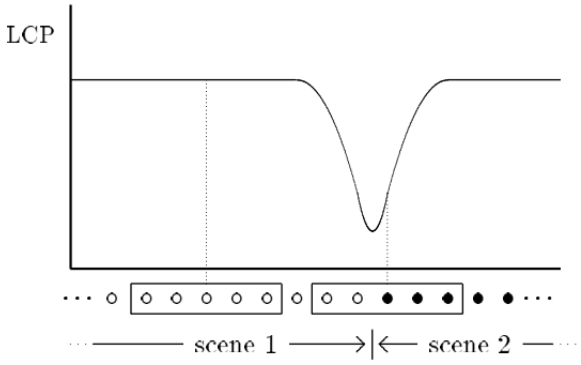


Fig. 2.2 LCP graph shows alternation of scene (Kozima and Furugori, 1994)

In order to compute LCP, the lexical cohesiveness between words is computed first by spreading activation across a semantic network paradigm (see Figure 2.4), which is an interval of [0,1]. A paradigm is a semantic network constructed systematically from a subset of an English dictionary called Glossem, that is used to analyse word meaning (Kozima and Furugori, 1993). The paradigm consists of 2,851 nodes and 295,914 links between them, nodes are constructed from the Glossem dictionary inputs. "Each node consists of a head-word, a word-class, an activity-value, and two sets of links: a referent and a referee". Lexical cohesiveness between two words w and w' is computed as follows:

$$\sigma(w,w') = s(w') \cdot a(P(w),w')$$

where $s(w') = \frac{-log(count\ of\ w'/size\ of\ corpus)}{-log(1/size\ of\ corpus)}$, $P(w)$ is the activated pattern that is produced in the paradigm and $a(P(w),w')$ is the activity value of node w' in $P(w)$. Then, text cohesion is estimated based on the cohesiveness $c(S)$ of the text S by determining the mutual cohesiveness

between the words in S , which is computed by:

$$c(S) = \sum_{w_i \in S} s(w_i) \cdot a(P(S), w_i)$$

where *P(S)* is the activated pattern produced after activating each word w_i in text S. Thus, LCP is computed by taking the record of cohesiveness $c(S_i)$ of a local text S_i at every position in the text. The Hamming window size for LCP that gives the best correlation with scene boundaries is 51 words. This window size is chosen after experimenting with 18 window sizes ranging from 11 to 121 words. However, this window size is only applicable to the text under experiment, and that size may change according to the nature and structure of text under experiment (e.g. average scene length). The authors stated that they have no effective way to adapt the window width for any text. The technique is evaluated by comparing it to human judgements (a simplified version of O.Henry's "Springtime a la Carte" was manually segmented by 16 annotator into scenes) by plotting a histogram of human judgement, the original paragraph boundaries and LCP having a window width of 51 words(see Figure 2.3). Unfortunately, there were no inter-annotator agreement results reported in this paper and no results comparing LCP with human judgements other than this Figure and two very short paragraphs.

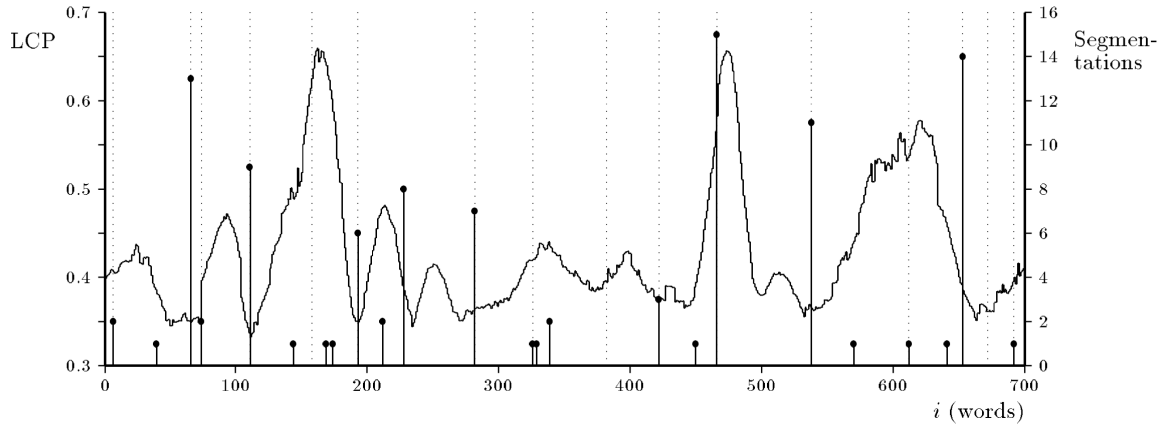
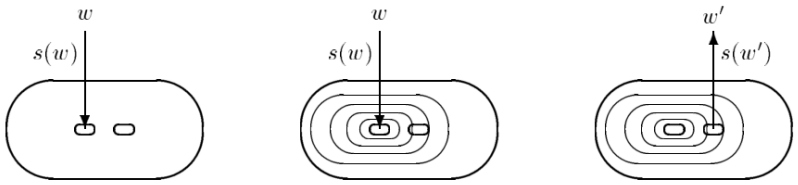


Fig. 2.3 Graph shows correlation between human judgements and LCP - dotted lines represent actual paragraph boundaries, and the solid bars represents human judgements histogram (Kozima and Furugori, 1994)

However, cohesiveness and cohesion of text do not always work well. Sometimes, the text cohesiveness measure *c(S)* does not work well on incoherent text that is lexically cohesive. In addition, in some cases some texts are coherent but are considered incohesive by *c(S)*.



(1) Start activating w. (2) Produce an activated pattern. (3) Observe activity of w'.

Fig. 2.4 Process of computing lexical cohesiveness (Kozima and Furugori, 1994)

Cutting (2014) addressed the problem of event segmentation. His study investigated the problem of narrative shifts in location, character and time. These shifts may align with the viewer's segmentation of the event (scenes) or may not agree with them. As discussed in section 2.3.1 "scene" is defined here as "a medium-size chunk found in all the narrative arts and often synonymous with the concept of an event". The dataset consists of 24 movies on drama, action, comedy, released from 1940 to 2010 (Cutting et al., 2012). They recruited eight viewers, to segment the movie shots into events, with each movie is segmented by three viewers. Authors then analysed the shifts between the segments based on changes in location, time frame. The study focused on the signals of new events in movies by indicating the discontinuity across boundaries of visual narrative shots. A scene has three parameters: location, character and time. Narrative shifts are divided into seven types, and depend on the presence or absence of changes in any of the scene parameters (location, character and time). The presence of a change is denoted by (1), and the absence of a change is denoted by (0); the seven types then become 111, 110, 101, 100, 011, 010 and 001. These seven types of narrative shifts are applied to movies; then, this approach is compared to other approaches in text that include investigating the effects of shifts in location, characters, time and other factors on readers. It was clear that the shifts in the three indices were uncorrelated with each other in the text. Regarding shifts in text, authors only interested in the effect of these shifts

on the speed of readers. There is no mention in the paper about methodology to segment the text and any annotated corpus of text.

The study of Kauchak and Chen (2005) investigated the segmentation of narrative documents. However, it did not tackle the problem of scene segmentation. Two narrative books were used here: “Biohazard” by Ken Alibek and “The Demon in the Freezer” by Richard Preston. The authors of these books segmented them into sections, and these sections were used as the true segment boundary locations. However, there was no discussion of what the criteria for segmentation were. “Biohazard” was split into three parts; two for experimenting and the third for testing, while "The Demon in the Freezer" was dedicated to testing only. “Biohazard” had 213 true and 5,858 possible boundaries, while “The Demon in the Freezer” had 119 true and 4,466 possible boundaries. The segmentation here was handled as a classification problem in which that segment boundaries could only fall on sentence boundaries, so each sentence boundary had to be classified as segment boundary or not using SVM classifier. The data is preprocessed first by tokenisation, stop word removal and stemming. Features used here for the classification include:

- **Word group**: These are the words that belong to the same parent node in the WordNet hierarchy. Thus, this feature is used to identify those word groups that are found at the boundaries.
- **Entity group**: This binary feature indicates the existence of a named entity (e.g. a person, city or disease) in a sentence.
- **Entity chains**: Entity chains are built in the same way as lexical chains; however, they include named entities. Two entities are related if they refer to the same entity; i.e. they belong to the same chain.
- **Full name**: By using a named entity extraction system, this feature indicates the existence of a full name in a sentence.
- **Pronoun**: This feature indicates whether a sentence has a pronoun and whether that pronoun is within the first five words of the beginning of the sentence.

- **Numbers**: This feature indicates the number patterns that often occur at the boundaries. It has been observed that numbers with four digits occur often, as they most commonly refer to a year.
- **Conversation**: The feature detects whether the sentence contains any conversations by detecting text enclosed within direct quotations.
- **Paragraph**: This feature specifies the start of a paragraph.

The authors used for evaluation, word error probability, sentence error probability (Beeferman et al., 1999) and windodiff (Pevzner and Hearst, 2002). Scoring 0.33 word error, 0.33 sent error, and 0.404 for window diff. Before testing the feature based approach, the authors first tried the traditional similarity based text segmentation approaches such as texttiling and PLSA. The authors state that the feature based approach outperforms the random classifier and the similarity based segmentation approach.

Using classification as an approach to the segmentation problem leads to the data losing its sequential nature. It might indicate segments with odd lengths (e.g. single sentences). Thus, it should include features that address the sequential nature of data.

Zehe et al. (2021) have proposed their own definition of *scene* in narrative text, have released a corpus of German language narrative texts annotated according to their scheme and have reported results on an automatic scene segmentation task. They follow the definition of a scene which they present in (Gius et al., 2019):

> A scene is a segment of the discours (presentation) of a narrative which presents a part of the histoire (connected events in the narrated world) such that (1) time is equal in discours and histoire, (2) place stays the same, (3) it centers around a particular action, and (4) the character constellation is equal

Following to this scheme, they annotated 15 dime novels written in German from various genres such as love, horror, and adventure, resulting in a total of 36K sentences and 550K tokens, with 971 scene segments and 34 non-scene segments (Figure 2.5 shows the distribution

Model (class)	Prec.	Rec.	F1	γ_o
TextTiling	0.02	0.97	0.04	0.01
TopicTiling	0.03	0.22	0.05	0.02
BERT (binary)	0.49	0.15	0.24	0.15
BERT (S-S)	0.43	0.13	0.2	0.15
BERT (S-NS)	0.85	0.24	0.38	
BERT (NS-S)	0.0	0.0	0.0	

Table 2.1 Scene segmentation models performance results.

of scenes and non-scenes by sentence length). The annotation process was carried out by two annotators (students specialised in computer science, digital humanities, and German studies, with previous experience in annotating for comparable tasks).

The authors conducted experiments to build scene segmentation models using two different approaches; (1) unsupervised approach, and (2) supervised approach. The unsupervised approach uses the TextTiling (Hearst, 1997) method implemented in NLTK (see also our description of TextTiling below in Section 2.6). They also implemented a model using TopicTiling (Riedl and Biemann, 2012) by training an LDA model. For the supervised approach, the authors fine tuned a pre-trained BERT model using FARM[3] (Fast Adapters for Robust Metrics) which is an open source library for natural language processing tasks. The model was fine tuned across ten epochs using a leave-one-out approach. Table 2.1 shows results of these approaches. You can see that the unsupervised approached performance was very poor (0.04, and 0.05 F1 score); the supervised approach was much better but it is still considered as not very good performance (0.24 F1 score).

Zehe et al. (2021) work differs from ours in several respects. First, their definition of scene states that a scene is a segment in a narrative in which the time, place and characters remain constant and which centres around one action. This contrasts with the definition in SceneML that does not take into account the actions in a scene and allows multiple actions to happen in one scene (see section 2.3.2 above and Gaizauskas and Alrashid (2019) for discussion around our choice of definition). Secondly, their scheme is less comprehensive – it does not define narrative progression links between scenes or scene transition segments,

[3]https://github.com/deepset-ai/FARM

and only distinguishes scene and non-scene segments. Thirdly, they follow a container principle (small places make up larger places) to detect a change in place, e.g. if the action of characters moves from a corridor to dining room that will not indicate a change in place as they are both part of a hotel, where as our definition counts these as two different places. Finally, they work on German texts while we are working on English texts. There are also differences in the algorithmic approaches we take to automatic scene segmentation. These are discussed further below in Chapter 6.

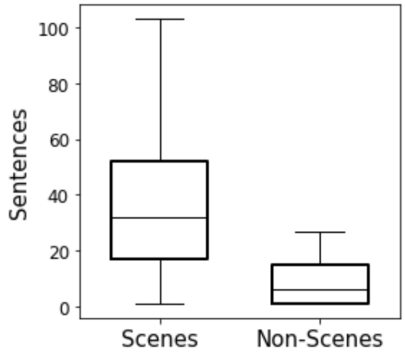


Fig. 2.5 Distribution of lengths (in sentences) in scene and non-scene segments of Zehe et al.'s annotated corpus of German dime novels.

In summary, the previous section provided an overview of four research works that are relevant to the topic of scene annotation and segmentation. The initial study introduced the LCP method for segmenting narrative segments into scenes. However, their implementation of scene segmentation had several limitations. Notably, the manually annotated dataset was of a small scale, consisting of only one short story. Additionally, the absence of detailed guidelines for the annotation process and the lack of inter-annotator agreement results were notable shortcomings in their approach. This makes it not clear if the method can be generalised to other narrative texts for the scene segmentation task (Kozima and Furugori, 1994). The second study introduced the task of feature-based segmentation of narrative text. The authors here treated the segmentation task as a classification task in which sentence boundaries are classified according to whether they hold a segment boundary or not. Similarity-based methods for segmentation were also carried out on the same dataset and results were compared to results of feature-based methods. The authors state that the

feature-based method outperform the similarity-based method and they also highlighted some of the issues with the feature based approach (Kauchak and Chen, 2005). The third study investigated the task of event segmentation (Cutting, 2014). Narrative shifts in location, character, and time were studied. Although the definition of event (scene) somewhat aliened with ours in that both definitions have three main elements: location, character, and time. Cutting (2014)'s study was purely about manual segmentation of movie shots. The fourth study tackled both the scene annotation and scene segmentation of narrative texts (Zehe et al., 2021). They annotated novels of different genres into scenes using their developed annotation guidelines. They tested both a supervised and unsupervised approach to scene segmentation and observed that a supervised approach was better in segmenting

None of the preceding authors' work has explicitly addressed the task of annotation of scenes in narrative text, which is the key task we address below in Chapter 6. The only other work on annotation of scenes in narrative texts of which we are aware is that of Zehe et al. (2021). Drawing implications from these studies, we realise the need to develop annotation guidelines to aid the process of annotating narrative scenes. We also find it advisable to explore a supervised approach for scene segmentation.

2.4 Classification

The task known as *classification* involves categorising a set of inputs into predefined classes. Classification is a *supervised learning* task in which models are trained using examples that are manually categorised (Marsland, 2009). Classification is a key technology underlying the work in Chapters 4 and 6. Section 2.4.1 discusses the general model whereby features are identified manually and feature extraction methods are written to extract the defined features which are then passed along with labels to a supervised learning algorithm. Section 2.4.2 gives examples of classification algorithms that work with human engineered/selected features. Section 2.4.3 explains neural networks and the deep learning approach to classification[4].

[4]Some part of this section has been adapted from my MSc (Alrashid, 2017)

2.4.1 Feature Engineering Approaches to Classification

Every input possesses distinct properties referred to as *features* including attributes such as shape, size, etc. Models classify inputs based on these features, either individually (e.g., by size or shape) or in combination (e.g., by both size and height). Additionally, the classification models that are heavily dependent on the properties or features of the training data may lead to weak generalization on unseen data (Bird et al., 2009). This challenge is commonly known as *overfitting* that often occurs when a large number of features are applied to a small training data. To overcome this problem, the most suitable features for classification should be selected from the feature list, as the choice significantly impacts the performance (Marsland, 2009).

Figure 2.6 represents the structure for a supervised learning process of feature extraction, training the classifier on the extracted features with the true labels, and finally producing predicted labels on the test data (unseen data).

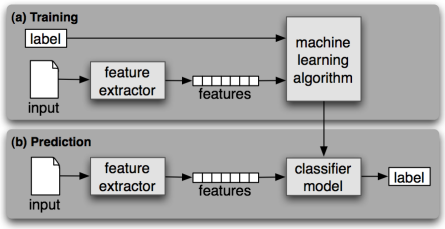


Fig. 2.6 A supervised classification task using feature engineering. Reproduced from http://www.nltk.org/book/ch06.html.

Another crucial step in developing a classifier model, is error analysis that is often performed after the feature set has been determined and selected and after training to refine the feature set and adjust hyper-parameters (see, e.g. NTLK). The dataset used to train the classifier is usually divided into two sets: a development set and a test set. The development set is in turn, further segmented into two subsets: a training set and a development-test set (Figure 2.7).

The primary function of the training set lies in training the classifier model. On the other hand, the development-test set serves a dual purpose – facilitating error analysis and

optimising hyper-parameters crucial to the learning process, thus avoiding overfitting. The test set is used then for the system evaluation.

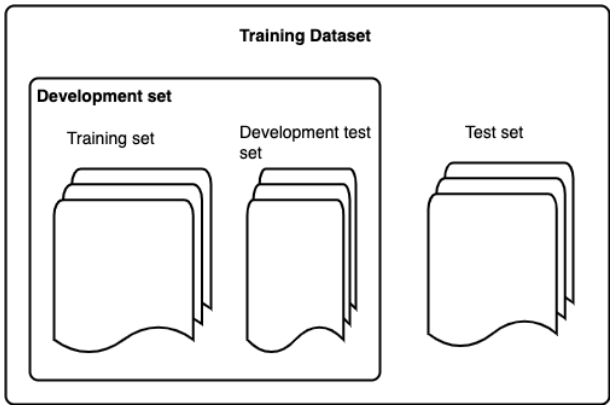


Fig. 2.7 Training Dataset.

2.4.2 Supervised classification Algorithms with Feature Engineering

These are examples of classification algorithms that work with human engineered/selected features.

Support Vector Machine – SVM

The support vector machine (SVM) is a non-probabilistic classifier, that according to Rogers and Girolami (2016) is most useful in cases where the number of features is more than the number of training instances. The SVM algorithm works by identifying a hyperplane (or a decision boundary) within an N dimensional space (where N refers to the number of features) that in turn can classify data points into distinct classes. There could be a lot of decision boundaries to choose from, and the goal relies on finding the boundary that has the maximum distance between the data points and the boundary (margins of the boundary). A linear decision boundary is defined by the following equation:

$$t_{\text{new}} = \text{sign}\left(\mathbf{w}^{\top}\mathbf{x}_{\text{new}} + b\right)$$

Data points which lie on one side of the boundary are put in the class $\mathbf{t}_{\text{new}} = 1$ and data points on the other side are classified as $\mathbf{t}_{\text{new}} = -1$. The labels of the classes are 1, and -1. The learning process is achieved by finding the values of w and b. In order to do that, parameters that identify the maximum margin need to be found.

"The margin is defined as the perpendicular distance from the decision boundary to the closest points on either side" (Rogers and Girolami, 2016). Figure 2.8 illustrates how the margin is calculated (the margin is represented by 2γ).

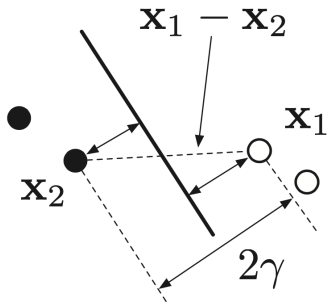


Fig. 2.8 Calculating the margin (γ). 2γ is equal to the vector $X_1 - X_2$ in the direction perpendicular to the boundary (Rogers and Girolami, 2016).

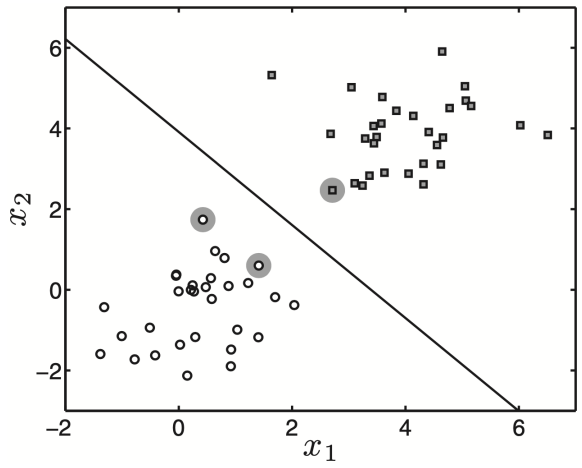


Fig. 2.9 Linear decision boundary with highlighted points that represent support vectors (Rogers and Girolami, 2016).

Support vectors are the data points closest to the maximum margins of the decision boundary (Figure 2.9). They are named support vectors because they help in determining the

decision boundary, as the goal for the decision boundary is to have the maximum margins, and in order to do that we need to find the closest points that determine the size of the margins.

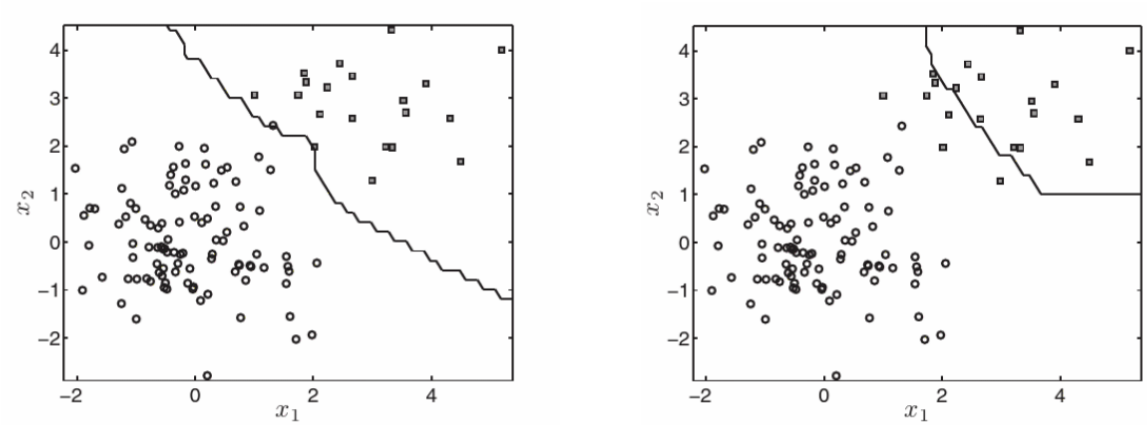
What happens if the data cannot be separated by a linear decision boundary? Instead of changing the algorithm, the data can be transformed to a new space that makes them separable by the linear boundary. Kernel functions are used to transform the data. These are the most commonly used kernel functions:

$$\begin{aligned} \text{linear } k(\mathbf{x}_n, \mathbf{x}_m) &= \mathbf{x}_n^\top \mathbf{x}_m \\ \text{Gaussian } k(\mathbf{x}_n, \mathbf{x}_m) &= \exp\left\{-\gamma(\mathbf{x}_n \mathbf{x}_m)^\top (\mathbf{x}_n - \mathbf{x}_m)\right\} \\ \text{polynomial } k(\mathbf{x}_n, \mathbf{x}_m) &= \left(1 + \mathbf{x}_n^\top \mathbf{x}_m\right)^\gamma \end{aligned}$$

k-Nearest Neighbours – kNN

k-Nearest Neighbours (kNN) is another non-parametric learning algorithm (Vemuri, 2020). Unlike other machine learning algorithms that don't make use of the training data after building the models, KNN keeps a memory of all the training data. So, whenever the model is given a data point to classify, it first searches the previously saved data and finds the k training examples closest to the given data point. Then the model returns the majority class (the most frequent label of the k closest training examples). In order to find the closest examples from the training data we need to find the distance from them to the data point that needs to be classified, there are multiple functions to measure the distance that can be used:

- cosine similarity
- Euclidean distance
- Hamming distance
- Mahalanobis distance
- Chebychev distance



(a) k = 5. (b) k = 39.

Fig. 2.10 Decision boundary comparison with different k values (Rogers and Girolami, 2016).

One problem that arises with KNN is that tie may result when we try to find the majority label of the k nearest points. To solve this problem a label can be chosen. Another more effective solution is to give a weight for each label based on the distance of the data point and according to the distance function chosen (Rogers and Girolami, 2016).

