Olena Mashkina, BSc

Linear Text Segmentation with Neural OIE on Novels and Subtitles

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl-Inf. Dr. Stefanie Lindstaedt

Graz, September 2020

Affidavit

Acknowledgments

I am deeply grateful to my supervisor Ass.Prof. Dipl.-Ing. Dr.techn. Kern for encouraging my work. Thank you for your valuable feedback as well as your ability to stay positive throughout the course of dataset acquisition and methodology challenges.

I would like to thank my fiancé Bernd for his support, understanding, patience and his ability to make me laugh even during the difficult times. I am immensely grateful to have you in my life, without you I would