To chose k, a number needs to be chosen that is not big nor too small. This is because with a small k, a lot of noise can appear with the data and with too big a k this can lead to always assigning instances with one class if the number of instances in one class is significantly less the chosen number of k. Figure 2.10 shows the decision boundary with different values of k , one reasonable value with k=5 and the other too big value k=39. The dataset here has 50 data pointw with class 0 and 20 with class 1, you can see the decision boundary with k=5 is more reasonable.

Another way to choose k is by using cross validation. k can be chosen by picking the value that maximises classification accuracy on the training data.

Multinomial Naive Bayes – MNB

Multinomial naive Bayes (MNB) is a probabilistic classifier. The word *Bayes* is in its name as it is a Bayesian classifier. It is also referred to as *naive* it simplifies the assumptions about feature interactions. Our account of MNB in the following is based on that in (Jurafsky and Martin, 2014).

Figure 2.11 shows the idea behind the classifier. It represents a document as a bag of words where the order of the words is not important nor what sentence it appeared in. It just keeps score of how many times a word appeared in the whole document. So, in the example in Figure 2.11, we do not represent the word order in the phrases “I love this movie” and “I would recommend it”, we just say that the word *I* appeared 5 times and the word *it* 6 times and so on.

Since Naive Bayes is a probabilistic classifier, so the classifier needs to return the class $\hat{c}$ that has the maximum posterior probability of all classes $c \in C$ in a given document d.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

The work of Bayes (1763) introduces the idea of Bayesian inference, and Mosteller et al. (1964) were the first to apply it to text. Bayes rule can help in breaking down a probability into three probabilities to make it simpler, it is given in the following equation:

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)}$$

Bayes rule can be applied to the previous equation to transform it into the following:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

And by further simplifying this equation we get:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d) = \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

where $P(d|c)$ is the likelihood and $P(d \mid c)P(c)$ is the prior probability. We can represent a document d as a set of features $f_1, f_2, ..., f_n$. Naive Bayes assumes that the probabilities $P(f_i \mid c)$ are independent and hence can be multiplied. This gives us the final equation:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f \mid c)$$

In the case of text classification we can take as features the set of words that occur in the document to be classified. And to estimate the probability of particular text belonging to a specific class we can consider the words that occur at each position in the text. And in order to do that, the previous equation will be as the following:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in position} P(w_i \mid c)$$

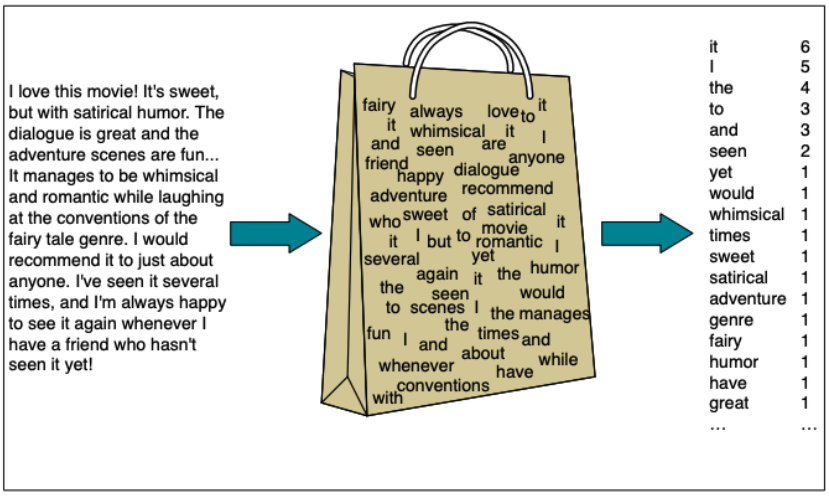


Fig. 2.11 Applying the idea of MNB classifier to a movie review (Jurafsky and Martin, 2014).

Conditional Random Field – CRF

The conditional random field (CRF) is a discriminative sequence labelling model that is based on log linear models. Here we describe the linear chain CRF widely used in language processing. Our account closely follows that of Jurafsky and Martin (2023)

Assume we have a sequence of words $X = x_1, x_2, ..., x_n$, the goal is to get the sequence of labels $Y = y_1, y_2, ..., y_n$ corresponding to these words. To do that, a CRF model calculates the posterior probability $P(Y \mid X)$, by training the model to be able to discriminate among various label sequences:

$$\hat{Y} = \underset{Y \in y}{\operatorname{argmax}} P(Y \mid X)$$

CRF computes a probability for every possible label sequence Y, given the word sequence X. To do that, it calculates log-linear functions at each time step based on a set of features. These features are then combined and normalised to generate a single overall probability for the entire label sequence. Each feature function F maps the input sequence X and the output sequence (labels) Y, to a feature vector. If we assume there are K features where feature F_k has weight w_k, then the posterior probability of label sequence Y given input X is computed by:

$$p(Y \mid X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)$$
$$Z(X) = \sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k\left(X,Y'\right)\right)$$

where $F_k(X,Y)$ is referred to as a global feature function. This is because they represent characteristics of the entire input sequence X and the output sequence Y. In a linear chain CRF these global features are calculated by breaking them down into a sum of local features for each position i in the output sequence Y according to the following equation:

$$F_k(X,Y) = \sum_{i=1}^{n} f_k\left(y_{i-1}, y_i, X, i\right)$$

Like the other supervised algorithms described above defining what the features are remains a task for the human language engineer. In work below in Chapter 4 and Chapter 6 the scikit learn implementation of CRF is used.

2.4.3 Representation Learning: NNs and Deep Learning

The first neural network architecture was invented by McCulloch and Pitts in 1943 in their paper “A logical calculus of the ideas immanent in nervous activity” (McCulloch and Pitts, 1943). They proposed a model called artificial neurons that has multiple binary inputs (on/off) and one output.The output neuron becomes active if one or or more of the inputs are active. They established that a network built of these artificial neurons could compute logical propositions.

The *perceptron* is an example of a simple network architecture that can deal with numbers as inputs and outputs, first proposed by by Frnak Rosenblaatt (Rosenblatt, 1958). The artificial neurons here are a bit different and are called *threshold logic units* (TLU). TLUs associate a weight w_i with each input *xi* and then sum these weighed inputs to get an output value *z*: $z = w_1x_1 + w_2x_2 + ... + w_nx_n = x^T w$. After that a step function is applied to this sum $h_w(x) = step(z)$. The Heaviside step function is mostly used in perceptrons. The perceptron consists of a single fully connected layer (dense layer) of TLUs; fully connected layer means that each TLS is connected to all inputs in the previous layer. The input layer consists of input neurons and its main task is just passing the input to the next layer (no computations are made here). And finally, a bias neuron is added to the input ($x_0 = 1$) that always outputs 1. So, the output of a fully connected layer is computed by:

$$h_{W,b}(X) = \phi(XW + b)$$

where, X is the input matrix, W is the weight matrix (the weight of bias neuron is computed here), b is the bias vector, and ϕ is the activation function, in our case here the step function. A diagram of the architecture is given in Figure 2.12) (Géron, 2022).

The perceptron training algorithm is inspired by Hebb's rule, "cells that fire together wire together", that is a connection between two neurons gets stronger when they fire at the same time. The learning or perceptron is then given by the following equation:

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta \left(y_j - \hat{y}_j\right) x_i$$

Where $w_{i,j}$ is the weight of the connection between the input neuron i and the output neuron j, x_i is the i^{th} input of the current instance being trained, $\hat{y}_j$ is the j^{th} output of the current instance being trained, and η is the learning rate.

Combining multiple Perceptrons yields in a multilayer perceptron (MLP). MLP consists of one input layer, one or more layers of TLUs (hidden layers), and an output layer (the final layer of TLUs). All layers have a bias neuron, except of the output layer (Figure 2.13). When a network has multiple layers (more than two) then it is called deep neural network.

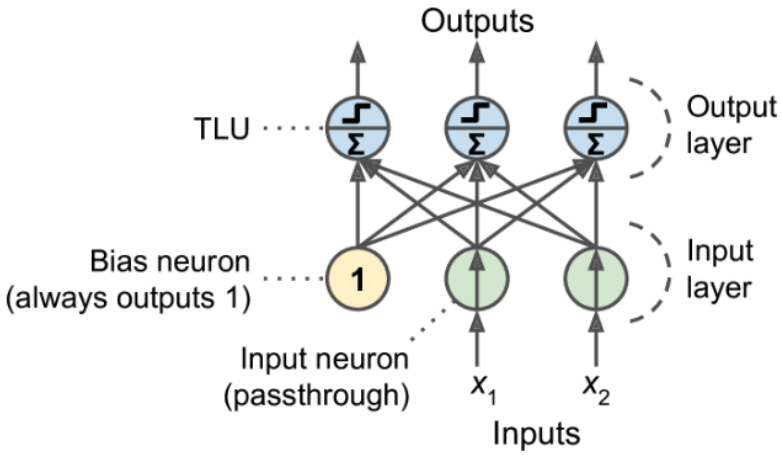


Fig. 2.12 Architecture of a perceptron with two input neurons, one bias neuron and an output layer of three output neurons (Géron, 2022).

Backpropagation algorithm made training MLP easier, it is a Gradient descent that is computed automatically in two passes over the network (a forward and a backward pass). For every training instance, the backpropagation algorithm first begins by making a prediction (forward pass) and assessing the error. Then, it traverses each layer in reverse, gauging the contribution to the error from each connection (reverse pass). Finally, it adjusts the connection weights to minimise the error through a Gradient Descent step.

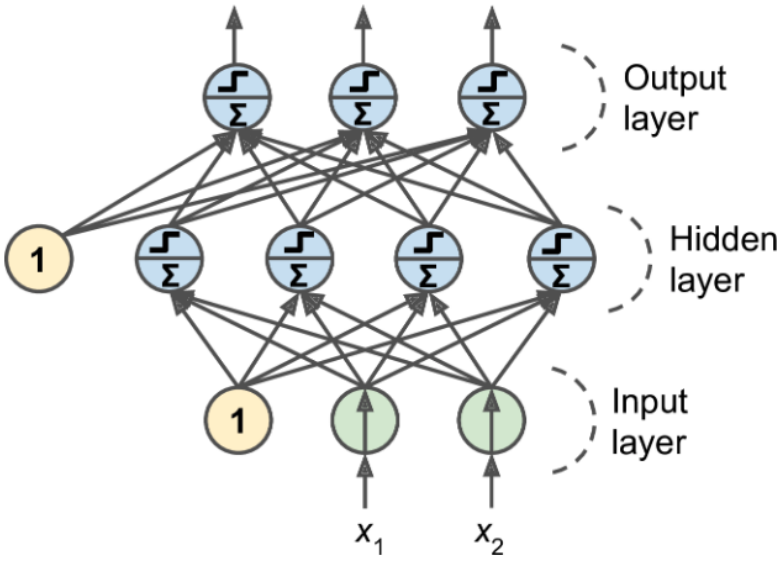


Fig. 2.13 Architecture of a multilayer perceptron (Géron, 2022).

Transformers

To understand the idea behind transformers, we first need to understand the **attention mechanism** as it plays a major role in the transformer architecture. The attention mechanism lets the encoder focus on a specific word (e.g. the word being translated in a given sentence). For example, for a translation task, when the decoder wants to output the word *lait*, it will only focus on the word *milk* sent from the encoder. This idea was introduced by Bahdanau et al. (2014). In this mechanism, all the encoder output is sent to the decoder (not just the *last* hidden state, which is the case in simpler RNN-based encoder-decoder models). At each time step, a weighted sum is computed and the decoder decides which word to focus the attention on this time step based on this computed weight. The weights are computed by an alignment model (or attention layer). This attention layer consists of a time distributed layer (dense layer) and a softmax layer. The dense layer takes the output from encoder and the hidden state and foe each input it measures how well it is aligned with the decoder's previous hidden state (this is considered as the weight for the input). The softmax layer then ensures that all weights add up to 1. This is the first attention mechanism used and it is called *concatenative attention*, after that several attention mechanisms appeared such as *multiplicative attention*.

The **Transformer** is an architecture created by a team of Google (Vaswani et al., 2017), they introduces the architecture in their paper "Attention Is All You Need". Figure 2.14 shows the model architecture. As you can see the architecture consists of encoder (on the left), and a decoder (on the right).

The encoder has a stack of N identical layers (in the paper N = 6), each layer consist of two sub layers; a multi-head attention layer and a position-wise feed forward layer. The input to the encoder is a collection of sentences, with each sentence being presented as a sequence of word IDs. Each word is encoded into a 512 dimensional representation. The purpose of the encoder is to systematically convert the input, which is the word representations of an English sentence, in such a way that each word's representation accurately reflects the word's meaning within the context of the sentence.

The decoder also has a stack of N identical layers (also 6), and apart from the two sub layers in the encoder, it also has a third sub layer. This additional sub layer applies multi-head

attention on the output taken from the encoder. The self attention sub layer here is modified so that it prevents positions from attending positions after them. The decoder takes as input the target sentence presented as a sequence of IDs of the its words. These sequences are shifted by one to the right (a start token is inserted at the beginning of the sequence). The decoder's task is to systematically convert the word representations within the translated sentence into word representations for the next word in the translation.

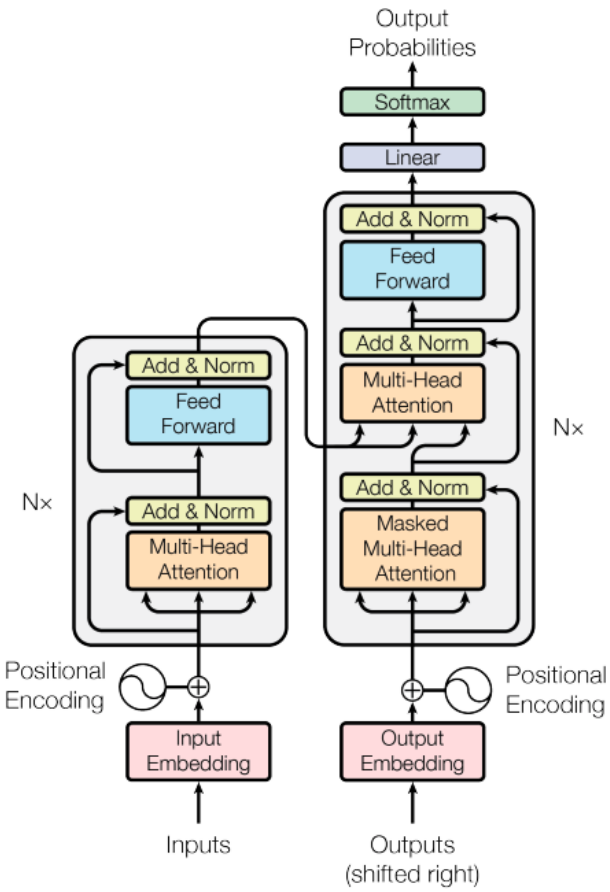


Fig. 2.14 Transformer Architecture (Vaswani et al., 2017).

The multi-head attention layer in the encoder enhances each word representation by focusing (paying attention to) on all the other words in the same sentence. This process is capturing its exact meaning within the context of the sentence. The masked multi-head attention layer in the decoder, has a similar task but when processing a word it does not look at the words after the current word being processed in the sentence, it only look (pay

attention to) at the words before. The upper multi-head attention layer in the decoder is part responsible in paying attention to the words in the original sentence being translated. The **positional encodings** in the figure 2.14 are dense vectors similar to word embeddings that have positional information about the position of each word. As all layers in the architecture do not consider word positions, and the order of words is important to language understanding, positional encodings are used to provide this information to the transformer. An **attention** function is a process that takes a query and a collection of key-value pairs and produces a result. All of these elements, the query, keys, values, and result, are represented as vectors. The result is determined by calculating a weighted sum of the values, with each value's weight determined by how well the query matches the corresponding key.

2.5 Named Entity Recognition

Named entity (NE) recognition is the task of recognising mentions of a set of predetermined named entity types in a given source text. The task involves determining the *extent*, i.e. the textual span, of each named entity mention and identifying the *type* of the mention. Entities to be extracted belong to different types (Jurafsky and Martin, 2014):

- **Individuals:** e.g. persons, organisations, places, books, films.
- **Kinds:** e.g. chemical compounds, disease, proteins.
- **Times:** e.g. dates, times of day.
- **Measures:** e.g. prices, sizes, quantities.

When a named entity in a text is mentioned multiple times, it is often referred to using different strings. These strings are said to corefer and this phenomenon is known as *coreference*. The coreference task is to link all the strings that refer to the same named entity. This may be treated as a separate task to recognising named entities or may be treated as part of the same task (Gaizauskas, 2017).

Named entity recognition is used as a preprocessing step in information extraction, information retrieval and other text processing tasks. (Jurafsky and Martin, 2014; Mikheev

et al., 1999; Tjong Kim Sang and De Meulder, 2003). Named entity recognition is a useful initial step for more complex information extraction processes such as event extraction and relation extraction.

NER is used later in the thesis as feature for supervised classification models.

2.5.1 Named Entity Recognition Approaches

Rule Based Approaches Rule based systems depend on the use of lexicons such as gazetteers which contain lists of proper names for each of the named entity types (person, organisation, company designators, person titles, locations, etc.) (Wakao et al., 1996). For example, the approach of (Wakao et al., 1996) has three steps: lexical preprocessing, NE parsing, and discourse interpretation. The lexical processing step starts by splitting sentences and tokenizing text, and then part of speech tagging each token of the tokenized text. After that, the system searches for a matching name in the gazetteers to get the named entity either for a phrase or a word. Then, *trigger words* are identified to help to recognise named entities. Trigger words are words that appear with named entity phrases that indicate their named entity type (e.g. in *Wing and Prayer Airlines* the word *Airlines* indicates that Wing and Prayer is an organisation). After that, NE parsing applies a set of hand-produced rules to classify unclassified proper names, e.g. "a sequence of upper case initial words followed by a company designator is an organisation" (e.g. *Delta Electronics Ltd*). The last step, discourse interpretation, consists of two tasks: the first is to do coreference resolution which looks for unclassified proper names that refer to already known named entities, then it assigns the class of the known entities to the unclassified ones. The second task involves inferring NE types from the argument types of some relations.

Sequence Labelling Approaches Sequence labelling is also used to get the labels for the named entities. Each word is tagged by considering the types and boundaries of named entities, a sequence classifier is trained to predict the label of words that indicates presence of named entity type (Jurafsky and Martin, 2014). **IOB** tagging - which stands for (**I**nside,

Outside, **B**eginning) - is used to tag the beginning of a named entity, inside the named entity, and tokens outside the named entity. Figure 2.15 shows an example of IOB tagging output.

Words	IOB Label
American	B-ORG
Airlines	I-ORG
,	O
a	O
unit	O
of	O
AMR	B-ORG
Corp.	I-ORG
,	O
immediately	O
matched	O
the	O
move	O
,	O
spokesman	O
Tim	B-PER
Wagner	I-PER
said	O
.	O

Fig. 2.15 IOB tagging example - (Jurafsky and Martin, 2014)

Feature based approach: In order to train IOB sequence labelling to predict tokens types, tokens must be represented as features that could be about the token itself or could include features about the words around it. Figure 2.16 shows possible features that are used to train sequence labelling classifier.

Deep learning based approach: Named entity recognition using **IOB** tagging can be implemented by fine tuning transformer based models (e.g. BERT). After the sentence is tokenised, each token is given a tag by passing the output vector of the token to the classifier. The classifier is a feedforward layer followed by a softmax, and it learns the weights for the possible tags of tokens. This gives a probability for each possible tag for the given token. Argmax tag of the token is then chosen, and this is used to produce the output tag sequence. Alternatively, a CRF layer could be added to produce the output tag sequence.

2.6 Text Segmentation

Text segmentation is the process of breaking down text into meaningful units. Many researchers have tried to automate this process in the field of Natural Language Processing.

identity of w_i, identity of neighboring words
embeddings for w_i, embeddings for neighboring words
part of speech of w_i, part of speech of neighboring words
base-phrase syntactic chunk label of w_i and neighboring words
presence of w_i in a **gazetteer**
w_i contains a particular prefix (from all prefixes of length ≤ 4)
w_i contains a particular suffix (from all suffixes of length ≤ 4)
w_i is all upper case
word shape of w_i, word shape of neighboring words
short word shape of w_i, short word shape of neighboring words
presence of hyphen

Fig. 2.16 Features for NER System - (Jurafsky and Martin, 2014)

Approaches to automatic text segmentation and evaluation are discussed in detail in the following sections.

Approaches

Lexical Cohesion Based Approaches

TextTiling, which is an algorithm that is used for automatic text segmentation, was proposed by Hearst (1997). Its basic idea lies in scoring sentences and plotting those scores. Segment boundaries are then considered to lie at the largest valleys on the graph. The algorithm has three steps: (1) tokenisation, (2) scoring adjacent sentence pairs and (3) locating the topic boundaries, depending on the scores that are obtained during the second step.

The first step tokenises the text into words and pseudo-sentences of a predefined size m so that all sentences will have the same size.

For the second step, Hearst uses three methods for scoring sentences: blocks, vocabulary introduction and chains. These methods use the "lexical co-occurrence and distribution of text" (Hearst, 1997).

The block method assigns a score by comparing adjacent blocks (all blocks have the same size k) and measuring how many words occur in both blocks; these similarity scores are computed for every gap between the blocks. The similarity between block b_1 and block b_2 at the gap i can be computed as follows:

$$score(i) = \frac{\sum_t w_{t,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 w_{t,b_2}^2}}$$

where $w_{t,b}$ is the weight assigned to the term that is the frequency of the word, and t is the collection of terms.

The vocabulary introduction method assigns the score to the gaps, depending on the number of new words seen, so that the score between two blocks can be computed as follows:

$$score(i) = \frac{NumNewTerms(b_1) + NumNewTerms(b_2)}{w * 2}$$

The third step is to identify the boundaries, which will allow the algorithm to compute a depth score that measures the distance from the peaks that lie on the sides of the valley to the valley itself. The gaps with the largest depth are selected as the boundaries. In addition, as mentioned previously in section 2.3.3, Kozima (1993) handled the task of automatic text segmentation by *LCP*, which keeps a record of the text's lexical cohesiveness. The algorithm slides a window through the text and plots the cohesiveness scores. It then chooses the valleys that will be the boundaries' positions. The Kozima (1993) algorithm is explained in detail in section 2.3.3.

Feature Based Approaches

Beeferman et al. (1999) investigated the problem of automatic text segmentation using trained statistical models. Their work depends on the approach to feature selection of Berger et al. (1996); Della Pietra et al. (1997), which assigns a probability after each sentence that predicts a boundary existence. The statistical model is $q(b|X)$, where b could be either a *YES* or *NO*, and X denotes the context at some position i in the text corpus with K words on the left and the right of the position i. Therefore, b_i is equal to *YES* if there is a boundary in that position i, and it is equal to *NO* otherwise. They use an exponential model that is incrementally built for feature extraction and uses two types of features: (1) topical features that use a language model for the detection of topics and (2) cue word features that depend on finding cue words that often occur around a segment boundary.

Other Approaches

Choi (2000) proposed an algorithm for automatic text segmentation called *c99*. The algorithm idea was inspired by image segmentation algorithms. The algorithm firstly computes a cosine similarity score between each pair of adjacent sentences x and y using the following equation:

$$sim(x,y) = \frac{\sum_j f_{x,j} \times f_{y,j}}{\sqrt{\sum_j f_{x,j}^2 \times \sum_j f_{y,j}^2}}$$

where $f_{i,j}$ refers to the frequency of word j in sentence i. The algorithm then plots these similarity scores into $n \times n$ image, where each similarity score is represented by a pixel at the sentence's location (*i,j*). n indicates how many sentences in the text are being segmented. The algorithm then applies a 11×11 mask to filter the image, followed by applying a divisive clustering method to increase the density of the coherent squares around the diagonal to extract them. These are considered the boundaries' locations.

Utiyama and Isahara (2001) proposed a statistical method that is domain independent for text segmentation. Because they used a Bayesian model, the algorithm does not require training data. It selects the segment $\widehat{S}$ with the highest probability by the following equation:

$$\widehat{S} = \arg\max_S Pr(W|S)Pr(S)$$

, where $Pr(W|S) = \prod_{i=1}^{m} \prod_{j=1}^{n_i} Pr(w_j^i|S_i)$, and $Pr(w_j^i|S_i) = \frac{f_i(w_j^i)+1}{n_i+K}$ where n_i is the word count in segment S_i, m is the number of segments in the document, K is the size of "different words of the document" and $f_i(w_j^i)$ is the word count of the word w_j in segments S_j and $Pr(S) = n^{-m}$.

Evaluation

P_k, is a text segmentation evaluation metric that was proposed by (Beeferman et al., 1999). It works by sliding a window of size k across the text, where k is set to half the average of the true segment size. The metric gives a penalty of 1 whenever the two windows (one on the true segmented text and the other on the automatically produced segmented text) disagree on

a boundary. These penalties are summed and then normalised between 0 and 1 by dividing the sum by the number of windows.

Pevzner and Hearst (2002) identified four problems of the P_k metric: (1) it penalises false positives more than false negatives, (2) it heavily penalises near misses, (3) it does not penalise some errors in some situations when they fall within k sentences from a true boundary and (4) it is affected by segment size variations. To solve these problems with the P_k metric, Pevner and Hearst proposed a new metric called *WindowDiff*, which is a modification of the evaluation metric P_k. It works by moving a window of a fixed size N over the segmented text. It is a penalty metric; therefore, it compares each time the number of boundaries in the reference includes segmented text within the window with the number of boundaries in the automatically segmented text and gives a penalty when the number of boundaries in the window does not match the number of boundaries in the true window of the text. It is computed by the following equation:

$$WindowDiff(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k})| > 0)$$

where N is the number of sentences in the text, and *b(i,j)* is the number of boundaries between positions i and j.

By contrast, Lamprier et al. (2007) identified that the *windowDiff* metric gives fewer penalties to the errors that are located at the beginning and end of a text. Niekrasz and Moore (2010) stated that both metrics are biased towards the number of boundaries of the hypothesised segmentation. Fournier (2013) proposed a new segmentation evaluation metric called *BoundarySimilarity (B)* to solve the previously mentioned metrics' problems. It works by finding the similarity between boundaries; instead of using a sliding window, it relies on boundary edit distance (BED), which differentiates between full and near misses. BED uses the following main edit operations to measure full and near misses: (1) addition/deletion (AD) to represent full misses, (2) "substitution (S) for confusing one boundary type with another" and (3) n-wise transposition (T), which represents near misses. Figure 2.17 shows

an example of BED operations on two segmented texts s_1 and s_2. We can see that there is a near miss represented by (T), and (M) indicates that we have matching boundaries in the two segmented texts. We also have two AD operations that refer to full misses. These full misses occurred because we set the maximum distance between boundaries n_t to be 2 sentences (the vertical white bars in the figure are sentence boundaries).

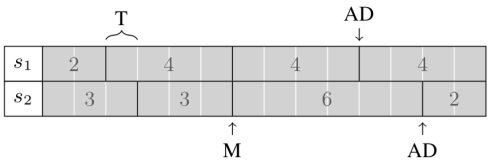


Fig. 2.17 Boundary Edit Operations - (Fournier, 2013)

BoundarySimilarity (B) is then represented by the following equation:

$$B(s_1, s_2, n_t) = 1 - \frac{|A_e| + w_t\_span(T_e, n_t) + w_s\_ord(S_e, T_b)}{|A_e| + |T_e| + |S_e| + |B_M|}$$

where A_e is the set of ADs, T_e is the set of n-wise transpositions, S_e is the set of substitutions and M_M is the set of matching boundary pairs (Fournier, 2013). n_t is the *maximum transposition spanning distance*, which sets the distance in sentences that indicates a near miss between two boundaries. $w_t\_span(T_e, n_t)$ is a scaling function that is used to represent the severity of near misses using a constant w_t, which is set to 0 by default. $w_s\_ord(S_e, T_b)$ is a function that is used to calculate the weight of the substitution operations using a constant w_s.

2.7 Summary

In this chapter we have reviewed prior work relevant to the research we will present below. The focus of our research work is on recognising visually descriptive language in text and on scene recognition in narrative text. Therefore, we began the chapter by presenting previous work on visually descriptive language in Section 2.1 and previous work on scene recognition in Section 2.3.

The rest of the chapter addresses prior work on computational methods relevant to the tasks of recognising visually descriptive language in text and automatic segmentation of narrative text into scenes. Section 2.4 described techniques for supervised text classification, both feature engineering and representation learning techniques. These underlie all of our experimental work below, where we will use them both for recognising visually descriptive language and for scene segmentation. They also underlie modern approaches to named entity recognition, which we discussed in Section 2.5 and text segmentation presented in Section 2.6. Modern approaches to NER use BIO tagging, which we will adapt for identifying segments of visually descriptive language within sentences. Scene segmentation is a type of text segmentation and we will adapt techniques both for computing and for evaluating the latter for the former.

Chapter 3

Research Questions and Methods

This chapter discusses the research questions that are addressed by this project. As discussed in chapter 1, our research addresses two broad areas. The first is the automatic recognition of visually descriptive language (VDL) in text, both at the sentence and sub-sentence level. The second is the automatic segmentation of narrative text into scenes. In this chapter, we refine these broad areas of interest into specific research questions and discuss at a high level the methods we use to attempt to answer these specific questions. Section 3.1 introduces the research questions addressed by this thesis, section 3.2 gives a brief description of the research methods used to address each of the research questions.

3.1 Research Questions

Identifying VDL in Text

1. How well can we automatically classify sentences into three classes, VDL, not VDL, and partially VDL?

Using the definition of VDL supplied in Section 2.1, this question targets sentence-level classification, in which sentences need to be tagged either as visually descriptive, not visually descriptive or partially visually descriptive.

2. Can we detect segments of sentences that are visually descriptive?

This question targets segment-level classification in which segments of sentences (e.g. words or phrases) need to be identified either as visually descriptive, not visually descriptive or impure visually descriptive (see Section 2.1 for precise definitions of these terms).

Scene Definition and Segmentation

3. Can we automatically segment a story into scenes?

This question studies the task of automatic segmentation of narrative text into scenes. We start with the definition of scene provided in SceneML (Section 2.3.2). However, in the course of our work we found that this definition needed to be refined and extended in various ways (see Chapter 5 below). In order to do this, first, a clear definition of what a scene means should be formulated. Next, narrative text (e.g. a novel) is selected from an open source website, and then annotated into scenes. Finally, features are extracted and an automatic segmentation of the story is conducted. The next section explains in detail the methodology to achieve this.

4. Can VDL be used to help in scene segmentation?

Our final research question targets the use of VDL for the task of scene segmentation, and whether it could be a good indicator of scene segments. Since automatic scene segmentation is developed by using a feature-engineering supervised learning approach, VDL can be treated as one of the features of the model.

3.2 Research Methodology

In this section we discuss the methodologies used to address each of the research questions identified in the section above.

Identifying VDL in Text

To address questions 1 and 2, the following methodology is followed:

- **Data selection and annotation:** The data that are used in this part of the research comprise a corpus of text from a children's story and from a number of samples of different genres in the Brown Corpus. Certain chapters of these books were randomly selected and annotated as either VDL, not VDL or partially VDL. The annotation was done at the segment level and then sentence level annotation annotation was inferred from this. We followed the definition presented by Gaizauskas et al. (2015) that state: "by text segment here we mean a phrase, clause, sentence or sequence of sentences, i.e. a sequence of contiguous words". Details of the data are provided in chapter 4, section 4.1.

- **Data pre-processing:** Data are pre-processed first: tokenisation (i.e. where a sentence is divided into smaller parts (words)) and stop word removal (i.e. where a list of English common words are removed from the text) are applied to the data.

- **Feature extraction:** The features are then extracted from the pre-processed text via different methods, such as tf-idf term weightings, word embeddings and methods that use other resources, such as VerbNet, WordNet and combinations of such. Feature extraction methods are discussed in details in section 4.2.

- **Training:** The features are then fed into a supervised learning classifier, along with the necessary labels, to train the classifier. As research questions one and two represent two different classification tasks, they are managed using separate methodologies:

    1. The classification task for research question one is treated as normal sentence level classification where each sentence is to be given a tag as VDL, not VDL, or partially VDL.
    2. The classification task for research question two is a segment level classification task. That is, segments possibly smaller than a full sentence (e.g. words or phrases or even the whole sentence) are to be tagged as VDL, not VDL, or impure

VDL. To achieve this, the task here is treated as a Named Entity Recognition (NER) task, where the the approach of IOB is employed. Under this approach, each word is assigned a tag indicating whether it falls inside (I), outside (O), or (B) beginning of the designated segment.

Both tasks listed above have been experimented with different classifiers, SVM, kNN, and MNB classifiers. Further discussion on the experimentation is given in Chapter 4.

- **Evaluation:** *accuracy* is a widely used evaluation metric in classification tasks, however, since the data used here is imbalanced, using accuracy can be misleading. Balanced accuracy (Brodersen et al., 2010) is used as an evaluation metric as it gives equal importance to all classes (minor and major classes) as it computes the average accuracy achieved across both the minority and majority classes. This is given by:

$$Acc_{balanced} = \frac{1}{N} \sum_{i=1}^{N} \frac{P_i}{M_i} \tag{3.1}$$

where N is the number of classes, P_i is the number of correct predictions of class i, and M_i is the number of instances of class i.

Details of the above methods are explained in detail in chapter 4, which includes experiments that are related to the first and second research questions.

Scene Segmentation

The second set of questions address the task of automatic segmentation of narrative text into scenes. Including the use of VDL in the task of automatic scene segmentation. The following discusses the methodology that is used to address these questions:

- **Data collection and annotation:** Until now, no annotated data in English have been used for the automatic segmentation of narrative text into scenes. Therefore, a corpus of text that is dedicated to this task needs to be assembled and annotated. The data collection/selection and annotation comprises two stages, as explained below:

  - **Data Collection:** includes the identification and acquisition of the raw text that is going to be annotated. Several chapters need to be selected from narrative books (different novels from project Gutenberg).
  - **Data Annotation:**
    * **guideline specification:** Before annotation can occur, guidelines must be formulated, and a clear definition of a *scene* needs to specified. While the definition of scene provided by SceneML as described in Section 2.3.2 provides a good starting point, further refinement was required to turn this into a set of guidelines that covered the cases we found in dealing with real data. Section 2.3.2 details the definition that was chosen for *scene* and section 5.1 presents the guidelines that were formulated based on the SceneML.
    * **Data annotation:** This process includes recruitment of annotators, training them and assessing their understanding of the task and the guidelines. It also includes application of the annotation specification to the collected text. In which annotators were asked to annotate scenes segments in the text provided, a segment here can be a sentence, a paragraph, or a whole chapter. Details of the annotation process is given in Chapter 5.

- **Data Pre-processing:** After the data collection and annotation phase is complete, the data are pre-processed to create a suitable format for being fed into supervised machine learning classifiers. The pre-processing phase includes sentence splitting (splitting text into sentences) and tokenisation (splitting sentences into words).

- **Scene Segmentation:** The scene segmentation task is addressed using three different approaches. As scene segmentation is a complex task, thus results of these distinct approaches are then compared with a view to understanding the strengths and weaknesses of these different approaches. All the three approaches are developed using supervised learning methods.

- **Approach 1:** The task here is treated as a sequence labelling problem as the order of sentences is important for this particular task, as is knowing whether the previous sentence was deemed to be on a scene boundary or not. It is very similar to Named Entity Recognition (NER) task, but instead we are dealing with sentences instead of words. The sequence here is the sequence of sentences that belong to the same narrative text that is being segmented. Each sentence is then given a tag (1 for sentences on the boundary of a scene and 0 otherwise). To train the classifier, the following steps are followed:
  - **Feature extraction:** After data are pre-processed, features are extracted via widely-used methods, such as computing tf-idf term weightings and word embeddings. VDL could be identified as it might be a useful feature to help detect segment boundaries, on the hypothesis that there might be more VDL segments at the beginning of a scene segment to describe a change in location than occur at other positions in a scene. In addition, named entity recognition is used to extract named entities such as time that could be used as a feature. Other features and feature extraction methods are described in detail in Chapter 6.
  - **Training and testing:** After features are extracted, they are designated as part of either the training set or the test set. The training data are then used to train supervised learning models. For this task the CRF is chosen. The test data are used to test the learning model that predicts segment boundaries.
- **Approach 2:** The second approach treats the task not as a sequence labelling task but as an isolated sentence classification task using a pre-trained language model. The model is implemented with Ktrain library (Maiya, 2020) and with the use of BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018). The model is trained and tested using two different settings: one with BERT-cased, and the other with BERT-uncased. The models with the two different variations were evaluated using the metrics given above.

- **Approach 3:** The scene segmentation task here is treated as a sentence pair classification using a pre-trained language model. As the task name suggests, a pair of two sentences is given a tag. For our particular task and to capture as much contexts as possible, each sentence is paired with a context that is the sentence concatenated with the two preceding sentences and the following two sentences. In addition, two settings have been implemented and tested one with BERT-cased and the other with BERT-Uncased. The model is then evaluated using the evaluation metrics above.
- **Evaluation:** Since the task is treated as a classification task, classification evaluation metrics are used. However, the data used here is also imbalanced, so balanced accuracy (as explained above) is going to be used instead of the accuracy score. In addition, the F1 score is also used here as an evaluation metric as it is also suitable for imbalanced data (as it considers both precision and recall) and to be able to compare our work with others who used the F1 score to evaluate their models. The F1 score is computed by:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

While precision is given by:

$$\text{Precision} = \frac{TP}{TP + FP}$$

and recall is:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Details of all three approaches to scene segmentation together with results from training and testing them on the annotated corpus described above are presented in Chapter 6.

3.3 Conclusion

This chapter has outlined the research questions addressed in this thesis, specifying each in some detail. In addition, it gave a brief explanation of the methodologies we followed to address each of the research questions.

Chapter 4

Recognising Visually Descriptive Language

Adopting the definition of VDL proposed by Gaizauskas et al. (2015) and discussed above in section 2.1, that VDL is language that asserts propositions whose truth can be confirmed through visual sense alone, in this chapter we address the first two of the research questions introduced in the last chapter. First, in section 4.1 we give a brief description of the data used in this chapter. Then, in section 4.2, we investigate the specific task of classifying sentences as wholly, partially or not at all visually descriptive. We discuss different approaches to the task, and report results on automatic classifiers trained on the annotated dataset. Then, in section 4.3, we investigate the task of segment level classification of VDL and report results on this task too.

4.1 Data

The corpus compiled by (Gaizauskas et al., 2015) (**VDL-C1**) is the initial annotated corpus. The data that was selected for the annotation task in Gaizauskas et al. (2015) included two chapters from The Wonderful Wizard of Oz (173 sentences)and six samples from the Brown chosen (779 sentences) that were extracted from various pre-selected categories (two news articles, one biography and three novels). All annotations were done on segment level; 3

annotators annotated the WOZ texts and 2 annotators annotated Brown texts using brat rapid annotation tool. The sentence level annotations were inferred from the segment level ones.

The corpus used in this work (**VDL-C2**) is an extended version of (VDL-C1) that contains six additional chapters from WOZ that were annotated by 5 annotators[1]. The extended WOZ has a total of 916 sentences. As the sentences are multiply annotated, we further filtered the annotations of the extended corpus by choosing only the sentences where both annotators agreed, in the case of two annotators, and where three or more agreed in the case of five annotators. This gave a total of 1,337 sentences. Table 4.1 presents the statistics of the extended Corpus including the inter-annotator agreement at sentence level.

The third annotated corpus used in this study is (**VDL-C3**) which a corpus that extends (VDL-C2) with Flickr data. We used a proportion of Flickr8k, comprising 400 captions, and labelled them all as 'VDL'. Choosing a small subset of Flickr8k helps to prevent the dataset from becoming highly imbalanced as all captions are considered 'VDL'.

4.2 Sentence Level Classification

Supervised machine learning was used to automatically classify sentences as either 0 (NOT VDL), 1 (VDL), or 2 (partially VDL). Subsection 2.2 summarises initial work on feature extraction and feature-based classification done during MSc. Subsection 4.2.1 covers additional work on feature extraction during PhD. Subsection 4.2.2 reports results from all the feature based classification approaches and Subsection 4.2.3 explores an application of these approaches to sentence-based VDL classification of entire novels from Project Gutenberg. In order to carry out the task of classifying sentences into three classes (NOT VDL, VDL, or partially VDL), several feature extraction approaches were explored as discussed in the following subsections.

[1] Annotators were UK based former PhD students, they were paid base GTA rate.

	Text	Type	$\|S\|$	S=1	S=2	VDL	IVDL	%Agree	κ	IoU
Oz	Ch1	Children's Story	59	0.19	0.54	40	14	0.68	0.47	0.72
	Ch3	Children's Story	124	0.12	0.40	57	18	0.78	0.64	0.73
	Ch5	Children's Story	112	0.07	0.43	48	21	0.72	0.52	0.56
	Ch7	Children's Story	95	0.13	0.51	51	47	0.76	0.73	0.65
	Ch9	Children's Story	78	0.12	0.42	38	23	0.72	0.69	0.62
	Ch11	Children's Story	203	0.15	0.40	99	33	0.73	0.56	0.70
	Ch13	Children's Story	52	0.06	0.67	41	15	0.77	0.51	0.73
	Ch15	Children's Story	193	0.03	0.30	52	26	0.72	0.41	0.40
Brown	A13	Sports Reportage	111	0.11	0.27	25	20	0.78	0.60	0.51
	A30	Culture Reportage	128	0.04	0.34	31	21	0.78	0.56	0.57
	G32	Biography	101	0.02	0.47	32	29	0.74	0.50	0.43
	L05	Mystery Fiction	151	0.21	0.31	65	20	0.87	0.79	0.63
	N13	Western Fiction	122	0.12	0.46	58	38	0.70	0.49	0.57
	P15	Romance Fiction	179	0.08	0.24	40	21	0.82	0.62	0.73

Table 4.1 Descriptive Statistics and Sentence Level Inter-annotator Agreement Scores of VDL-C2 Corpus. Column $|S|$ is the total number of sentences in each chapter. Column **S=1** is the proportion of VDL sentences in each chapter. **S=2** is the proportions of sentences of partially VDL sentences in each chapter. **VDL**, is the number of VDL segments. **IVDL** is the number of IVDL segments in each chapter. Column **%Agree** is the percentage agreement and column κ is the kappa score. **%Agree** and κ are both used to assess inter-annotator agreement at the sentence level. Finally, column IoU is the intersection over union and it is computed at segment level.

4.2.1 Further Work on Feature Extraction for Sentence Level VDL classification

To extend our early work on feature extraction, as detailed in section 2.2, we explored various enhancements, introducing several additional features and experimenting with variations and modifications to existing features.

Word Embeddings

Word embeddings are vector representations that capture the semantic similarities between words. They result from a process designed to capture the distributional properties of words. Since semantically similar words tend to have similar distributional properties, i.e. occur in similar contexts, closeness between word embeddings in vector space gives a measure of semantic similarity between words. Approaches based on neural networks embed words into a low dimensional space.They represent words as vectors of real numbers, so that words with similar meanings would have similar vectors (Jurafsky and Martin, 2023; Levy et al., 2015; Mikolov et al., 2013). Word embeddings are used here to extract features from text. Word vectors are obtained for each word in the sentence using spaCy. Then, the average of the word vectors is taken to produce a vector for the sentence.

SpaCy uses a word vector model that provides 300-dimensional GloVe vectors for over 1 million English terms (spa). GLoVe is a global unsupervised learning model that obtains vector representations of words (Pennington et al., 2014). It is trained on aggregated global word-word co-occurrences. It is also trained on "five corpora of varying sizes: a 2010 Wikipedia dump with 1 billion tokens; a 2014 Wikipedia dump with 1.6 billion tokens; Gigaword 5 which has 4.3 billion tokens; the combination Gigaword5 + Wikipedia2014, which has 6 billion tokens; and on 42 billion tokens of web data, from Common Crawl" (Pennington et al., 2014, p.1538) .

WordNet - Variations

We considered several ways to attempt to make more effective use of WordNet:

- Obtaining an overall visual score by summing the visual score of each verb, object and subject. This approach works by extracting the main verb of the sentence and then looking for its type in WordNet to determine whether it is a visual verb or not. If it is visual, the visual score is incremented by one. After this, we extract the object and subject of the sentence and look "for them in the WordNet hypernym hierarchy (taking the first synset). If they are visual, the visual score is incremented for each one of them. The feature representation here is different from the ones in section 2.2 in that instead of having binary values, here we have an integer value that record the overall visual score.

- Including all objects and subjects of the sentences rather than only the object and subject of the main verb. The original WordNet approach –described in section 2.2 – only includes the subject and object of the main verb; however, here we look for all the objects and subjects in the sentence and their hierarchies in WordNet to determine whether they are visual or not. We then include this information in the feature vector.

- Having six features instead of three; three features indicating the visual information and three features indicating whether the word was found in WordNet. Thus, a feature is added that indicates that the main verb, the object and the subject of the main verb are all visual. In addition, a feature that indicates that they are found in WordNet is added for each of them. This is meant to distinguish between words that are not visual and words that are not found in WordNet, since the classifier might consider words not found in WordNet as not visual.

- Using Named Entity Recognition for person, organisation and location to indicate the visualness of a word. One of the reasons that a word may not be found in WordNet is that it might refer to a person's name or a place's name. Thus, in the WordNet approach, a condition is added to test whether the subject or object is found in WordNet. If it is not found, then we use the named entity recognition from NLTK to indicate which type of word it is. If the named entity recognition indicates that the word refers to a PERSON, PLACE or ORGANISATION, a feature that indicates it is visual is added.

After testing all of the above methods of improving WordNet's feature extraction approach, unfortunately, it appears that they did not help to improve the accuracy score.

Combining Features

We tried to merge the features of the WordNet approach with those of the tf-idf approach by combining their vectors. This combination is denoted by (**WordNet+tf-idf**) in Table **??**. Then, the word embedding features matrix was merged with the tf-idf matrix, which is denoted by (**Embedding+tf-idf**) in Table **??**. The merging of features is done by horizontally stacking (concatenating) matrices of the features by using the function `scipy.sparse.hstack`.

4.2.2 Experiments, Results and Analysis

In this section, we report and discuss the results of our experiments on the VDL sentence-level classification task, comparing three supervised classifiers with the different representations explained above. The three classifiers that were explored and tested are: (i) a weighted k-nearest neighbours (kNN) classifier with a Euclidean distance measure (we use $k = 5$ in our experiments); (ii) support vector machine (SVM) with a linear kernel; and (iii) multinomial Bayes (MNB). We used the implementations in scikit-learn[2].

We tested our proposed classifiers on two corpora VDL-C2 (presented in the following lines), and VDL-C3 (presented at the end of this subsection). For **(VDL-C2)** we annotated the corpus via 10-fold cross-validation. We report the average accuracy across the different folds.

We tokenised all sentences in the dataset as a preprocessing step. Stop words were only removed for the **word embedding** representation. We did not remove stop words for **tf-idf** because the approach worked better without stop word removal. We also did not remove stop words for the explicit representations, as they require complete sentences for parsing to be performed.

[2]http://www.scikit-learn.org/

We found the filtered dataset to be skewed towards class **0** (non-VDL, 53.70%), compared to classes **1** (fully VDL, 9.65%) and **2** (partially VDL, 36.65%). Thus, besides the *accuracy* metric, we also evaluated our classifiers using a *balanced accuracy* metric to account for the class imbalance. Formally:

$$Acc_{balanced} = \frac{1}{N}\sum_{i=1}^{N}\frac{P_i}{M_i} \tag{4.1}$$

where N is the number of classes, P_i is the number of correct predictions of class i, and M_i is the number of instances of class i.

Model	**VerbNet**	**WordNet**	**tf-idf**	**Embedding**	**WordNet+tf-idf**	**Embedding+tf-idf**
kNN	0.2901	0.4543	0.5124	0.6804	0.5624	0.6941
SVM	0.1786	0.5108	0.7380	**0.7434**	0.7324	0.7378
MNB	0.1790	0.4928	0.4647	–	0.4536	–

Table 4.2 Balanced Accuracy Results. Note that the MNB classifier does not support the negative values of the **Embedding**-based representation.

Results

Table 4.2 shows the results using our balanced accuracy metric, results were obtained using three classifiers with 10-fold cross-validation. Note that we did not test the MNB classifier on **Embedding**-based representations as MNB does not allow negative feature values (word embeddings can have negative values). It is clear that the **VerbNet** approach with the KNN classifier scored the lowest, whereas the **WordNet** approach was slightly better. Balanced accuracy scores increased with **tf-idf** to 0.738, with the SVM and **Embedding** feature scoring to 0.743. In addition, combining **tf-idf** with **Embedding** features (**Embedding+tf-idf**) and **WordNet** features with **tf-idf** (WordNet features with **tf-idf** (WordNet+tf-idf)) did not have significant improvement.

Results for **tf-idf**, **Embedding**, **WordNet+tf-idf** and **Embedding+tf-idf** with the SVM classifier are comparable, with **Embedding** having a slight edge, giving the highest balanced accuracy score of 0.7434. As a baseline, we also evaluated a simple majority class prediction, which resulted in a balanced accuracy of 0.3333.

Analysis

Multiple factors affect the accuracy of the classifiers based on **WordNet** and **VerbNet** representations. One is parser inaccuracy, which may lead to incorrect identification of the main verb and/or the subject and object associated with the verb. The VerbNet database may also not cover all English verbs, so some verbs may be unclassified. In addition, VerbNet has many classes of verbs (237 classes) compared to WordNet (15 classes). This is likely to reduce the accuracy of prediction in the case of Verbnet and may well be the reason why WordNet's accuracy results are higher than those of VerbNet. The following section provides more in-depth analysis of errors caused by either the parser or the WordNet database.

As we previously reported, class **1** represents only a small part of the data. This class imbalance may also be an issue with learning to classify VDL.

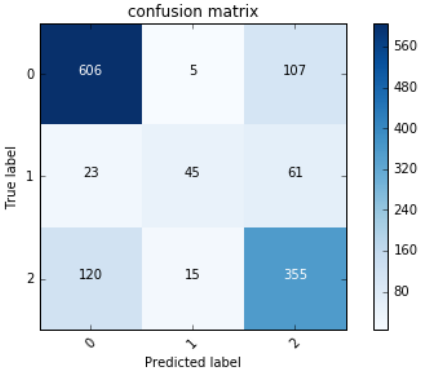


Fig. 4.1 Tf-idf Confusion Matrix

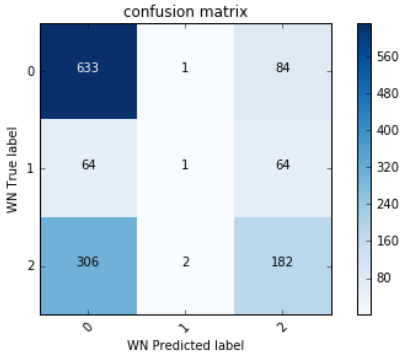


Fig. 4.2 WordNet Confusion Matrix

Figure 4.1 shows the confusion matrix for the **tf-idf** representation with the SVM classifier. For class **0** (non-VDL), 112 of 718 were misclassified (5 were misclassified as class **1**, and 107 were misclassified as class **2**). For class **1** (fully VDL), only 45 of the 129 total were classified correctly, whereas 61 were misclassified as class **2** and 23 were misclassified as class **0**. For class **2** (partially VDL), 355 out of 490 were correctly classified, whereas 15 were misclassified as class **1** and 120 were misclassified as class **0**. This classifier frequently misclassified text that contains VDL. Figure 4.2 shows another confusion matrix, in this case for the **WordNet** representation with the MNB classifier. Comparing it with the tf-idf confusion matrix. What is striking in the WordNet matrix is that (1) only 1 VDL is

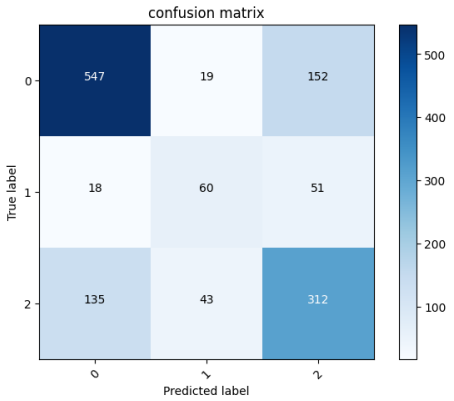


Fig. 4.3 Word embeddins Confusion Matrix

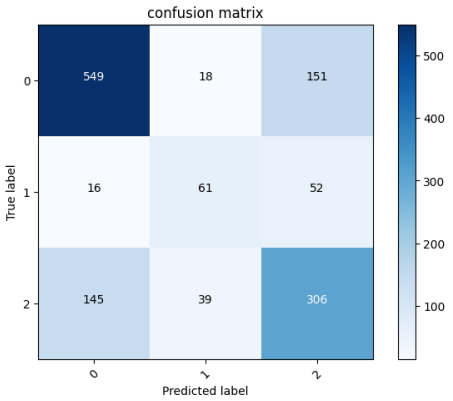


Fig. 4.4 Word embeddings+tf-idf Confusion Matrix

recognised as such with many more true VDLs being incorrectly classified as non-VDLs (2) fewer partially VDLs were correctly classified with many more incorrectly classified as non-VDL. Overall WN leans to classifying things as non-VDL.

In addition, Figure 4.3 and Figure 4.4 show the confusion matrices for the top two features: **Embedding** and word embedding combined with tf-idf (**Embedding+tf-idf**), respectively. The confusion matrices of the two features share similarities in their tendencies to lean towards classifying instances as non-VDL. In the confusion matrix of **Embedding**, a total of 547 non-VDL instances were correctly identified, while **Embedding+tf-idf** improved this count slightly to 549. Both matrices displayed comparable correctness in identifying VDL instances, with **Embedding+tf-idf** showing a slightly higher count for class **1** (fully VDL). However, both approaches encountered challenges in correctly identifying partially VDL instances, with misclassifications observed in both directions (as non-VDL and fully VDL).

Notably, the confusion matrices for both features exhibit a common trend of effectively distinguishing non-VDL instances. However, they differ in their handling of partially VDL instances, where both approaches face challenges with misclassifications in both directions (as non-VDL and fully VDL). This suggests a complexity in accurately identifying instances with partial VDL characteristics.

Error Analysis

In order to improve the feature extraction methods that are already applied, we needed to know what kinds of errors or mistakes are happening in them during the feature extraction. Thus, starting with an attempt to improve the WordNet feature extraction method, we tried to identify the number of missed words either by the parser or by the WordNet database. By looking deep into the data, the following errors were found:

1. **Missed objects**: A total of 935 (69.9%) objects were missed from the parser; this meant that no object was found for the main verb that was extracted by the parser in the sentence.

2. **Missed Subjects**: A total of 274 (20.5%) subjects were missed from the parser; this meant that no subject was found for the main verb that was extracted by the parser in the sentence.

3. **Wrong Verbs** There were 332 (24.8%) wrong verbs; by 'wrong verb' we mean that the main verb that was extracted by the parser from the sentence was not actually a verb.

4. **WordNet Object Errors**: There were 97 (7.3%) objects extracted by the Stanford parser that were not found in the WordNet database. Thus, the feature list was expanded to include six features instead of three, as explained earlier.

5. **WordNet Subject Errors**: There were 425 (31.8%) subjects extracted by the Stanford parser that were not found in the WordNet database. This may have occurred because the subject might have referred to a person's name, street name or place, for example. A named entity recognition section was added to the WordNet feature extraction method to tackle this problem.

Experiment of Adding Flickr Data (VDL-C3) We tried to train and test with and without the Flickr data to determine the effect this data could have on the original ones. The following

outlines the proportions of the training and testing data on WZ, Brown and Flickr, along with the accuracy results.

- Training data: 90% on Brown and WZ; Testing data: 10% on Brown and WZ. Gave an accuracy score of 0.7611.
- Training data: 90% on Brown, WZ and Flickr; Testing data: 10% on Brown and WZ. Gave an accuracy score of 0.753.
- Training data: 90% on Flickr and only the not VDL and partially VDL portions of Brown and WZ; Testing data: 10% on Brown and WOZ. Gave an accuracy score of 0.4.

The above results show that the addition of the Flickr data to the training corpus worsened the accuracy results. This is due to the nature of the original corpus (Brown and WZ), which is completely different from those of Flickr8k.

4.2.3 An Application of VDL Sentence Classification to Novels in Project Gutenberg

To demonstrate that our sentence-level VDL classifier can be applied to gain insight into different authors' use of visually descriptive language, we also performed an experiment where we attempt to classify sentences from books from Project Gutenberg[3]. We selected three books for each of the five top authors in Project Gutenberg: Charles Dickens, Arthur Conan Doyle, Mark Twain, William Shakespeare, and Lewis Carroll. We segmented the texts into sentences, and classified all sentences using the SVM sentence-level classifier with the **Embedding+tf-idf** representation as it is the most accurate classifier in our experiments.

Figure 4.5 shows the distribution of our classifer's predictions of non-VDL (**0**), fully VDL (**1**), and partially VDL (**2**). Conan Doyle and Twain have the highest proportion of pure VDL sentences, while Twain and Dickens have the largest proportion of sentences classified as partially VDL. On the other hand, most of Carroll's and Shakespeare's sentences are not

[3]http://www.gutenberg.org/

classified as visually descriptive. Overall, the proportion of sentences classified as fully VDL was small, mirroring the class imbalance in the training set.

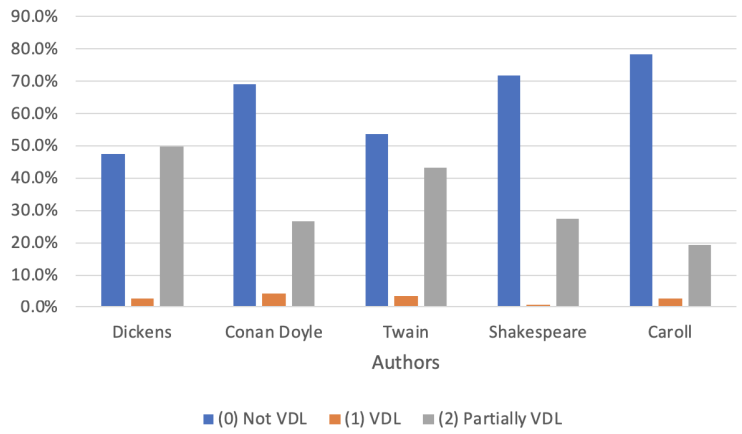


Fig. 4.5 Classification of The Top Five Authors' Books from Project Gutenberg

4.3 Segment Level Classification

This section addresses the second research question explained in section 3.1, this question focuses on segment-level classification of VDL, where words or phrases inside sentences need to be classified as either visually descriptive, not visually descriptive, or impure visually descriptive.

4.3.1 Supervised Learning Approach

Using the data explained in section 4.1, we used supervised machine learning to train classifiers to predict segment boundaries. The problem is treated as a Named Entity Recognition (NER) problem adapting the idea of **IOB** in which each word is given a tag; inside (**I**) or outside (**O**). The Begin (**B**) tag is not used here as we only have one type of a segment (VDL), there is no need to distinguish between segments as we do not have overlapping text spans in our VDL segments (any two adjacent chunks of VDL are just one long chunk of VDL). Named Entity Recognition is explained in detail in section 2.5.1. We used four classifiers: conditional random fields (CRF) – as CRF models capture the interdependencies between

nearby words and contextual data. CRF models may manage complicated entity boundaries and increase the overall precision and recall segments tags by taking the surrounding context into account (CRF is explained in detail in section 2.4.2). We also wanted to see how the model performs using other classifiers that are frequently used for text classification tasks. Therefore, we conducted further experiments with the Multinomial Bayes classifier (MNB), Perceptron and Support Vector Machine classifier (SVM). Feature extraction was used to extract features, they are: the word itself, the word before and the word after, POS of the word and the words before and after. Additionally, suffix features including the last three and last two characters of the word were added to capture potential morphological information. Binary flags for the beginning (BOS) and end (EOS) of the sentence were added to the feature list to capture the position of each word in the sequence. In addition, we included a feature that indicates if the word is visual or not, for the word itself and the words before and after.

To know if a word is visual or not, first we get the *concreteness value* of the word by looking in the list of words in the concreteness file. Brysbaert et al. (2014) presented concreteness levels for 40,000 generally known English word lemmas, 37,058 English words and 2,896 two-word expressions (e.g. *zoom in*). They used crowdsourcing to collect data, having 4000 participants. So, if the word is found in the concreteness file, then its visualness is given based on the concreteness values, if it is above or below 2.5 as concreteness values ranges from 0 to 5.

If the word is not found, then we check the part of speech of the word, and if it is a verb then we search WordNet to get the visualness of the verb. As described above in Section 2.2, WordNet stores verbs in 15 files, the names of the 15 WordNet database files that store verbs are treated as verb classes and each class is manually determined to be either visual or not. The algorithm determines which verb class the verb occurs in. If that verb class is visual, a feature of value 1 is added or 0, otherwise.

If the word is not a verb, then WordNet hierarchy is searched for the word. The algorithm looks for the hypernyms of the word, going up the hypernym hierarchy and checking each time if the hypernym contains the word ‘physical’ until it either finds the word ‘physical’ or it reaches the root hypernym. If the word ’physical’ is found, this indicates that the argument

Word	Visualness	Tag	Concreteness
House	1	I	5.0
Table	1	I	4.9
Ran	1	I	3.82
Road	1	I	4.75
Bite	1	I	4.44
Little	1	O	3.67
Protect	1	O	2.86
Green	1	I	4.07
Burned	1	I	3.72
Sun	1	I	4.83
Carefully	0	I	2.15
Great	0	I	1.81
Impression	0	I	2.23
Good	0	I	1.64
Quite	0	I	1.72
Strange	0	I	1.86
Love	0	O	2.07
Wicked	0	O	2.11
Moment	0	O	1.61
Possible	0	O	1.56
Think	0	O	2.41

Table 4.3 Examples of words with their corresponding concreteness scores, visualness feature values, and the tag of the word.

is visual and a feature with a value 1 is added to the feature set. Otherwise, a value of 0 is added.

Finally, if the word is not found in WordNet, NER is run on the sentence to get the type of each word. Then we determine if it is visual or not based on its type (i.e. if it is of type *Person* or *Organisation*). Table 4.3 shows an example of a few words with their concreteness values and the obtained visualness feature.

Results and Analysis

Table 4.4 shows accuracy, precision, recall and F1 scores for each one of the classifiers. The CRF classifier is experimented with three variations of the features:

- **CRF-All** with all the features described above.

Classifier	Precision	Recall	F1
CRF-all	**0.81**	**0.81**	**0.81**
CRF-NoVisual	0.80	0.80	0.80
CRF-ExtContext	0.80	0.80	0.80
SVM	0.78	0.76	0.76
MNB	0.75	0.74	0.74
PER	0.75	0.70	0.70

Table 4.4 Classification results using four classifiers: conditional random fields (**CRF**), Support Vector Machine classifier (**SVM**), Multinomial Bayes classifier (**MNB**), and Perceptron (**PER**). CRF is tested for three variations of features, **CRF-All** with all the features, **CRF-NoVisual** without the visualness feature, and **CRF-EXtContext** with extended context.

- **CRF-NoVisual** is the CRF classifier with all features except the visuallness features.
- **CRF-ExtContext** is the CRF classifier with all features but with extended context that include two words before and two words after the test word.

The results from the classification table reveal that the Conditional Random Fields classifier, with all features (CRF-All), demonstrates superior performance with an accuracy, precision, recall, and F1-score all at 0.81. It appears that the visualness feature has a positive impact on the classification results, as the exclusion of the visualness feature in CRF-NoVisual model reduces the performance to (0.80). In addition, extending the context of the feature list does not enhance the performance, in fact it lowers the accuracy from 0.81 to 0.80. The SVM classifier achieved lower results (F1 of 0.77), but falls short compared to CRF. Multinomial Bayes (MNB) and Perceptron (PER) classifiers show lower overall performance, with MNB performing marginally better than PER. Overall, CRF classifier with its variations outperform all other classifiers, and that because CRF can capture the sequential dependencies between elements in the segment level classification of VDL.

As a baseline, we also used the majority class classifier that always predicts the majority class. The majority class baseline achieved a precision of 0.30, recall of 0.54, and F1-score of 0.38. Comparing this baseline to the other classifiers, it serves as a benchmark for performance evaluation. While all other classifiers outperform the baseline, it is clear that

the CRF classifier, especially the CRF-All model emerged as the most effective model in this context.

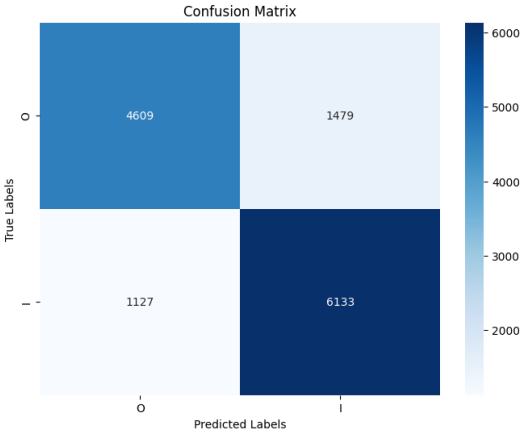


Fig. 4.6 CRF-All Confusion Matrix

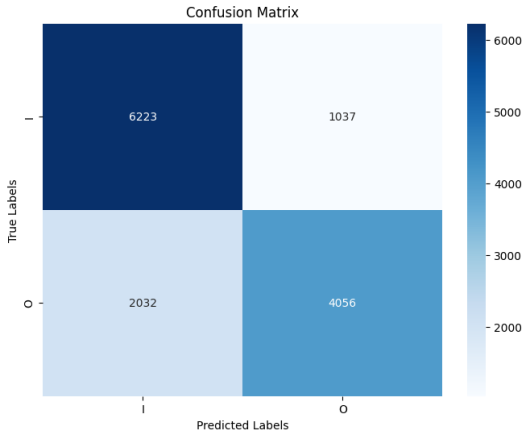


Fig. 4.7 SVM Confusion Matrix

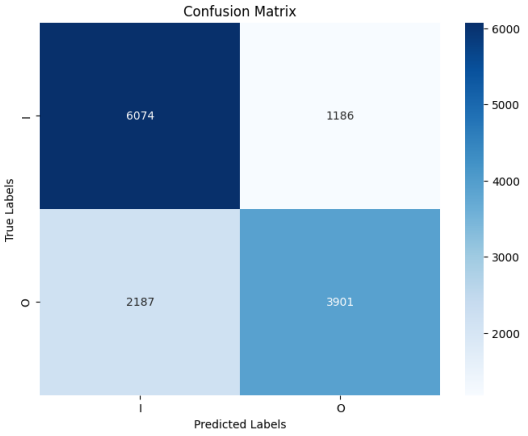


Fig. 4.8 MNB Confusion Matrix

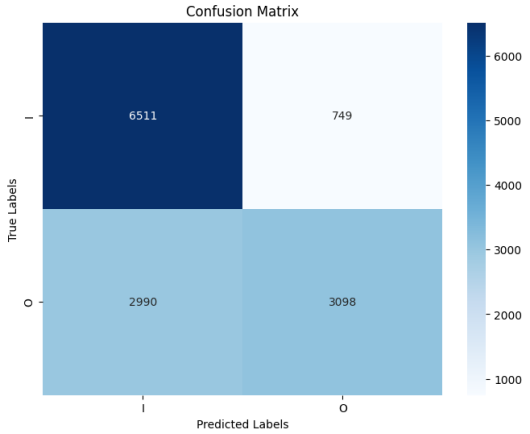


Fig. 4.9 Perceptron Confusion Matrix

Figures 4.6, 4.7, 4.8, and 4.9 show confusion matrices for CRF-All, Svm, MNB, and PER models respectively. The model CRF-All achieved a balance with 4609 true negatives and 1479 false positives for class 'O', while for class 'I' it achieved 6133 true positives and 1127 false negatives. Despite having a larger false negative count of 2032, the Support Vector Machine (SVM) classifier demonstrated decent performance in recognising 'I' instances, with 4056 true positives. MNB classifier shows similar performance to SVM, however, for

class 'I', SVM achieves a higher true positive count (4056) compared to MNB (3901). On the other hand, MNB displays fewer false positives (1186) for 'I' compared to SVM (1037). The Perceptron (PER) classifier, despite capturing a considerable number of 'I' instances (3098 true positives), also displayed a substantial number of false negatives (2990).

In our attempt to delve deeper into the analysis, an error analysis was conducted to examine cases where the classifier predicts the wrong label. We investigated the 'Visualness' feature values and part-of-speech (POS) tags for these misclassified cases for each class. The breakdown is as follows:

- Class 'I' (true label):
  - Total error count: 1051;
  - 58.29% of misclassified cases (612 instances) occurred with visualness value of 0;
  - And 40.50% of total errors (439 instances), occurred with visualness value of 1;
  - POS counts: 'VERB': 179, 'NOUN': 159, 'ADP': 138, 'PRON': 123, 'DET': 99, 'ADJ': 98, 'PUNCT': 92, 'ADV': 80, 'CCONJ': 38, 'PROPN': 24, 'INTJ': 12, 'PART': 4, 'NUM': 4, 'X': 1.
- Class 'O' (true label):
  - Total error count: 1462;
  - Among the total errors, 59.50% (869 instances) were associated with a visualness value of 0;
  - And 40.50% (593 instances) were linked to a visualness value of 1;
  - POS counts: 'VERB': 248, 'NOUN': 239, 'ADP': 233, 'DET': 168, 'ADJ': 144, 'PUNCT': 106, 'PRON': 96, 'CCONJ': 93, 'ADV': 82, 'PROPN': 27, 'NUM': 13, 'INTJ': 10, 'PART': 3.

This breakdown provides insights into the patterns of misclassifications based on the visualness feature for each class. It appears that, for both classes, misclassifications occur more

frequently when the visualness feature is 0. This information can guide further investigation into why instances with visualness = 0 are challenging for the classifier.

For both classes, the most common POS tags among misclassified words are verbs (VERB) and nouns (NOUN). In Class 'I', adpositions (ADP) and pronouns (PRON) also stand out, while adpositions (ADP), determiners (DET), and adjectives (ADJ) are more common in Class 'O'. In addition, among the total number of misclassified words for both classes, there are 428 words not found in the concreteness file. While examining the misclassified words with a concreteness value of < 2.5, an effort was made to identify words that potentially should have been assigned a visualness value of 1 instead of 0. Among these misclassified words, a few proper nouns and animal names were identified. However, their occurrence was relatively minimal compared to the overall size of the dataset.

These cases were analysed using the CRF-All model. In an attempt to enhance performance, we experimented with the removal of stop words and punctuations, considering their presence in the examined faulty cases. Unfortunately, that decreased the accuracy to 0.79

4.4 Summary

In this chapter we have proposed the task of recognising visual language as a potentially interesting and useful challenge for vision and language research. It addresses the first two research questions described in section 3.1.

We have recapitulated the definition of VDL as presented in Gaizauskas et al. (2015) and reported the extension of our corpus of VDL-annotated text.

Section 4.1 describes the data used in this chapter, the original VDL data (VDL-C1), the extension of VDL data (VDL-C2), and the Flickr 8k dataset (VDL-C3) used by one of the experiments.

Section 4.2 addresses the first research question of how well can we automatically classify sentences into three classes, VDL, not VDL, and partially VDL? We start by summarising initial work in the task carried out during my MSc. We then present initial, promising results of developing classifiers for carrying out the sentence level task of distinguishing sentences

that are wholly VDL, partially VDL or not VDL at all. We present several approaches taken to address this task such as VerbNet, WordNet variations, TF-Idf and word embeddings using 3 different machine learning classifiers. Our experiments showed that using word embeddings and an SVM classifier achieved the highest results in classifying text as VDL, partially VDL and not VDL with (0.74 balanced accuracy).

Section 4.3 addresses the second research question of can we detect segments of sentences that are visually descriptive? We treated this task as a sequence labelling task in which each word is given an inside (I) or outside (O) tag. We experiment with different lexical and morphological features and features pertaining to the position of the word in the sentence. This is used along with different context sizes. Results showed that that combining all former features with a context size of one word before and one word after the test word using a CRF classifier achieves the highest F1 score of 0.81.

The work done in this chapter provides resources that help us in our automatic scene segmentation task presented in Chapter 6 in which we investigate the impact of applying the results of automatic VDL to the task of automatic scene segmentation.

Chapter 5

Scene Annotation

This chapter addresses research questions (3) Can we automatically segment a story into scenes? and (4) Can VDL be used to help in scene segmentation? that are described in section 3.1. As described in the methodology section 3.2, in order to develop a system to automatically segment narrative text into scenes, we need to gather a manually annotated corpus to train and evaluate the automatic system. To gather the manually annotated corpus, a set of guidelines was established based on sceneML that is described in detail in section 2.3.2. This chapter describes the details of this process and provides a detailed description of the resulting annotated corpus. This chapter is structured as follows: section 5.1 describes the annotation guidelines and the process of how it was developed. Section 5.2 describes the process of annotation that was carried out. Section 5.3 introduces the resulting annotated corpus and all the information about it.

5.1 Developing Scene Annotation Guidelines

Initial guidelines were derived directly from the specification of SceneML as given in that paper. The next step was then to run a pilot annotation exercise to test how well annotators could manage with these guidelines. This is described in 5.1.1. The final guidelines emerged from analysis of the pilot results together with some further work on annotating Project Gutenberg texts by my supervisor and me. These final guidelines are discussed in 5.1.2.

SceneML framework has been explained in detail in the guidelines document (see appendix A). It contains instructions on how begin the annotation task, and how to annotate each entity along with examples annotation.

The guidelines document introduced a new relation type between SDSs called `is in the same scene as`. This new relation was added to link between SDSs that belong to the same scene, as the annotation tool used to annotate the text does not support abstract entity annotation.

In addition, nonscene segments were also introduced in the guidelines document. This section of the document explains parts of text that should not be annotated as scenes. So far three types of a non-scene segment have been identified; general philosophising or opinion segments, background information segments, and narrative summary or narrative catchup.

A new element has also been added to the scheme *Scene Transitioning Element*, typically short phrases or a sentence or two that indicate a character is moving from one scene to another. For example: “and soon I emerged back into the corridor looking like a new man”. to tackle this issue.

5.1.1 Pilot Study

A small-scale pilot study was carried out to investigate how well-defined our definitions and annotation framework are with respect to scene boundary identification [1].

Methods

Two chapters (chapters 3 and 4) of “Bunnies from the Future” (Corcoran, 2016), a children’s story for reading ages 10-13, were selected to be annotated. Three annotators, distinct from the author, were identified (postgraduate students, non-native speakers of English) in addition to one of the author. They were given annotation guidelines based on the framework explained in section 2.3.2 and were instructed to use the Brat[2] annotation tool to annotate the two chapters following the guidelines, with two simplifying exceptions: (1) annotators

[1] Much of the material in this section was published in Text2Story 2021.
[2] http://brat.nlplab.org/

were asked not to annotate the `scene` abstract discourse element; (2) they were asked *not* to annotate explicitly any of the 4 relation types mentioned in the original SceneML paper which are:

- `sequence` links are assigned when one scene follows on from another.
- `analepsis` occurs when there is a flashback in the scene e.g. memory of the past
- `prolepsis` (or flashforward) occurs when then we are taken forward in time
- `concurrent links` are assigned between two scenes when the transition happens because there is another thread of the story happening at the same time so the transition take us to different characters and different place but the same time.

The first of these simplifications was imposed because the Brat annotation tool, which is a very easy-to-use tool for annotation text spans, does not support the creation of zero-span annotations. That functionality is necessary for creating abstract discourse elements, which can then be linked to spans in the text. We could not find any tool that would allow this and was equally easy-to-use, and did not have the resources to create our own. The problem can be circumvented by linking all SDSs in one scene together with a "same-scene-as" relational link, thus extensionally specifying a scene as the set of its SDSs.

The second simplification was imposed for several reasons.

1. First, regarding narrative progression links, for this first annotation exercise we were primarily concerned with determining whether or not annotators could accurately identify and agree on SDS boundaries. The problem of determining what the progression narrative links between scenes are is very much a secondary problem compared to this. As it turns out, all but one of the scenes in the chosen texts consist of a single SDS and follow each other in temporal sequence. Single-SDS scenes are likely a characteristic of this level of children's stories and one reason why we chose such stories for this pilot, though we did not deliberately chose chapters because they had few scenes with multiple SDSs.

Table 5.1 Statistics about the annotations. *SDS, Character, Time* and *Location* columns refer to the number of segments marked as each entity type per chapter for each annotator. Averages of these numbers by chapter/by *SDS* are also shown.

	Chapter 3				**Chapter 4**			
	SDS	**Char**	**Time**	**Loc**	**SDS**	**Char**	**Time**	**Loc**
Ann1	4	19	2	5	5	14	4	6
Ann2	5	21	3	3	6	28	4	6
Ann3	13	30	8	12	10	28	4	10
Ann4	8	21	0	5	9	19	1	6
Av/Chapter	7.5	22.75	3.25	6.25	7.5	22.25	3.25	7.25
Av/SDS	–	3.67	0.4	0.78	–	3.33	0.55	1.11

2. Second, regarding relational links between SDSs and the scenes they comprise, given that virtually all scenes consist of a single SDS, the need to link same-scene SDSs is low. This led us to ignore it altogether for this exercise, though clearly it will need to be addressed when dealing with more complex narrative structure.

3. Finally, regarding relational links connecting times, characters, locations to the scenes in which they participate, given the absence of explicit scene elements in the annotation, the times, characters and locations needed to be annotated in each SDS participating in a scene. This has the disadvantage of forcing the annotator to annotate these things multiple times per scene, when there are multiple SDSs per scene, but means that (1) the relations linking the scene elements time, character and location to the scene can be inferred from simple presence of annotated strings of these types within an SDS.

Figure 5.1 shows an example annotation from chapter 4 of Bunnies from the Future; entity segments are highlighted.

Results

Tables 5.1, 5.2 and 5.3 show the results of the annotation exercise. Table 5.1 shows information on the numbers of entity mentions annotated by each annotator and the averages of these numbers. Note that the chapters differ in size: chapter 3 is 124 sentences long (2756

Fig. 5.1 Screenshot showing an example of the pilot annotation using Brat.

words) and chapter 4 is only 65 (1775 words). The columns *SDS, Character, Location* and *Time* indicate the number of entity mentions annotated for each category by each annotator. Two averages are computed: (1) entity mentions per chapter averaged over annotators (2) average entity mentions per *SDS* for each annotator averaged across all annotators.

Table 5.2 shows inter-annotator agreement results. Pair-wise inter-annotator agreement results are computed using Cohen's kappa (Cohen, 1960). Averaged inter-annotator agreement results between all annotators are computed using Fleiss's kappa (Fleiss, 1971). $\kappa 1$ refers to the kappa score for segments of type *SDS*. For SDSs the kappa score is computed by considering each sentence as a potential candidate for a scene segment boundary. So,

Table 5.2 Inter-annotator agreement results. $\kappa 1$ refers to the kappa score for *SDS*, $\kappa 2$ refers to kappa score for all other entities together.

	Ann2				Ann3				Ann4			
Ann1	Ch3		Ch4		Ch3		Ch4		Ch3		Ch4	
	$\kappa 1$	$\kappa 2$	$\kappa 1$	$\kappa 2$	$\kappa 1$	$\kappa 2$	$\kappa 1$	$\kappa 2$	$\kappa 1$	$\kappa 2$	$\kappa 1$	$\kappa 2$
Ann1	0.60	0.54	0.33	0.27	0.15	0.27	0.20	0.30	0.23	0.19	0.10	0.21
Ann2					0.27	0.20	0.42	0.39	0.19	0.24	0.35	0.30
Ann3									**0.72**	0.33	**0.95**	0.52
Average $\kappa 1$ Ch3: 0.36							Average $\kappa 2$ Ch3: 0.29					
Average $\kappa 1$ Ch4: 0.41							Average $\kappa 2$ Ch4: 0.34					

Table 5.3 Percentage agreement results between annotator pairs for each entity type by token. Here *O* refers to the *Outside* tag.

	Ann2				Ann3				Ann4			
	Character	**Location**	**O**	**Time**	**Character**	**Location**	**O**	**Time**	**Character**	**Location**	**O**	**Time**
Ann1	0.24	0.15	0.99	0.8	0.28	0.15	0.99	0.26	0.33	0.05	0.99	0
Ann2					0.36	0.09	0.99	0.33	0.48	0.05	0.98	**1**
Ann3									**0.73**	**0.88**	0.98	**1**

each sentence is represented by either a 1 or 0, 1 if the sentence either contains a scene segment boundary or is preceded or followed by one, 0 otherwise. $\kappa 2$ refers to the kappa score computed for all other entity types (*Time, Character* and *Location*). Here we treat the problem of recognising these three entity types as a token classification problem, following the widely used named entity recognition approach of IOB tagging (see section 2.5.1). Each word is tagged as either *Time, Character, Location* or *Outside*, where the *Outside* tag is given to words that are not part of any entity mention. For simplicity we do not include a 'B' tag, as instances of contiguous distinct entity mentions of the same type are extremely rare.

Table 5.3 shows percentage agreement results between each annotator pair for each entity type. These are computed by creating a confusion matrix of token entity type labels (including type *Outside*) for each annotator pair and dividing each value on the diagonal in this confusion matrix by the corresponding row total.

Discussion

Here we discuss disagreement between the annotators with a view to determining how our annotation guidelines and/or processes should be improved and whether or not there are any underlying conceptual problems with our approach.

In general, the Cohen's kappa scores show what is commonly interpreted as fair (0.21–0.40) to moderate (0.41–0.60) agreement with a few cases of slight and substantial agreement around the edges. However, as the qualitative interpretation of kappa scores is contentious, we use these scores primarily as a diagnostic tool, highlighting areas of relative agreement and disagreement. Looking the scores overall, several observations can be made. First, the annotator pair (Ann3,Ann4) agree much more than any other annotator pair. Second, generally and on average, $\kappa 1$ scores are higher than $\kappa 2$ scores, i.e. agreement on SDSs is higher than agreement on entities.

Regarding the percentage agreement results on entity annotation in Table 5.3, it can be seen that for most cases, *Character* and *Time* entities have higher agreement results than *Location* entities (agreement for *Outside* tags will always be high given the unbalanced nature of the data, i.e. most tokens are outside of any entity mention). Note that these figures are heavily dependent on agreement in SDS annotation. If one annotator annotates two SDSs where another annotates just one, the first annotator will have twice the number of time, location and character annotations, since these entity types are to be annotated for each SDS. Hence low scores are to expected where agreement in the number of SDSs is not high. Indeed we can see that for annotators 3 and 4, where SDS agreement is higher than for any other annotator pair, agreement on entities is also very much higher than for any other annotator pairing.

Analysis of the annotations reveals two underlying causes of the observed disagreement: (1) lack of understanding of the guidelines and task and (2) lack of clarity or specificity in the guidelines. These are often not easy to distinguish.

Lack of understanding

It emerged in questioning following the annotation exercise that some annotators (Ann1 and Ann2) relied only on the author's verbal explanation of the guidelines and task and had either not read the guidelines at all or had not read them carefully. E.g., in some places we find two distinct *location* entities tagged in the same *SDS*, in clear contradiction of the guidelines, suggesting either the annotator did not pay enough attention or simply did not understand. Although the annotators were all good speakers of English and studying at the postgraduate level in well-respected English-speaking universities, being a non-native speaker of English led to misinterpretations of some sentences or expressions in the text that obviously caused mistakes in annotation. For example, one annotator labelled *earth* as a scene *location* in the idiomatic expression *how on earth had he ...* where clearly it is not. In another case, in the sentence *Sorry, old chap, had an attack of the wobbles. Dashed embarrassing*, the word *Dashed* was annotated as a character.

Lack of clarity and detail in the guidelines

Some annotators included definite articles in the annotated entity mention, e.g. *the stone ages* was annotated as a time by one annotator and *stone ages* by another (to assess the effect of such minor variation in annotation we re-measured inter-annotator agreement after removing all stop words and found that results slightly improved). Some *time* entities were annotated as the time of a scene while in fact they just reference other times. E.g., in *Do you not have good fabrics in the future?* the word *future* was annotated as the time of the scene. Regarding tagging characters, confusion arose as to whether to annotate the fullest form of the character mention, the first mention, or every mention. These small divergences pointed to the need to more fully and explicitly address them in the guidelines.

Two deeper issues were detected with respect to scene boundary determination and account for most of the variation between annotators with respect to SDS boundary placement. One is to do with "scene transition segments", typically short phrases or a sentence or two that indicate a character is moving from one scene to another. For example: *and soon I emerged back into the corridor looking like a new man*. Should this text be annotated as

belonging to the preceding or succeeding scene? Or is it a scene in its own right? Or part of no scene, but a new "scene transition" element that should be added to the annotation scheme? The other issue is to do with granularity of scene segmentation. E.g. if one minor character leaves a scene does this imply a new scene, given our definition of scene as "a unit of a story in which the elements of time, location, and main characters are constant"? Or should this be viewed as too minor a change to count as a scene change?

Again, if, for example, a character goes into a dressing room off another space and his changing clothes in the dressing room is described while he continues talking to another character outside the dressing room (Corcoran (2016), Chp. 4), is this a significant enough change of location to constitute a scene change? These edge cases highlighted the need to refine and extend the guidelines to take them into account.

5.1.2 Revised Scene Annotation Guidelines

The first pilot study was carried out to investigate how well-defined our definitions and annotation framework are with respect to scene boundary identification. Analysis of the annotations in that study revealed several causes of observed disagreement: (1) lack of understanding of the guidelines and task, (2) lack of clarity or specificity in the guidelines, (3) failure of non-native English speakers to fully grasp the meaning of certain expressions (e.g. idioms). Before proceeding with another exercise, problem (2) "lack of clarity and specificity in the guidelines" needs to be addressed to cover the issues identified by the pilot study, so we produced a new version of the guidelines. Because the pilot highlighted the fact that agreement on time/place/character annotation is dependent on prior agreement on SDS boundaries we decided to focus on that part of the task only for now. Since we wanted to move to more complex forms of narrative than children's stories we decided to trial some annotation on novels from Project Gutenberg. My supervisor and I then did just SDS and scene annotation on several novels of different periods and styles. As a result, several changes has been made to the annotation guideline, these changes include:

- Introducing a new relation type named `is in the same scene as` to connect SDSs within the same scene due to limitations in the annotation tool for abstract entity annotation.

- Adding **non-scene segments**, categorizing them into three types: general philosophizing or opinion segments, background information segments, and narrative summary or catchup segments. These segments are identified as portions of text that should not be annotated as scenes. The following is an example of a non-scene segment.

> *It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife. However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.*
> (Pride and Prejudice. p. 1)

- A notable addition to the scheme is the **Scene Transitioning Element (ST)** which encompasses short phrases or sentences indicating a character's movement from one scene to another. The following is an example of ST.

> *And so in ten minutes I had left my armchair and cheery sitting-room behind me, and was speeding eastward in a hansom on a strange errand, as it seemed to me at the time, though the future only could show how strange it was to be.*
> (Sherlock Holmes. p. 1)

- Instructing annotators to focus on annotating Scene Description Segments (SDSs) and Scene Transitioning Elements (STs), while omitting the annotation of other entities such as time, place, and character.

The updated guideline document can be found in Appendix A.

5.2 The Annotation Process

Having arrived at a revised set of guidelines and a slightly revised annotation task, we were then in a position to carry out another round of annotation. In this section we describe

the process of annotation we carried out and in the following section the results of that annotation.

In addition to the problems with clarity and specificity of the guidelines, as discussed above, the pilot revealed problems with lack of understanding of the guidelines and task, and failures of non-native speakers to grasp the meaning of certain expressions. In order to overcome these difficulties, (1) we recruited of native English-speaking annotators with sensitivity to text analysis, (2) we carried out various training sessions to ensure that the annotators fully understand the task and to clear up as many misunderstandings as we can.

5.2.1 Recruiting annotators

For the first pilot study, three annotators, distinct from the authors, were identified (postgraduate students, non-native speakers of English) in addition to one of the authors. They were given annotation guidelines based on sceneML and were instructed to use the Brat annotation tool to annotate the selected text following the guidelines. Various disagreements arose due to some misunderstanding of English (e.g. idioms) as discussed above in 5.1.1. To overcome this problem, we decided to recruit only annotators who were native speakers of English. After a long process of trying to assemble a team of four annotators from amongst the PhD students in the School of English at the University of Sheffield[3], we ended up with two annotators, one a PhD student from the School of English and the other a PhD student from the Centre for Doctoral Training in Speech and Language Technology in the Department of Computer Science.

5.2.2 Annotator Training

A PowerPoint presentation was prepared to explain the task in detail, the guidelines in brief and to give some annotation examples. A small annotation task on part of chapter 2 of “Bunnies from the Future” , was demonstrated by me. Then, annotators were asked to

[3]This process was seriously disrupted by the Covid-pandemic.

annotate the other part of the chapter. After that, I analysed their annotations and discussed any disagreement with the annotators.

When the training session was over, the annotators were asked to annotate two chapters (chapters 3 and 4) of "Bunnies from the Future" that had already been annotated by me, as an assessment of their understanding of the task and the guidelines. Inter-annotator agreement results were obtained, and the annotations were manually analysed to look for disagreements. Then the disagreements were discussed with the annotators to clear up any misunderstanding, or to gain more insights on improving the guidelines.

5.2.3 The Annotation Task

The annotation process was carried out through a web-based interface to a local instance of Brat Annotation Tool [4]. Annotators used swipe and click operations to annotate SDSs and STs, nonscene segments are implicit that anything not annotated is non-scene. Multiple SDSs that are part of the same scene were linked using the Brat relation annotation tool to signal that a `same-scene-as` relation holds between them. The annotated data is stored and made available in BRAT standoff annotation format.

The text were annotated by two annotators and saved in a separate text file. Both annotators' annotations are supplied with the corpus. Further annotations together with a consensus annotation may be made available in the future.

5.3 The ScANT Corpus

This section introduces the ScANT corpus, subsection 5.3.1 describes the sources of the corpus and how and why they were chosen. Subsection 5.3.2 provides statistical information about the corpus. Subsection 5.3.3 provides the inter-annotator agreement on each chapter of the ScANT corpus. Finally, subsection 5.3.4 discusses results of the inter-annotator agreement.

[4] https://brat.nlplab.org

5.3.1 Corpus Sources

The corpus consists of fourteen chapters from six different narrative sources (as one of the chapters is quite long it has been divided into three parts for analysis) , from children's stories and from out-of-copyright adult novels. The former were hypothesised as likely to have a simpler narrative structure and hence to be a good place to trial our approach; the latter as likely to possess more complex narrative structure and hence pose a more challenging test to our approach. The sources are:

1. *Bunnies from the Future*, a middle grade children's story by Joe Corcoran[5].The author has personally granted permission for us to release annotated chapters of this work.

2. *The Wonderful Wizard of Oz*, originally released as part of the Brown Corpus[6] and free for non-commercial purposes.

3. *Pride and Prejudice*, *A Tale of Two Cities*, *The Adventures of Sherlock Holmes* and *The Great Gatsby* from Project Gutenberg[7]. These are out of copyright in the US and UK and freely re-distributable subject to Project Gutenberg's terms and conditions.

The selection of novels was based on several criteria: (1) being popular in terms of number of downloads on Project Gutenberg, (2) covering a range of periods (1813 - 1925) and styles (full novels/short stories) and language (British and American English).

5.3.2 Corpus Statistics

Table 5.4 shows summary statistics for the ScANT corpus and associated annotations. Note that the relation between scenes and SDSs is largely one-to-one. With one exception this is always true for the children's stories, while there is somewhat more variation, suggesting more complex narrative form, in the adult novels.

The variation between annotators A1 and A2 is relatively small in terms of SDSs and Scenes. However, they are far apart regarding both scene transition segments and non-scene

[5]https://freekidsbooks.org/author/joe-corcoran/
[6]https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/packages/corpora/brown.zip
[7]https://www.gutenberg.org

Text	Sents	Words	SDSs		STs		Scenes		NSSs	
			A1	A2	A1	A2	A1	A2	A1	A2
Bunnies Ch3	124	2756	8	10	1	0	8	9	0	0
Bunnies Ch4	65	1775	10	8	0	0	9	7	0	0
Bunnies Ch5	173	3514	10	7	0	3	10	7	0	0
Bunnies Ch6	117	2911	10	10	0	6	10	10	0	0
WOZ CH2	132	2449	4	8	0	2	4	8	0	1
WOZ CH3	123	2361	9	8	1	7	9	8	1	1
Sherlock Holmes Ch1 P1	268	4200	11	10	0	4	11	10	17	0
Sherlock Holmes Ch1 P2	277	4784	23	11	1	6	20	11	0	0
Sherlock Holmes Ch1 P3	93	1333	8	6	0	5	8	6	2	0
Sherlock Holmes Ch6	561	10974	31	34	2	15	26	34	8	0
Pride and Prejudice Ch1	60	1018	1	3	0	2	1	3	6	0
Pride and Prejudice Ch3	86	1984	12	10	0	5	12	10	0	0
A Tale of Two Cities Ch1	19	1140	0	5	0	4	0	5	19	0
A Tale of Two Cities Ch3	73	1920	4	13	0	4	4	13	11	0
The Great Gatsby Ch1	337	7209	19	43	1	12	19	43	52	0
The Great Gatsby Ch3	288	5307	31	24	0	10	29	24	13	0
Total	2796	55635	191	210	6	85	180	208	129	2

Table 5.4 Summary Statistics for the ScANT corpus, showing for each annotated text the count of sentences and words and of SDSs, STs, Scenes and Non-scene sentences (NSSs) for each annotator (A1 and A2).

Chapter	SDS Median	N = 30%	N = 0	N = 1	N = 3	N = 5
Bunnies Chapter 3	7	0.79	0.74	0.74	0.79	0.79
Bunnies Chapter 4	5.5	0.60	0.60	0.60	0.60	0.60
Bunnies Chapter 5	14.5	0.45	0.29	0.29	0.45	0.45
Bunnies Chapter 6	5.75	0.68	0.47	0.47	0.72	0.76
WOZ Chapter 2	19.75	0.47	0.11	0.11	0.27	0.41
WOZ Chapter 3	11	0.77	0.57	0.57	0.72	0.77
Sherlock Holmes Chapter1 P1	10.5	0.16	0.07	0.07	0.16	0.16
Sherlock Holmes Chapter1 P2	8	0.53	0.42	0.42	0.53	0.53
Sherlock Holmes Chapter1 P3	8.75	0.75	0.75	0.75	0.75	0.75
Sherlock Holmes Chapter 6	8.5	0.37	0.25	0.25	0.37	0.40
Pride and Prejudice Chapter 1	16	0.65	0.65	0.65	0.65	0.65
Pride and Prejudice Chapter 3	6.25	0.65	0.43	0.43	0.65	0.65
Tale of Two Cities Chapter 3	6.5	0.21	0.01	0.01	0.30	0.39
The Great Gatsby Chapter 1	7.5	0.34	0.22	0.22	0.34	0.39
The Great Gatsby Chapter 3	6	0.49	0.38	0.38	0.58	0.66
Average	8.94	0.53	0.40	0.40	0.53	0.56

Table 5.5 IAA Results, showing Cohen's kappa under varying degrees of leniency, where N indicates the number of sentences apart SDS boundaries may be to count as a match or, where $N = 30\%$, the number of sentences expressed as a percentage of the median SDS sentence length for that text.

segments. We discuss this further below in Section 5.3.4. First, however, we examine inter-annotator agreement regarding SDSs in more detail.

5.3.3 Inter-annotator Agreement

Table 5.5 shows inter-annotator agreement results for SDSs using Cohen's Kappa Cohen (1960). Kappa is computed as was discussed above in relation to the pilot study in section 5.1.1, each sentence is given a tag, 1 for sentences on the boundary of an SDS (either beginning or end) and 0 otherwise. Boundaries of STs are ignored as it is clear the two annotators' understanding of the task is so different that precise quantitative analysis is not merited.

Aside from calculating only exact matches as agreement ($N = 0$ in Table 5.5) we also investigated a more lenient approach to calculate the agreement in which annotators are

deemed to agree if they place a sentence boundary within *N* sentences. This was prompted by the observation that in many cases annotators seemed to be placing SDS boundaries relatively close to each other, but not exactly in the same place. Kappa scores have been calculated for various *N* sizes: $N = 30\%$ of the median SDS length in sentences in each chapter, *N=1*, *N=3* and *N=5*. By the median SDS length, what we mean is that we combined the length of SDSs for both annotators in a single list and then we took the median of this. Then , 30% of the given median is calculated. We have omitted one chapter from Table 5.5 – *A Tale of Two Cities*, chapter 1, because one annotator believed it contained nothing but non-scene segments, while the other thought it contained 5 scenes. This gave a kappa score of 0, which skewed the rest of the results.

5.3.4 Discussion

Regarding differences between our annotators, it is clear that the two annotators have a different conception of what STs and NSSs are. This is probably due to the fact that the children's stories which we used as training materials contain very few of either of these, particulary of NSSs (in fact this appears to be an interesting difference between children's and adult narrative). For example, you can see that *Sherlock Holmes Ch1 P1* and *The Great Gatsby Ch3* scored the lowest agreement between the annotators. This may be due to the fact that *Sherlock Holmes Ch1 P1* was the first text provided to the annotators after the children's story *Bunnies From The Future*. As a result, the annotators might have found it difficult to apply the annotation guidelines to a more complex text without sufficient training. Additionally, the early chapters of novels often contain more non-scene segments than later chapters. As mentioned earlier, the two annotators may have different interpretations of these segments.

Any future annotation effort should ensure that these concepts are more clearly understood by annotators. On examination our view is that A1 has followed the guidelines much more closely regarding STs and NSSs and therefore, if one is to train or test a classifier on these materials, our recommendation would be to use the A1 annotations only. However recent

work such as that reported in Uma et al. (2022) highlights the potential value of learning from disagreement, so we have included both sets of annotations in the corpus.

Concerning the kappa scores for SDS agreement, they fall in the range that has been interpreted as "fair" or "fair to good". However, kappa scores are known to be lower when there are fewer labels (just two in our case) and where the labels are not equiprobably occurring (also true in our case, since 0 labels are much more frequent than 1's) so results should be viewed in this light (Wikipedia, 2023). Percentage agreement scores for SDS annotations are around 90%. Note that kappa scores rise significantly if we are prepared to allow some leniency in terms of non-exact matching. How legitimate this is needs further examination to determine whether the improvement is a reflection of genuine uncertainty about the precise boundary between what the annotators clearly agree are distinct scenes or whether it is the result of conflating separate scenes, according to the different annotators' perceptions.

While the corpus is too small to start making generalisations about stylistic differences between different authors, it is worth noting that the amount of non-scene content in the adult novels (6.21% of the total sentences if we accept A1's NSS annotations, which we believe are more accurate) is vastly greater than that in the children's stories (0.14%), suggesting that much beyond simple event narration goes on in adult fiction.

5.4 Summary

In this chapter we partly addressed **RQ3**. In order to build a system that can automatically segment a story into scenes, first we need a dataset annotated into scenes. This chapter introduced a set of guidelines based on sceneML. First, a pilot study was carried out to asses the viability of our guidelines, and to investigate how good the guidelines were. Inter-annotator agreement of the pilot study was analysed, and the guidelines were updated according to that. After the guidelines were updated, the annotation process started with recruiting two PhD students at the University of Sheffield whom are native speakers of English. They were given a training session and a small annotation exercise to clarify any

misunderstanding of the task or the guidelines. Then, the annotation task was carried out. The outcome of this annotation task is the first dataset of English narrative texts annotated for scene boundaries in compliance with SceneML and a set of annotation guidelines to be used in future annotation tasks. The dataset consists of fourteen chapters of novels and children's stories annotated for scene description segments and scene transitions segments as defined in SceneML. The sources are: *Bunnies from the Future, The Wonderful Wizard of Oz, Pride and Prejudice, A Tale of Two Cities, The Adventures of Sherlock Holmes*, and *The Great Gatsby* . A total of almost 200 scenes have been annotated, with a total of 2796 sentences and 55635 words. Inter-annotator agreement results were computed using Cohen's Kappa score. Our obtained Kappa scores fall in the range that has been interpreted as "fair" or "fair to good".

The following chapter investigates automating the task of scene boundary detection and to ascertain the sufficiency of ScANT for this task through training a model on the corpus to.

Chapter 6

Automatic Scene Segmentation

The previous chapter described how we gathered a corpus of narrative texts and segmented them into scenes according to our SceneML-inspired guidelines. Given this resource we can now turn to training and evaluating models to automatically segment narrative texts into scenes in order to answer research questions (3) Can we automatically segment a story into scenes? and (4) Can VDL be used to help in scene segmentation? as posed in Chapter 3.
This Chapter is structured as follows: section 6.1 describes the details of the dataset used in this chapter. Section 6.2 describes the experimental setup to train and test models for the task of automatic segmentation of narrative text into scenes. Section 6.3 provides results of the experiments and discussion about them.

6.1 Data

As explained in the previous chapter, the dataset is composed of chosen chapters from children's stories and adult novels that are no longer protected by copyright. We specifically selected children's stories with the expectation that they would exhibit a simpler narrative structure, providing an ideal testing ground for our approach. Conversely, we included adult novels to incorporate more complex narratives, posing a greater challenge for our approach. There were three dataset sources. The first was 'Bunnies from the Future', a middle-grade children's story authored by Joe Corcoran. The second source was 'The Wonderful Wizard

of Oz', originally part of the Brown Corpus. Finally, the third source comprised 'Pride and Prejudice', 'A Tale of Two Cities', 'The Adventures of Sherlock Holmes' and 'The Great Gatsby' obtained from Project Gutenberg. The dataset has 2,796 sentences, 55,635 words and 191 SDSs according to A1, as on inspection we believe that A1 better understood the guidelines.

6.2 Experimental Setup

To build a model that can automatically segment narrative text into scenes (SDSs) using machine learning, first, we need to train the model using training data. To make the problem easier for automatic scene segmentation, we treated the task as a sentence classification problem instead of text segmentation, where each sentence is given a tag (i.e. 1 is designated for sentences on the boundary of an SDS, either at the beginning or the end, and 0 otherwise). Scene Transition Segments are not considered here as their numbers in the annotated data were very small compared to the number of annotated SDSs.

The machine-learning models were trained using the training data and evaluated with the testing data. To ensure robust evaluation, stratified 10-fold cross-validation was implemented using the scikit-learn library. This technique splits the data into 10 equally sized folds while preserving the class distribution, allowing the models to be trained and tested on each fold independently. This approach helps in obtaining reliable performance estimates for the models. Notably, the data were not shuffled to preserve sentence order. The following section presents and explains the machine-learning models used for the task.

Three machine-learning models were trained and tested on the annotated data. Then, we compared the models' performances to determine which model is optimal for our task. The following subsections provide a brief description of each of the models.

6.2.1 Model 1 - The Conditional Random Field (CRF) Model

In the first model, we treated the problem as a sequence-labelling problem, where the order of sentences is significant and the wider textual context of the sentence being labelled is

important. Herein, the sequence refers to the ordered sentences of each chapter and their corresponding tags. A CRF model (see section 2.4.2) was trained on the training data. For this endeavour, we first extracted the following features:

- Transitioning phrases: This is a binary feature, where if the sentence contains a transitioning phrase, the feature is given a tag of *1* and *0* otherwise. This feature aims to identify transitions between different segments within the text. Transitioning words/phrases (e.g. *later on*, *after*, etc.) are hypothesised to appear more in sentences on the boundaries of a scene. The list of transitioning phrases is presented in Table. 6.1.

- Beginning or end of a paragraph: This is also a binary feature, where if the sentence occurs at the beginning or end of a paragraph, the feature is given a tag of *1* and *0* otherwise. This feature aims to capture paragraph-level patterns that might influence the classification of the current sentence.

- End of a chapter (true/false): This binary feature denotes whether the current sentence occurs at the end of a chapter, as the end of a chapter usually indicates the end of a scene.

- Part-of-speech (POS) tags: Incorporating the part-of-speech tags of each word in the current sentence being classified was carried out using spaCy. In addition, POS tags were extracted for the two preceding sentences and the two following sentences.

- Named entity: Each word in the sentence being classified was given a BIO tag. The Named Entity Recognition (NER) function used was implemented by spaCy, using the NER model `en_core_web_md`. Named entities can include names of people, organisations, locations or other specific entities. In addition, the words of the two preceding sentences and the two following sentences were also given named entity tags.

- Contextual information (2 sentences before and after): This feature considers the two sentences preceding and the two sentences following the current sentence. By

incorporating neighbouring sentences, the model can capture contextual dependencies and the influence of surrounding information on the classification of the current sentence. This information presented to the model as a set of features. The same set of features extracted from the test sentence is also extracted from the two preceding sentences and the two following.

- Visually descriptive language (VDL): Visually descriptive information from Chapter 4 is used here as a feature. The classifier developed in Chapter 4 was used to classify sentences as (*0, 1,* or *2*), where:
  - *0* tag: not visually descriptive
  - *1* tag: visually descriptive
  - *2* tag: partially visually descriptive

  To assess the effectiveness of the VDL feature on the performance of the CRF classifier, the model was tested twice—once with the VDL feature added to the list of features and once without.

Table 6.2 presents the results of the two versions of the CRF model. Section 6.3 summarises and compares the performance results of all models.

Next	Afterward	Finally	Later
Last	Lastly	At last	Now
Subsequently	Then	When	Soon
Thereafter	After a short time	The next week	A minute later
In the meantime	Meanwhile	On the following day	At length
Ultimately	Presently	Above	Behind
Below	Beyond	Here	There
To the right (left)	Nearby	Opposite	On the other side
In the background	Directly ahead	Along the wall	As you turn right
At the tip	Across the hall	At this point	Adjacent to

Table 6.1 Transitioning Phrases (Miami Dade College, n.d.)

6.2.2 Model 2 - Bidirectional Encoder Representations from Transformers (BERT)

The second model developed is a deep-learning model that uses a pre-trained language model. The ktrain library (Maiya, 2020) was utilised to implement the model, with the use of BERT (Devlin et al., 2018) from Hugging Face transformers. Ktrain is a Python library that serves as a wrapper for the deep-learning library (TensorFlow Keras) and other libraries to help in building, training, evaluating and deploying deep-learning models. It offers an easy-to-use API that enables researchers to easily build and experiment with ML models without requiring extensive deep-learning expertise. In addition, Hugging Face is an open-source library that consists of pre-trained transformer models available to the research community (Wolf et al., 2019).

Two experiments were conducted on the model to explore the most effective implementation: one with BERT cased and one with BERT uncased. BERT can read input up to 512 tokens at a time – this size can be adjusted during fine tuning. Any sentence less than the maximum length specified is padded, and any sentence larger than the maximum length is truncated. BERT is pre-trained on a large amount of text. It learns its model through two unsupervised tasks: masked language model and next-sentence prediction. Then, it only needs to be fine-tuned for our specific task.

In order to use BERT for our task, it has to be fine-tuned – by exposing the model to our labelled data to learn the relevant parameters. The entire input sequence is represented as a sentence embedding, where a special token `[CLS]` is inserted at the beginning of the input sequence of sentences. The input to BERT is tokenised into subwords and the special tokens `[CLS]` and `[SEP]` are added; this is shown in figure 6.1. The output vector of the final layer for the `[CLS]` is then passed to the classifier head (a dense layer followed by a softmax activation function in our case) that provides the predictions (0s or 1s). To map y_{CLS}, that is, the output vector of `[CLS]`, to a set of scores to the possible output class, a set of weights W_C needs to be learned. Learning the weights (fine-tuning) is done by training the model on the training labelled data. During training, `cross-entropy` loss between the true label and the

output of the softmax function that is the predicted label and the true label is used to learn the weights W_C. So, the classification is done by multiplying the weights W_C, by the vector y_{CLS}, and providing this to the `softmax` function (Jurafsky and Martin, 2023).

$$y = softmax(y_{CLS}W_C)$$

The model was fine tuned using a learning rate of 1.44E-05, with 3 epochs, and maximum length of 128 for `Bert-Cased`. And 5 epochs, and maximum length of 256 for `Bert-Uncased`.

$$[s_1, s_2, \ldots, s_n]$$
$$\downarrow$$
$$[CLS]\ w_{s_1 1}, \ldots, w_{s_1 n}\ [SEP]\ \ldots$$

Fig. 6.1 Representation of a sequence input to BERT.

6.2.3 Model 3 - Sentence Pair Classification with BERT

In an attempt to capture as much context as possible, the task of scene segmentation was treated as a sentence pair classification task, where the relationship between a current sentence and its surrounding context is assessed. The input consists of a pair: the first-pair part is the current sentence and the second-pair part is the concatenated form of the two preceding and two following sentences. This allows the broader context surrounding the current sentence to be considered during classification (see figure 6.2).

Fine tuning BERT for sentence pair classification is fairly similar to the task of pre training BERT on next sentence prediction. To capture the information that the two inputs are actually a pair, the special token `[CLS]` is added to the beginning of the pair, and the token `[SEP]` is added between the two pair-parts and at the end of the last token of the pair. The input to BERT in this case is tokenised into subwords and the special tokens `[CLS]` and `[SEP]` are added, as illustrated in figure 6.3.

And, as explained in the previous subsection, during fine0tuning (training the pre-trained model on our labelled data and inferring the model's weights W_C), the output of the final layer of the `[CLS]` is then used to predict the label of the pairs (1 for SDS boundary, and 0 for a non-boundary sentence).

As with model 2, this model was implemented using the ktrain library. To explore different variations, two experiments were conducted using pre-trained BERT models from the Hugging Face model repository: BERT cased and BERT uncased. In leveraging these powerful language models and the ktrain library, the aim was to enhance the model's ability to capture and comprehend complex sentence relationships, thereby improving sentence pair classification performance.

$$[s_1, s_2, \ldots, s_N]$$
$$\downarrow$$
$$[(s_1, c_1), (s_2, c_2), \ldots, (s_N, c_N)]$$

$$\text{where } c_i = s_{i-2} + s_{i-1} + s_i + s_{i+1} + s_{i+2}$$

Fig. 6.2 Sentence pair input, where c refers to the concatenated form. Input to the classifier is formed by pairing each sentence in the original text with a context, which is the concatenation of the two preceding and two following sentences.

$$(s_i, c_i)$$
$$\downarrow$$
$$[CLS]\ w_{s_i 1}, \ldots, w_{s_i n}\ [SEP]\ w_{c_i 1}, \ldots, w_{c_i n}\ [SEP]$$

Fig. 6.3 Representation of a tokenised pair of inputs to BERT.

The model was fine tuned using a learning rate of 1.44E-05, with 10 epochs, and maximum length of 512 for `Bert-Cased` and `Bert-Uncased`.

6.3 Results and Analysis

Model	Acc	Balanced Acc	P(W)	R(W)	F1(W)	P(M)	R(M)	F1(M)	F1 class 0	F1 class 1
Bert Cased	0.92	0.58	0.88	0.89	0.87	0.64	0.58	0.59	0.94	0.24
Bert Uncased	0.92	0.56	0.87	0.89	0.88	0.61	0.56	0.57	0.94	0.20
Sent-Pair Cased	0.90	0.51	0.86	0.85	0.84	0.56	0.55	0.54	0.91	0.17
Sent-Pair Uncased	0.90	0.55	0.87	0.85	0.84	0.55	0.53	0.51	0.92	0.18
CRF(VDL)	0.90	0.52	0.85	0.88	0.85	0.59	0.52	0.52	0.93	0.12
CRF	0.87	0.52	0.86	0.89	0.86	0.64	0.52	0.53	0.91	0.12
MCC	0.92	0.50	0.84	0.92	0.88	0.46	0.50	0.48	0.96	0.00

Table 6.2 SDS boundary classification results, (W) refers to weighted average and (M) refers to macro averaging. P refers to Precision, R refers to Recall and Acc means accuracy. Tenfold-cross-validation was carried out on each of the models. The average of the results of the 10 folds is reported here. MCC refers to the most common class classifier (as a base line).

Table 6.2 presents the performance results of the six machine-learning models, namely, CRF, CRF(VDL), which refers to CRF models with the VDL feature added, BERT cased, BERT uncased, Sent-Pair Cased, which is a sentence pair classification with a BERT cased model, and Sent-Pair Uncased (with BERT base uncased). The models are evaluated using different metrics, including accuracy, balanced accuracy, precision (with both macro and weighted average), recall (with both macro and weighted average) and F1 (with both macro and weighted average and F1 score for each class 0 and 1 independently). Tenfold-cross-validation was used to test each of the six models.

The findings indicate that accuracy alone showed relatively high values across the models (ranging from 0.87 to 0.92). However, since the dataset is imbalanced (there are more 0 tags than 1 tags), accuracy alone is not sufficient to compare between models. We added other metrics that can give a better insight into the performance of models, such as balanced accuracy and F1 for each individual class.

As can be seen, most of the metrics used in testing the models yielded highly similar results, which made it difficult to determine which model performed best. However, if we focus on the two metrics that can be used to reflect the performance of models on imbalanced datasets, the balanced accuracy metric is often considered when one of the classes is a lot

larger than the other. The BERT cased model achieved the highest balanced accuracy of 0.58, indicating its ability to handle imbalanced data more than the other models.

In addition, we obtained the F1 score for each class and focused on the results for a minority class (class 1) that could help elicit an insight on which of the models would perform better in predicting class 1. Similarly, the BERT cased model scored the highest, with a 0.24 F1 score.

Notably, although the BERT cased model achieved the highest scores among all models in terms of balanced accuracy and F1 (for class 1), it is difficult to derive a conclusion as the results for the majority of the models are highly similar. To see whether the differences in our models' performances are significant, we conducted a statistical test on the results, as reported in the following section.

Statistical Analysis

Model Comparison	**Sent Pair-BERT Uncased**	**Normal-BERT Cased**	**NormL-BERT Uncased**	**CRF-VDL**	**CRF**
Sent Pair-Bert Cased	0.3847 | 0.4048	0.1859 | 0.1402	0.3640 | 0.4043	0.7337 | 0.6488	0.8501 | 0.6770
Sent Pair-Bert Uncased	-	0.2123 | 0.2890	0.7336 | 0.8203	0.1212 | 0.1035	0.1859 | 0.1402
Normal-Bert Cased		-	0.6230 | 0.5449	**0.0211 | 0.0309**	**0.0257 | 0.0341**
NormL-Bert Uncased			-	0.1211 | 0.1397	0.2729 | 0.1617
CRF-VDL				-	1.0 | 0.7896

Table 6.3 Pairwise Mann–Whitney p-value results on the models' performance. For each pair, the p-value was calculated on the balanced accuracy results and on F1 for the minority class. The results were reported as BA | F1 in each case.

Comparison	**Sent Pair-Bert Cased**	**Sent Pair-Bert Uncased**	**Normal-Bert Cased**	**Normal-Bert Uncased**	**CRF-VDL**	**CRF**
MCC	0.1153	0.0014	6.39e-05	0.0014	0.0426	0.1153

Table 6.4 Mann-Whitney Test Results for Balanced Accuracy

A statistical analysis was conducted to analyse whether there is a significant difference in the performance of the six models. Our null hypothesis is that there is no significant difference in the performance of the six models that were used for scene boundary detection. Of the 10 metrics presented in Table 6.2, balanced accuracy and F1 for class 1 are the two metrics chosen to do the statistical analysis on. The metric values for each of the 10 folds were used as the data samples for the significance test. The Mann–Whitney U test

(MacFarland et al., 2016) was then conducted on these performance metrics of the six models. Mann–Whitney test is non-parametric text that does not require normally distributed data and works well with small data sizes which is the case in our task (10 BA and 10 F1 scores for each classifier) and size 10 could be considered relatively small. In general, as shown in Table 6.3, the results showed no significant difference (p-value> 0.05) in the performance of the models. In terms of balanced accuracy metric, the p-values ranged from 0.1859 to 0.7336, suggesting no significant difference in the performances of BERT cased, BERT uncased, sentence pair with BERT cased and sentence pair with BERT uncased. This is also the case for the results in terms of F1 for class 1. The p-values for the model comparisons ranged from 0.1402 to 0.8203, suggesting no significant difference in the performances of BERT cased, BERT uncased, sentence pair with BERT cased and sentence pair with BERT uncased. On the other hand, the p-values for both metrics (balanced accuracy and F1 for class 1) showed a significant difference (p-value < 0.05) in the performance of BERT cased compared with CRF and CRF with the VDL feature.

In addition, another statistical analysis was also conducted to analyse whether there is a significant difference in the performance of the six models compared to the MCC baseline. Table 6.4 shows Mann-Whitney p-value results between the most common class (MCC) classifier and each of the six models. These p-values were obtained for 10 folds BA, as the F1 for the minority class will be all 0s for the MCC. The results show that, there is a significant difference in the performance between the MCC and sentence pair with Bert Uncased, Bert Cased, and Bert Uncased. There is marginally significant evidence of a difference with CRF-VDL. And finally, there is no strong evidence of a difference with sentence pair with Bert Cased and CRF.

Discussion

The stronger performance of BERT could be attributed to the fact that BERT is pre-trained on the BookCorpus collected by (Zhu et al., 2015). The BookCorpus is made of 11,038 novels from 16 different genres (e.g. romance, science fiction, fantasy, etc.). Therefore, BERT has seen narrative text previously.

Overall, the findings suggest that the choice of model (BERT cased, BERT uncased, sentence pair with BERT cased and sentence pair with BERT uncased) may not significantly impact performance in our tasks. Users can select the model that best aligns with their specific requirements or preferences without compromising performance.

However, there is no significant difference both between the CRF models and the some of BERT models and between the two CRF models. This could suggest (1) the power of language models pretrained on large amounts of text and then fine-tuned for the task outweighs the use of features engineered for this specific task but then trained on a small amount of labelled data (2) VDL either offers no help for this task or the accuracy level of the VDL classification is too low to be useful here.

Finally, comparing the performance of our models to those of Zehe et al. (2021) on the binary scene segmentation task they define, we see that our results are broadly similar (.24 F1 measure). Given differences in task definition and dataset not to much should be made of this without further investigation. However, both their efforts and ours suggest this is a hard task.

6.4 Summary

This chapter introduced the automatic scene segmentation task that segments narrative text into scenes. We described the data used to train the models. We also experimented with three different machine-learning models using textual features and a deep learning approach. Then, we evaluated the performance of these models and conducted a statistical analysis using the Mann–Whitney test to determine if there are significant differences among these models in terms of performance. The results obtained showed no significant differences among the majority of the models. Notably, a significant difference was found between BERT cased and CRF.

Chapter 7

Conclusion and Future Work

In this chapter, we summarise the work conducted in this thesis. Section 7.1 gives a summary of the work conducted to address each of the research questions and draws tentative conclusions in relation to these questions. In section 7.2, we discuss the unaddressed issues that should be explored in future studies.

7.1 Summary of Contributions

To date, there has been great interest in the interdisciplinary area of vision and language. Significant research on vision and language tasks, such as image captioning, aligning books with movies, automatic text illustration and story picturing and automatic narrative generation, is being carried out. There is also a lot of research on computational understanding of narrative and narrative extraction from textual data.

The focus of the work in this thesis has been three-fold. First, we studied visually descriptive language and carried out experiments on this topic, conducted an analysis, and formulated conclusions. Second, we investigated the automatic segmentation of narrative text into scenes by choosing the most suitable definition of a scene, constructing a scene-annotated corpus of narrative texts and building scene segmentation models. Third, we investigated how useful visually descriptive language (VDL) is for scene segmentation, though in a very limited way.

We started by reviewing the definition of VDL in Chapter 2, looking into several possible definitions of scenes found in the literature and choosing the most suitable one. Next, we explained in detail relevant information about *SceneML*, the framework we used to develop our scene annotation guidelines. This was followed by a review of related work on scene annotation and scene segmentation. The work by Zehe et al. (2021) is the most relevant to ours. However, it differs in several respects, which are discussed in the chapter. To the best of our knowledge, there is no work on automatic scene annotation and segmentation based on SceneML.

In Chapter 3 we laid out the research questions we intended to address in the rest of the thesis. These are repeated below in our summary of work carried out in subsequent chapters of the thesis.

In chapter 4, we conducted experiments on the first half of the thesis on VDL. This chapter addressed two of our research questions. Section 4.1 addressed the research question:

RQ1: *How well can we automatically classify sentences into three classes, VDL, not VDL and partially VDL?*

We started by briefly describing the annotated dataset used to train the models. We then explained in detail the experiments carried out to determine the best model for the task of classifying sentences into the three classes, VDL, not VDL and partially VDL. We investigated three groups of features for this task: (1) explicit visualness representation that encodes whether the sentence has visual components – VerbNet and WordNet features were explored in this group (2) representations using word tokens – in this group of features two features were explored and tested, Tf-idf and word embedding (3) combinations of (1) and (2). The experiments were carried out using three different machine learning classifiers, kNN, SVM, and MNB. Our experiments showed that using word embeddings and an SVM classifier achieved the highest results with a balanced accuracy of 0.74.

Section 4.2 addressed the second research question:

RQ2: *Can we detect segments of sentences that are visually descriptive?*

In this section, we explained the experimental work for the task of segment-level classification of sentences into the three classes, VDL, not VDL and impure VDL.

We approached this task as a sequence labelling problem, where each word is assigned either an inside (I) or outside (O) tag. Our experimentation involved various lexical and morphological features, as well as features related to the word's position in the sentence. These features were combined with different context sizes. The results indicated that combining all the aforementioned features with a context size of one word before and one word after the target word, using a Conditional Random Field (CRF) classifier, yielded the highest F1 score of 0.81.

Chapter 5 and chapter 6 form the second half of this thesis on the automatic segmentation of narrative text into scenes. Chapter 5 partly addressed the third research question:

RQ3: *Can we automatically segment a story into scenes?*

One way to answer this question is to create a corpus of scene-annotated narrative texts for use in training and evaluating models of automatic scene segmentation. Chapter 5 addresses this challenge. We began by introducing our annotation guidelines that were developed based on SceneML, the framework for scene annotation introduced in Chapter 2. A pilot study on annotating scenes in narrative text based on the guidelines was carried out to assess the viability of the task and to test how good the guidelines were. A thorough analysis of the pilot annotation guidelines was performed, and changes were made accordingly. After improving the guidelines, we started the annotation process. We described in detail the process we followed and the inter-annotator agreement results. The outcome of this chapter was a set of annotation guidelines to be used in future annotation tasks and a dataset of annotated scenes in text[1]. To the best of our knowledge this is the first publicly available corpus of English narrative texts annotated with scene boundaries.

Chapter 6 addressed the technical aspect of the third research question. In this chapter, several experimental methods were carried out to build a model that can automatically segment a story into scenes. Several models were investigated and their performance assessed and compared to find the best model for the task. Three models were implemented and explored. The first model is feature-based model that treated the task as a sequence-labelling task, in which a CRF classifier is used with a set of handcrafted features. The second model

[1]The corpus and annotation guidelines are available at https://doi.org/10.15131/shef.data.21517908

treated the task as a sequence classification task. We addressed it using a pre-trained language model (BERT), and this was explored with two versions BERT cased and BERT uncased. The third model treated the task as a sentence pair classification task, in which the input was represented as a sequence of pairs. The fist item in the pair is the sentence to be classified and the second pair item is the concatenated form of the two preceding sentences and the two following ones.

This chapter also addressed the fourth research question:

RQ4: *Can VDL be used to help in scene segmentation?*

We tried to answer this research question by using our CRF feature-based model for scene segmentation and adding to it a feature dervied from the VDL model developed in chapter 4.

Looking into the balanced accuracy and F1 for the minority class, model 2 with BERT cased was the best performing model for the scene segmentation task with 0.58 balanced accuracy and 0.24 for F1. However, a statistical test using Mann-Whitney was carried out to compare models looking for any significant differences between the models. Comparing BERT based models (model 2 and 3), including their cased and uncased versions, revealed no significant differences. In addition, examining the two CRF models (with and without the VDL feature), showed that there was a significant difference between them and the BERT cased version of model 2. Finally, Mann-Whitney was also used to test for any significant difference between the performance of all models and the performance of the MCC baseline. The p-values showed that there was significant difference between the MCC and sentence pair with BERT Uncased, normal BERT Cased, and normal BERT Uncased models. There is no significant difference with sentence pair with BERT Cased. Interestingly, there was a significant difference (although marginal) between MCC and CRF-VDL while there was no significant difference between MCC and CRF.

While incorporating VDL into the task of scene segmentation in the manner we did so, did have a minor positive impact on raw accuracy scores, overall all it did not offer any significant improvement over the same model without the VDL feature. Whether this was due to inaccuracy in the identification of VDL or lack of utility of this information for this task remains unclear.

7.2 Future Work

This thesis contributes to knowledge about two main natural language processing (NLP) topics: (1) defining and identifying VDL in text and (2) scene definition and segmentation, both considered new tasks in the NLP field. Although we made progress in both tasks, there are still many areas that are open for future research opportunities.

Defining and identifying VDL in Text

While the work in this thesis provided a solution to the VDL sentence-level and segment-level classification tasks, there is room for improvement.

- **Improving VDL classification**

  Regarding the classification results, pre-trained language models (e.g. Bert) could be investigated to see if they could improve the classification results. In addition, more annotated data can also be gathered to give more accurate results for the classification task.

  In addition, the use of Large Language Models (LLMs) like GPT-3 (Brown et al., 2020) or GPT-4 (Achiam et al., 2023) could be considered to further enhance VDL identification task. This approach can be applied through zero-shot learning, which does not require specific training data, or few-shot learning, which only needs a few examples of the training data. Using LLMs could be effective since it only needs a few examples of the training data, and our dataset size is small.

  Furthermore, since the data is imbalanced other metrics, like GMR, ROC could be used to evaluate the model.

- **Using VDL for other tasks**

  VDL could be investigated for use in other NLP applications, e.g. in text/story illustration. Since illustrations are typically associated with elements of a text or story that can be visualized, a VDL-detector could be used to help suggest where illustrations should be placed in text.

Scene definition and segmentation

Automatic segmentation of narrative text into scenes is considered a new task in NLP, and there has to date only been limited research in this area. Following are some future directions that could be taken in this area:

- **Annotation guidelines**

  Although the annotation guidelines were refined many times in multiple cycles of pilot studies, they still require further refinement to cover instructions for a wider variety of narrative text types (e.g. biographies, historical and contemporary fiction for adults). In addition, it would be interesting to explore the application of our guidelines to the annotation of narrative text in languages other than English.

- **Scene annotated data**

  We tried to annotate as much data as possible, but as this is a PhD project with limited time and limited resources, the annotation process had to be stopped at a certain point. To build a supervised learning model that can more accurately automatically segment narrative text into scenes, more data is needed. The annotation process can be extended to cover historical and contemporary fiction for adults and biography. Non-textual narrative genres, such as film, can also be considered. In addition, other aspects of SceneML could be considered in the extended annotation process so as to include all SceneML elements, including narrative progression links, time, locations, and characters. The annotation process also can be extended to cover text from other languages once the guidelines have been validated in that regard. In addition, more scene annotated data can help in training models for other narrative-generating tasks such as Jin et al. (2022) and Mirowski et al. (2023) that would benefit from such a dataset.

- **Improvements to the segmentation model**

  While we have made progress in building a supervised model that can automatically segment narrative text into scenes, there is more that can be done to improve the accuracy of such models (scores for this, while comparable to Zehe et al. (2021) are

disappointingly low). One suggestion is to use a geographical and temporal references extraction model such as Ezeani et al. (2023) that can help to identify entities related to scenes and to identify scene changes. Another suggestion here is to develop another model that like the CRF approach treats the task as a sequence labelling task, but uses transformers. So instead of treating the task as a sentence classification or sentence pair classification task, treat the problem as sequence labelling over a sequence of sentences (so, one label per sentence). In addition, one necessary refinement to he segmentation model is adding the detection of non-scene segments and scene transition segments (STs) as further tasks. Furthermore, the use of large language models (LLMs) like GPT-3 (Brown et al., 2020) or GPT-4 (Achiam et al., 2023) could be explored to see if it can enhance the scene segmentation task. This can be applied through zero-shot learning, where no task-specific training data is required, or through few-shot training where only a few examples of the training data is required to be provided. This approach is ideal for our task given the limited size of the available training data and LLMs with their billions parameters could handle and understand the complex task of automatic segmentation of narrative text into scenes.

- **Scene clustering**

  One of the interesting questions this project raises is: *Can we cluster scenes into meaningful scene types?* After multiple stories have been segmented into scenes, we could, for example, cluster scenes of the same location type together (e.g. all café scenes would be grouped together). This question addresses the task of text clustering, with a focus on clustering scenes, depending on their type. Different clustering methods can be applied (e.g. k-means, hierarchical clustering, unsupervised neural networks, LDA, Markov clustering) and compared to rate their performance.

  There is also a potential opportunity to study the effectiveness of using VDL for the task of scene clustering.

- **Shared task on scene segmentation**

  Because there is only limited research on the task of automatic segmentation of

narrative text into scenes, a workshop on the topic and a shared task could be proposed to encourage collaboration by researchers in the NLP research community who are interested in narrative studies. This would provide an opportunity for interested researchers to meet, exchange ideas, come up with new suggestions and solutions and develop novel algorithms for this new NLP task.

References

Word vectors and semantic similarity. https://spacy.io/usage/vectors-similarity. Accessed: 2018-09-02.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Tarfah Alrashid. Gathering visually descriptive language from corpora. Master's thesis, University of Sheffield, 2017.

Tarfah Alrashid and Robert Gaizauskas. Scant: A small corpus of scene-annotated narrative texts [resource papers]. 2023.

Tarfah Alrashid and Robert J Gaizauskas. A pilot study on annotating scenes in narrative text using sceneml. In *Text2Story@ ECIR*, pages 7–14, 2021.

Tarfah Alrashid, Josiah Wang, and Robert Gaizauskas. Annotating and recognising visually descriptive language. In *Workshop on Interoperable Semantic Annotation (ISA-15)*, page 22, 2019.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210, 1999.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*, volume 43. 2009. ISBN 9780596516499. doi: 10.1097/00004770-200204000-00018. URL http://www.amazon.com/dp/0596516495.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010. doi: 10.1109/ICPR.2010.764.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior Research Methods*, 46 (3):904–911, Sep 2014. ISSN 1554-3528. doi: 10.3758/s13428-013-0403-5. URL https://doi.org/10.3758/s13428-013-0403-5.

Charles B. Callaway and James C. Lester. Narrative prose generation. *Artif. Intell.*, 139(2): 213–252, August 2002. ISSN 0004-3702. doi: 10.1016/S0004-3702(02)00230-8.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural*

Language Processing (EMNLP), pages 740–750, Doha, Qatar, 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1082.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

Freddy YY Choi. Advances in domain independent linear text segmentation. *arXiv preprint cs/0003083*, 2000.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

J. Corcoran. *Bunnies from the Future*. www.freekidsbooks.org, 2016. URL https://freekidsbooks.org/wp-content/uploads/2017/01/Bunnies-From-the-Future-FKB-MG-Books.pdf.

Bob Coyne and Richard Sproat. Wordseye: an automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496. ACM, 2001.

James E. Cutting. Event segmentation and seven types of narrative discontinuity in popular movies. *Acta Psychologica*, 149:69–77, jun 2014. ISSN 00016918. doi: 10.1016/j.actpsy.2014.03.003. URL http://linkinghub.elsevier.com/retrieve/pii/S000169181400078X.

James E Cutting, Kaitlin L Brunick, and Ayse Candan. Perceiving event dynamics and parsing hollywood films. *Journal of experimental psychology: human perception and performance*, 38(6):1476, 2012.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April 1997. ISSN 0162-8828. doi: 10.1109/34.588021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jesse Dodge, Amit Goyal, Xufeng Han, Alyssa Mensch, Margaret Mitchell, Karl Stratos, Kota Yamaguchi, Yejin Choi, Hal Daumé, III, Alexander C. Berg, and Tamara L. Berg. Detecting visual text. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 762–772, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL http://dl.acm.org/citation.cfm?id=2382029.2382153.

Will Dunne. *The dramatic writer's companion: tools to develop characters, cause scenes, and build stories*. University of Chicago Press, 2017.

Ignatius Ezeani, Paul Rayson, and Ian N Gregory. Extracting imprecise geographical and temporal references from journey narratives. In *Text2Story@ ECIR*, pages 113–118, 2023.

Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

Chris Fournier. Evaluating Text Segmentation using Boundary Edit Distance. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics*, page to appear, Stroudsburg, PA, USA, 2013. Association for Computational Linguistics.

Rob Gaizauskas. Information Extraction: Named Entity Recognition, 2017. lecture notes, Text Processing COM6115.

Robert Gaizauskas and Tarfah Alrashid. SceneML: A proposal for annotating scenes in narrative text. In *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation (ISA-15)*, pages 13–21, 2019.

Robert Gaizauskas, Josiah Wang, and Institut De Rob. Defining Visually Descriptive Language. Number September, pages 10–17, 2015.

Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. Deep learning approaches on image captioning: A review. *ACM Computing Surveys*, 56(3):1–39, 2023.

Evelyn Gius, Fotis Jannidis, Markus Krug, Albin Zehe, Andreas Hotho, Frank Puppe, Jonathan Krebs, Nils Reiter, Nathalie Wiedmer, and Leonard Konle. Detection of scenes in fiction. *DH*, 2019.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. *Cohesion in english*. Routledge, 2014.

Marti A Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64, 1997.

Yiping Jin, Vishakha Kadam, and Dittaya Wanvarie. Plot writing from pre-trained language models. *arXiv preprint arXiv:2206.03021*, 2022.

Dhiraj Joshi, James Z Wang, and Jia Li. The story picturing engine—a system for automatic text illustration. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):68–89, 2006.

Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

Daniel Jurafsky and James H. Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, January 7 2023. URL https://web.stanford.edu/~jurafsky/slp3/. Bug-fixing and restructuring release.

David Kauchak and Francine Chen. Feature-based segmentation of narrative documents. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing - FeatureEng '05*, number June, page 32, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1610230.1610237. URL http://www.aclweb.org/anthology/W/W05/W05-0405.

Hideki Kozima. Text segmentation based on similarity between words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 286–288. Association for Computational Linguistics, 1993.

Hideki Kozima and Teiji Furugori. Similarity between words computed by spreading activation on an English dictionary. In *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics* -, page 232, Morristown, NJ, USA, 1993. Association for Computational Linguistics. ISBN 9054340142. doi: 10.3115/976744.976772. URL http://portal.acm.org/citation.cfm?doid=976744.976772.

Hideki Kozima and Teiji Furugori. Segmenting narrative text into coherent scenes. *Literary and Linguistic Computing*, 9(1):13–19, 1994.

Sylvain Lamprier, Tassadit Amghar, Bernard Levrat, and Frederic Saubion. On evaluation methodologies for text segmentation algorithms. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 19–26. IEEE, 2007.

Beth Levin. *English verb classes and alternations: A preliminary investigation*. University of Chicago press, 1993.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. ISSN 2307-387X. doi: 10.1186/1472-6947-15-S2-S2. URL https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics

aligned pre-training for vision-language tasks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 121–137, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58577-8.

Thomas W MacFarland, Jan M Yates, Thomas W MacFarland, and Jan M Yates. Mann–whitney u test. *Introduction to nonparametric statistics for the biological sciences using R*, pages 103–132, 2016.

Arun S. Maiya. ktrain: A low-code library for augmented machine learning. *arXiv preprint arXiv:2004.10703*, 2020.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*, pages 1–9, 2015.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Stephen Marsland. Machine Learning: An Algorithmic Perspective, 2009. ISSN 87567016. URL http://books.google.com/books?id=n66O8a4SWGEC&pgis=1.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 1–8, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics. doi: 10.3115/977035.977037.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

Yue Ming, Nannan Hu, Chunxiao Fan, Fan Feng, Jiangwan Zhou, and Hui Yu. Visuals to text: A comprehensive review on automatic image captioning. *IEEE/CAA Journal of Automatica Sinica*, 9(8):1339–1365, 2022. doi: 10.1109/JAS.2022.105734.

Piotr Mirowski, Kory W Mathewson, Jaylen Pittman, and Richard Evans. Co-writing screenplays and theatre scripts with language models: Evaluation by industry professionals. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–34, 2023.

Aditya Mogadala, Marimuthu Kalimuthu, and Dietrich Klakow. Trends in integration of vision and language research: A survey of tasks, datasets, and methods. *Journal of Artificial Intelligence Research*, 71:1183–1317, 2021.

Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational linguistics*, 17(1):21–48, 1991.

Frederick Mosteller, David Lee Wallace, and John A Nerbonne. Inference and disputed authorship: The federalist. *(No Title)*, 1964.

John Niekrasz and Johanna D Moore. Unbiased discourse segmentation evaluation. In *2010 IEEE Spoken Language Technology Workshop*, pages 43–48. IEEE, 2010.

Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39: 103–134, 2000. ISSN 0885-6125. doi: 10.1023/A:1007692713085.

Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. Im2text: Describing images using 1 million captioned photographs. In *Advances in neural information processing systems*, pages 1143–1151, 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Stroudsburg, PA, USA, 2014.

Association for Computational Linguistics. ISBN 9781937284961. doi: 10.3115/v1/D14-1162. URL https://nlp.stanford.edu/pubs/glove.pdf.

Lev Pevzner and Marti A Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

Kirk Polking. . *Writing A to Z: The terms, procedures, and facts of the writing business defined, explained, and put within reach. Cincinnati, OH*. Writer's Digest Books. ISBN 0-89879-435-8, 1990.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. Iso-timeml: An international standard for semantic annotation. In *LREC*, volume 10, pages 394–397, 2010.

James Pustejovsky, Jessica L Moszkowicz, and Marc Verhagen. Iso-space: The annotation of spatial information in language. In *Proceedings of the Sixth Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, volume 6, pages 1–9, 2011.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Martin Riedl and Chris Biemann. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 student research workshop*, pages 37–42, 2012.

Simon Rogers and Mark Girolami. *A first course in machine learning*. CRC Press, 2016.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Wolf Schmid. *Narratology: An introduction*. Walter de Gruyter, Berlin, 2010.

Karin Kipper Schuler. Verbnet: A broad-coverage, comprehensive verb lexicon. 2005.

Erik F. Tjong Kim Sang and Fien De Meulder. 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 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119195.

Alexandra N. Uma, Tommaso Fornaciari, Dirk Hovy, Silviu Paun, Barbara Plank, and Massimo Poesio. Learning from disagreement: A survey. *J. Artif. Int. Res.*, 72:1385–1470, jan 2022. ISSN 1076-9757. doi: 10.1613/jair.1.12752. URL https://doi.org/10.1613/jair.1.12752.

WordNet. Princeton University. Princeton university "about wordnet.", 2010. URL http://wordnet.princeton.edu. Accessed: 2017-09-09.

Masao Utiyama and Hitoshi Isahara. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vijay K Vemuri. The hundred-page machine learning book: by andriy burkov, quebec city, canada, 2019, 160 pp., 49.99(hardcover); 29.00 (paperback); 25.43(kindleedition),(alternatively,canpurchaseatleanpub.comataminimumpriceof 20.00), isbn 978-1999579517, 2020.

Takahiro Wakao, Robert Gaizauskas, and Yorick Wilks. Evaluation of an algorithm for the recognition and classification of proper names. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 418–423. Association for Computational Linguistics, 1996.

Wikipedia. Cohen's kappa, 2023. URL https://en.wikipedia.org/wiki/Cohen%27s_kappa. Last accessed 26 March, 2022.

Olivia Winn, Madhavan Kavanur Kidambi, and Smaranda Muresan. Detecting Visually Relevant Sentences for Fine-Grained Classification. In *Proceedings of the 5th Workshop on Vision and Language*, pages 86–91, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-3213. URL http://aclweb.org/anthology/W/W16/W16-3213.pdfhttp://aclweb.org/anthology/W16-3213.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40, 2017.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, jun 2010. ISBN 978-1-4244-6984-0. doi: 10.1109/CVPR.2010.5539970. URL http://ieeexplore.ieee.org/document/5539970/.

Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN Database: Exploring a Large Collection of Scene Categories. *International Journal of Computer Vision*, 119(1):3–22, aug 2016. ISSN 0920-5691. doi: 10.1007/s11263-014-0748-y.

Albin Zehe, Leonard Konle, Lea Katharina Dümpelmann, Evelyn Gius, Andreas Hotho, Fotis Jannidis, Lucas Kaufmann, Markus Krug, Frank Puppe, Nils Reiter, et al. Detecting scenes in fiction: A new segmentation task. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3167–3177, 2021.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explana-

tions by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

Appendix A

Annotation Guidelines

The University Of Sheffield.

Scene Annotation Guidelines

Authors:

Tarfah Alrashid, Prof. Robert Gaizauskas

Supervisor:

Prof. Robert Gaizauskas

June 2021

Table of Contents

1 Introduction

To date, there has been a lot of research interest in integrating vision and language for different tasks. Image captioning for instance, is the task of generating natural language descriptions for a given image. Models are trained using datasets having texts aligned with images that are manually annotated. Another example of an image-based task is automatic text illustration, which is the task of automatically finding a suitable image for a given segment of text.

Such research integrating vision and language generally tries to solve image-based tasks, by relying on images aligned with texts. The problem is twofold: (1) many captions associated with images do not just describe the image; (2) the volume of captions overall is not great.

Therefore, we propose exploring how text not associated with images can be used to help in this type of research. We can find such text in narratives and drama, because we get descriptions of setting in fiction. We may get visual descriptions of things in other writing also, for example biography. So we propose to investigate ways of utilising these other purely linguistic image-based resources to help in tasks relating to images.

Our research aims to explore the idea that scenes are divisions found in all narratives which, amongst other things, have to do with physical setting. Usually there are descrip- tions that enable the reader to note scenes have changed. Automatic scene segmentation could be useful for purposes of illustration and alignment. This project proposes the idea of automatic segmentation of narrative text (stories) into scenes. In order to carry out this task, we first provide a definition of what a scene is. Then, we define guidelines for scene annotation that help in identifying scene change and scene boundaries. Automatic segmen- tation of text into scenes could help in building a text corpus of scenes that could be used by systems converting scenes into images. In addition, scenes could be clustered and common visual elements identified for scenes clustered together. If scene type labels can be given to these clusters, then these labels could be attached to images whose visual elements model these clusters.

1.1 Overview of the Report

The report is structured as follows:

- Section 2 gives a brief introduction to SceneML.
- Section 3 explains the annotation process.
- Section 4 explains how to use the annotation tool.

2 SceneML

2.1 The Annotation Framework

Taking into consideration the literature related to scene definition, we treat a scene as a unit of a story in which the elements: time, location, and main characters are constant. Any change in these elements indicates a change of scene. A scene is an *abstract* discourse element. It consists of a location or setting, a time and characters who are involved in the events that take place in the scene. These elements that constitute a scene exist in the real or fictive world (*storyworld,* as per narrative theory) that the narrative revolves around. “The scene itself is an abstraction away from the potentially infinitely complex detail of that real or fictive world, a cognitive construct that makes finite, focussed narrative possible" [2].

A scene in a textual narrative may consist of one or more *scene description segments* (SDS). An SDS is “a contiguous span of text that, possibly together with other SDSs, ex- presses a scene" [2]. Typically a scene consists of one SDS, unless the scene introduces other scenes such as scenes about past (memories) or future events, in which case the textual realisation of one scene may be split with text from another scene embedded within it, resulting in non-contiguous SDSs for the embedding scene. Furthermore, SDSs can be in the form of “spatially distinct locations that are topologically contained within or connected to the embedding SDS, or if the author is employing the narrative device of rotating between multiple concurrent scenes each of which is advancing a distinct storyline (a common technique in action movies)"[2].

It is important to define what each element of a scene (*characters*, *time*, *location*) is to facilitate the annotation process and to make it easier to detect scene changes if any of the elements changed. Scene changes have been studied by researchers such as [1] and others. Studying which of the scene elements (*characters*, *time*, *location*) changes in a scene can be considered a contribution to previous studies. Here we propose to adopt the definitions, and annotation standards, for the elements (*time*, *location*, and *spatial entities*) from Iso-TimeML and Iso-Space. As for characters, we will adopt the definition and annotation standards for named entities of type person from the ACE program which has been recently used in in the TAC 2018 entity discovery and linking task [2].

The previous standards will facilitate the annotation process for all mentions of times, locations/spatial entities and persons represented in the text. Nonetheless, we are inter- ested in the relations between the specific entities that construct a specific scene (i.e., the *characters*, *time*, *location* that construct a specific scene) [2].

2.2 SceneML Elements

SceneML elements are categorised into two main categories:

1. **Entities:** entities consists of scenes, SDSs, character, time, and locations
2. **Relations:** “scene-scene narrative progression links".

Scenes

Scenes are the main element in SceneML. A scene has attributes: id, time and location, these attributes are unique for each scene. They also include a list of character sub- elements as there may be more than one character for each scene [2].

SDSs

Scene description segments (SDSs) are the actual strings of text that compose the scene. SDSs cannot belong to more than a scene, but a scene can be composed of multiple SDSs. They include the following attributes: id and scene_id that is the id of the scene that the SDS belongs to, these attributes are unique to each SDS [2].

Time

Time elements used here are the ones developed by ISO-TimeML, they include an id attribute and a text segment. Time could also include the time of the storyworld and it is represented by the attribute base [2].

Location

Like time, we use the location element developed in ISO-Space; this also includes an id attribute that is unique for each location and a text span [2].

Character

Here we use the named entity or type person from the ACE English Annotation Guidelines for Entities, the only difference here is that animals and non-humans can be considered as characters. They have an id attribute that is uniquely assigned for each character and a text segment. Character might include the type attribute of the ACE specification [2].

Narrative Progression Links

Narrative progression links (nplinks) link between any two textually adjacent scenes, these links have different types depending on the type of temporal transition between the two scenes. So far we have four types of links [2]:

- sequence links are assigned when the scene change happen because of a change in location e.g. a character moves to another place
- analepsis when there is a flashback in the scene e.g. memory of the past
- prolepsis (or flashforward) when then we are taken forward in time
- concurrent links are assigned between two scenes when the transition happen because there is another thread of the story happening at the same time so the transition take us to different charters and different place but the same time

3 The Annotation Process

Novels and stories usually consist of a set of scenes. A scene as defined in literature is a unit of a story in which the elements: time, location, and the main characters are constant. Any change in these elements indicates a change of scene. A scene is an abstract discourse element, i.e. it is not explicitly indicated in the text. It consists of a location or setting, a time and characters who are involved in the events that take place in the scene. A scene in a story may be realised by one or more scene description segments (SDS). An SDS is "a contiguous span of text that, possibly together with other SDSs, expresses a scene"[1]. Typically a scene consists of one SDS, unless that scene involves other SDSs of other scenes such as scenes about past (memories) or future events. Furthermore, SDSs can be in the form of "spatially distinct locations that are topologically contained within or connected to the embedding SDS, or if the author is employing the narrative device of rotating between multiple concurrent scenes each of which is advancing a distinct storyline (a common technique in action movies)"[1].

Example: Figure 1 shows an illustration of how scenes should be connected through relations. *Scene1* consists of two non-adjacent SDSs, *SDS1* and *SDS3*. While *Scene2* consists of only one SDS, *SDS2*. *Scene1* and *Scene2* are connected by a flashback relation.

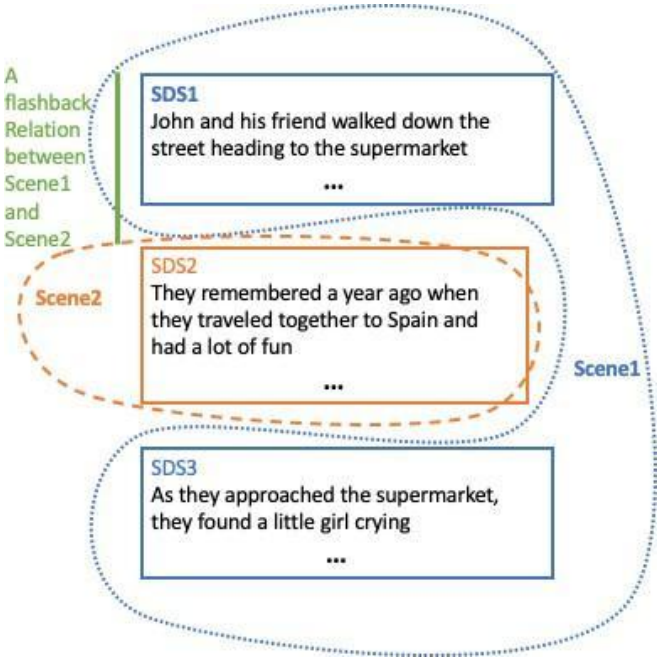


Figure 1: Two scenes, *Scene1* and *Scene2* connected with a flashback relation.

Textually adjacent scenes, i.e. scenes with textually adjacent SDS's, should be linked together through progression links. Note that these links are links between scenes not SDSs. We identify four types of progression:

- sequence, when one scene follows on from another, e.g. when characters move from

one location to another;

- analepsis (or flashback), when we are taken to another, earlier time and possibly other details such as location and characters change as well;
- prolepsis (or flashforward), when we are taken forward in time;
- concurrent, when we are taken to another location with different characters, where another thread of the story is developing at the same time as the textually preceding scene.

The annotation tool does not support the creation of abstract entities such as scenes. Therefore to capture the information that multiple SDSs belong to the same scene, we will annotate a relation between SDSs that are in the same scene ("is_in_the_same_scene_as"). Furthermore to capture progression relations between scenes, we will annotate these relations between the initial SDSs of related scenes, with the understanding that really the relation holds between scenes and not SDSs. Figure 2 shows how to annotate the example in Figure 1 with the BRAT annotation tool. The flashback relation is connecting *SDS1* from *Scene1* with *SDS2* from *Scene2*. *SDS1* is connected to *SDS3* with a "is_in_the_same_scene_as" relation to indicate that they are in the same scene. It is im- portant to define what each element of a scene (characters, time, location) is to facilitate the annotation process and to make it easier to detect scene changes if any of the elements changed.

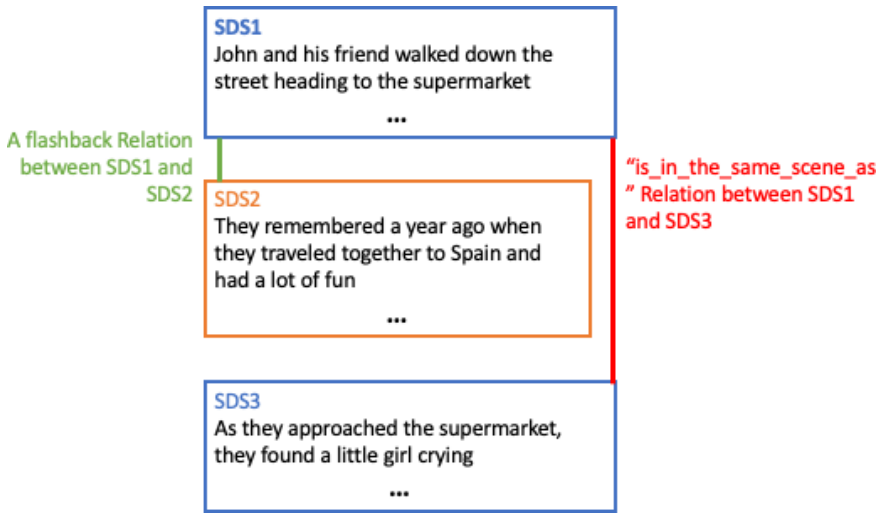


Figure 2: How scenes and SDSs are annotated with the annotation tool.

You are required to annotate SDSs in the text provided, then link these SDSs together if you see that multiple SDSs compose a scene. Note that each SDS must belong to only one scene, but every scene can contain one or more SDSs. In addition, you are required to annotate the elements: time, location, and characters. Please note that the change of these elements indicates a change in the whole scene not a change in the SDSs. Please also annotate the relations between scenes as explained above.

3.1 Detailed Annotation Guidelines

Below you will find more detailed explanations of each of these elements and instructions for the annotation process.

Entities to be Annotated:

Scenes

Scenes are the main element in SceneML. A scene has elements time and location, which are unique for each scene. A scene also includes a list of character sub-elements as there may be more than one character for each scene. Please note that this is an abstract element, which means that this element is not annotated explicitly in the text.

Then how can one annotate a scene if it is an abstract element? you can annotate a scene by: (1) if the scene has only one segment (SDS) then all you have to do is to annotate the segment as (SDS), (2) if the scene has multiple segments (SDSs), then annotate each one of these segments as (SDS) separately and then link them together with *is_in_the_same_scene_as* relation. By linking these segments together we indicate that they all belong to the same scene.

So all you have to do here is to build a conceptual understanding of where the scenes are in the text and annotate the SDSs that constitute these scenes. While you do that, you also need to pay attention to the three main elements that compose the scene, and once you notice a change in any of these elements you need to end annotating the current SDS that compose the current scene and start a new one by following points one and two above.

SDSs

Scene description segments (SDSs) are the actual strings of text that compose the scene. An identified SDS cannot belong to more than one scene, but a scene can be composed of multiple SDSs. In order to identify an SDS, you need to make sure that the 3 elements of a scene (time, location, characters) persist across that continuous span of text. The moment you notice a change in any of the elements, you need to consider a new scene and the end of the current SDS. Sometimes, the time entity might not be stated literally in the text; it can be inferred from the tense used, or it might be only mentioned at the beginning of the story. It is OK for an SDS to not have a time entity explicitly mentioned as long as the time persists from a previous scene; similarly for location. Please note that each SDS should be at least one sentence long.

Scene Boundaries and scene shifts:

Here you will find more information on how to end the current scene (SDS), and begin a new one. And what could be considered a sufficient change for a scene shift.

- **If there is a location change that results in a scene shift, where exactly should we stop the current SDS, and start the following one?**
  Transitioning sentences stating that a person is arriving at a new location or leaving the current location should be handled one of two ways: (1) if the location is preceded by a verb like *arrived*, *got to,* … etc. then we can consider the sentence to belong to the following, new SDS (2) if the location is preceded by a verb like *left*, *got out of,* …, etc.) then we can consider the sentence to belong to the current SDS.
- **Changing locations: when to consider a change in a location, if you are unsure about changing the scene?** This means if the location is changed but you are not sure if that change in the location (e.g. very minor change) will lead to a scene change.
  - If there is a very minor location change that you feel does not lead to a change in the scene, e.g. if the characters of the current scene only move from one side of the room to the other without exiting the current room, then this minor location change should be ignored, and there is not a shift in scenes.
  - Try to ask yourself, if you were to design sets for a play version, would you need a different backdrop for it?.
- **Changing characters:** when reading the whole text first, you'll be able to build a general idea about who are the characters that contribute more to the story and who are the characters that do not have a major impact on the scenes of the story. When major characters enter the current scene or leave the scene, you should end the scene at the current SDS and start a new one. Nevertheless, if minor characters that do not have a major role in the scene, have any change in their situation (entering, or leaving), it should not affect the current SDS of the scene, and thus the scene should continue.

Non-scene segments: please note that not all segments (i.e. paragraphs/ sentences/ phrases ... etc) need to be annotated as belonging to an entity. Some segments can be left without annotation if you see that they do not belong to a scene or do not create a scene. Some of these non-scene segments may reflect the author's general view of something, or may be a description of a character or a place or some period of time prior to setting that the rest of the story takes place in. So far we have identified three types of a non-scene segment; there may be more:

1. general philosophising or opinion segments -- these do not mention specific characters or events but rather offer the author's view on some topic or another;
2. background information segments -- they may lack precise details of time or location, are clearly set before the current narrative begins, and provide insight into characters or past events of relevance for understanding the story that is about to begin;
3. narrative summary or narrative catchup -- where various events, usually non-pivotal, are summed up in a relatively compressed passage that lets the reader know how events have moved on in between two scenes that are presented in greater detail, without precisely indicating all the details of character, location and time. Events themselves in these passages may not be distinguished from each

other, if the intention is to indicate a trend or only the outcome.

For example, the first two paragraphs of chapter 1 of the novel Adventures of Sherlock Holmes are non-scene segments because they contain background information. They describe the characteristics of the character *Sherlock Holmes* and the relationship between him and his friend *Watson*. There is nothing actually happening at these segments, so they should not be annotated as scenes. You can find the novel here: https://www.gutenberg.org/files/1661/1661-h/1661-h.htm#chap01. In addition, the first two paragraphs of the chapter 1 of the novel *A Tale of Two Cities* are non-scene segments (background information) as they describe the era and the character of that era, but no specific events are related. You can find the novel here: https://www.gutenberg.org/files/98/98-h/98-h.htm. In another example, the first two paragraphs of chapter 1 of Pride and Prejudice - *It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife… etc.* And the first two paragraphs of chapter 1 of *A Tale of Two Cities* are non-scene general philosophising segments. You can find *Pride and Prejudice* here: https://www.gutenberg.org/files/1342/1342-h/1342-h.htm#link2HCH0001. The first paragraph of chapter 3 of *Pride and Prejudice* is a nice example of a narrative summary non-scene segment.

As you can see from the examples above most of the non-scene segments occurred at the beginning of a story or the beginning of a chapter, this is what we have encountered so far but non-scene segments can be embedded in text. If you find a non-scene segment embedded in text that is less than a full sentence, ignore the segments and consider it to be part of the current SDS segment.

Scene Transitioning Segments (ST)

These are text segments that indicate a character or characters are moving from one location to another, where this transition is neither clearly related to the previous scene location nor the following scene. In addition, there is not any event of narrative significance happening during the transitioning segment other than the transition. These segments should not contribute to the narrative of the story (e.g. if there is a conversation happening between characters of the story, they should not speak important information related to the story) . For example,

> *And so in ten minutes I had left my armchair and cheery sitting-room behind me, and was speeding eastward in a hansom on a strange errand, as it seemed to me at the time, though the future only could show how strange it was to be.*

The above segment (*Adventures of Sherlock Holmes* chapter 6) is annotated as a transitioning segment (ST), it shows that the character Watson has left the previous scene location (his house) and is en route to the next scene location.

Time

Here you need to annotate the text span that refers to the time of the scene (i.e. the time the events of the scene are happening). Below you will find more examples of what could be considered a time of a scene and more instructions on the annotation process.

Time spans could be:

1. Noun (including Proper Nouns): e.g. today, Thursday

2. Noun Phrase (NP): e.g. the morning, Friday night, the last two years
3. Adjective: e.g. current
4. Adverb: e.g. recently
5. Adjective or Adverb Phrase: e.g. half an hour long, two weeks ago, nearly a half- hour

Please be aware to annotate the time entity that the scene events are happening in. Some scenes might have multiple mentions of time entities (they could be mentioned in a question or in a description of something else) although only one will be considered as the time of the scene. For example, consider this question that was asked by a character in a children's story: “Do you not have good fabrics in the future?” I asked, eyeing a box that seemed to be stuck to the wall" here the word future should not be annotated as the time of the scene in as it does not refer to the time the scene events are happening.

Examples of a time entity:

- **ONE TIMEX tag will be used when there is no intervening token between temporal terms:**
  - twelve o’clock midnight
  - Friday evening
  - 8:00 p.m. Friday
  - Tuesday the 18th
  - November 1943
  - Fall 1998
  - this year’s summer
- **ONE TIMEX tag will be used when certain prepositions appear within temporal expressions:**
  - the second of December
  - summer of 1965
  - ten of two ten minutes to three
  - half past noon
  - eleven in the morning
- **TWO TIMEX tags will be used when sequences of two temporal expressions that are ordered one relative to the other. They generally involve the use of temporal prepositions and conjunctions like from, before, after, following, prior to, etc. :**
  - I’m leaving on vacation [two weeks] from [next Tuesday]. John left [2 days] before [yesterday]
- **TWO TIMEX tags will be used when sequences of two temporal expressions that can be related by a temporal link:**
  - I tutored an English student [some Thursdays] in [1998]. The concert is at [8:00 p.m.] on [Friday]. The concert is [Friday] at [8:00 p.m.]

Location

Like time, you need to annotate the text span that refers to the location of the scene (i.e. the location the events of the scene are taking place in). Below you will find some examples of

locations, and some annotation cases you need to be aware of.

Some examples of possible locations of scenes:

- Building
- United States
- West Tikrit

Some notes in annotating locations:

- Please make sure when annotating a location that the location is the location of the current scene, as some scenes may have mentions of more than one location, though only one of them is the location where the scene events happen. For example, " "You just wait here until we're ready to take you home," he said, as he floated towards one of the exit tunnels". If the word "home" were to be tagged as the location of the scene that would not be correct as events of the scene do not take place at "home" — it is only mentioned in the conversation.
- Phrases like "centre of the room" was tagged as being the location of the scene, it is not completely wrong but "room" is supposed to be tagged alone as the (Location) of the scene instead of tagging the words (centre, of, and the) as it does not matter where in the room the scene is happening as long as it is in the room the scene would not change.
- Similarly, all the words in the sub sentence "back into the corridor" were annotated as a location and that is not correct, only the corridor should have been tagged as a location.

Character

Text spans that refer to characters need to be annotated, characters could be humans, non humans (e.g. animals), and subjects. Below some specific cases in annotating characters:

- Only the first mention of each character should be annotated in each scene. Characters, could be mentioned several times in every scene or sds, but only the first mention should be annotated.
- Sometimes subjects (e.g. he, she) could be used to a character that is already annotated within the same scene, then this subject need not to be annotated. For example: John is playing football with his friend. He is a good football player. If the two sentences belong to one scene, then we only annotate John. But if the two sentences belong to two different scenes, then we annotated John as an actor of the first scene and He as an actor of the second scene.
- If the subject of a certain character is mentioned before its character in the same scene, then we do not annotate the subject, we annotate the name of the character instead. The subject is mentioned before its name, when the name is mentioned in a previous scene, so in this case we annotated the first mention of the name in the scene. If the name is not mentioned in the scene at all, then we annotate its first mention of its subject.
- When multiple characters appear next to each other (e.g. Biff, Boff and Skip) each one of them should be tagged alone as a character, do not annotate all of them as one segment. Because then the word and will be also tagged as a character and that is not

true.

3.2 How to annotate

Start by reading the whole text first without annotating to gain a whole understanding of the story and events. Then, read it again and annotate all SDSs and within each SDS annotate the three entities if they exist as explained in the previous section.
If no time is given in the current scene, add a note to the scene with the time being as the previous scene.

4 BRAT (Annotation tool)

- Go to the page: http://sceneml.shef.ac.uk/brat/#/your name/

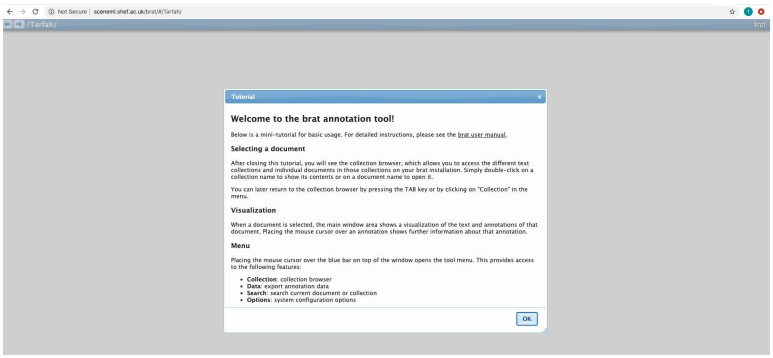


Figure 3

- A popup window will appear that contain all the files in you page, select the file you want to annotate and click OK.

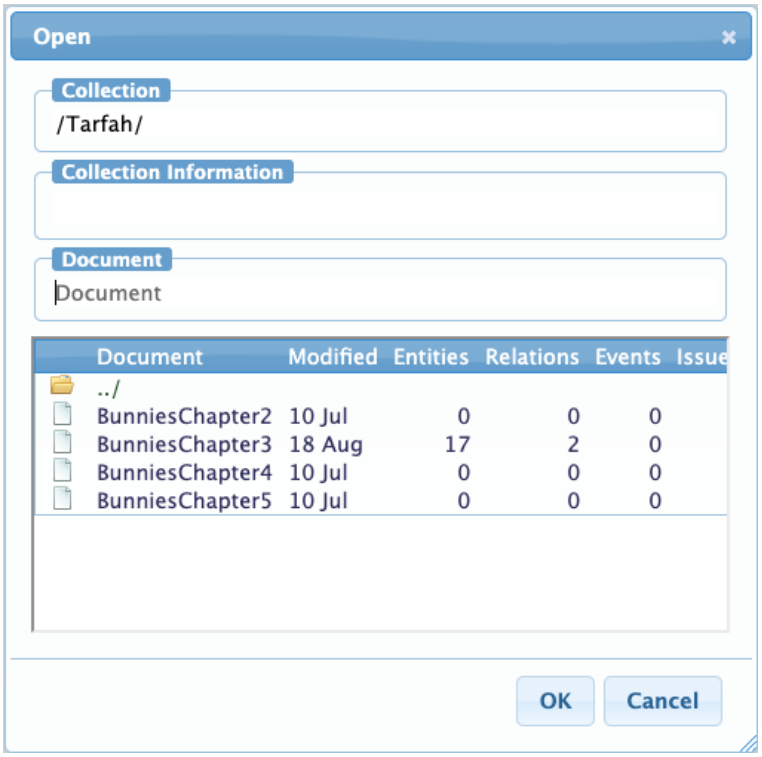


Figure 4

- In order to annotate, you'll need to login first. Username and password has been sent to your email. Hover the mouse on the login icon in the top bar and log in (see Figure 5).

Figure 5

- Select the text span you want to annotate as in Figure 6

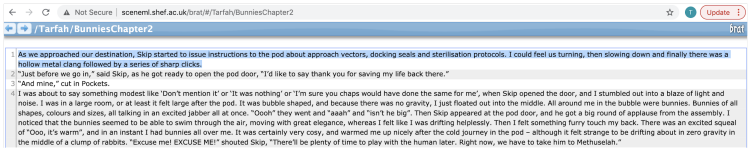
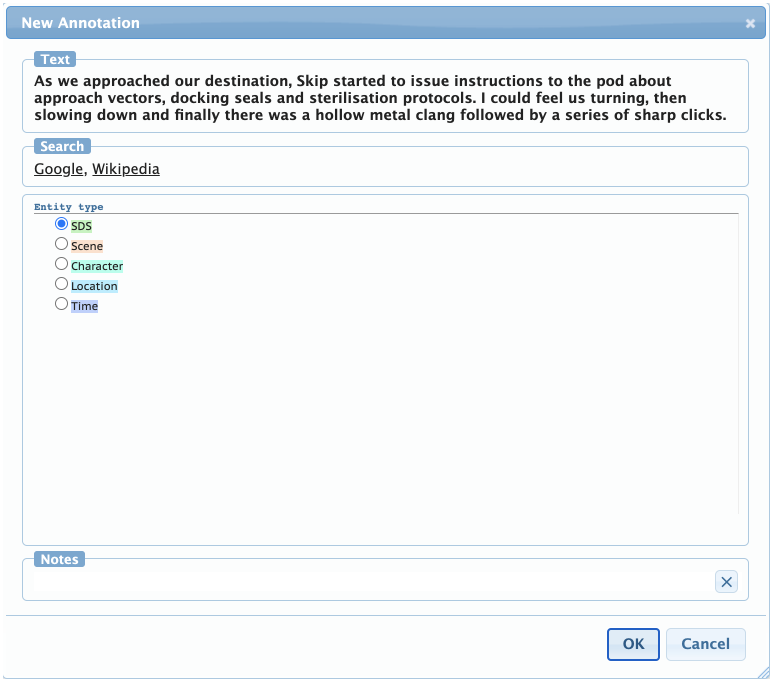


Figure 6

- A pop up menu will appear with the all entity types, select the entity type you want and click OK (see Figure 7)
- To connect two SDSs together;


New Annotation
Text
As we approached our destination, Skip started to issue instructions to the pod about approach vectors, docking seals and sterilisation protocols. I could feel us turning, then slowing down and finally there was a hollow metal clang followed by a series of sharp clicks.
Search
Google, Wikipedia
Entity type
SDS
Scene
Character
Location
Time
Notes
OK
Cancel

Figure

Bibliography

[1] James E. Cutting. Event segmentation and seven types of narrative discontinuity in popular movies. *Acta Psychologica*, 149:69–77, jun 2014. ISSN 00016918. doi: 10.1016/j.actpsy.2014.03.003. URL http://dx.doi.org/10.1016/j.actpsy.2014.03.003http://linkinghub.elsevier.com/retrieve/pii/S000169181400078X.

[2] Robert Gaizauskas and Tarfah Alrashid. SceneML: A proposal for annotating scenes in narrative text. In *Workshop on Interoperable Semantic Annotation (ISA-15)*, page 13, 20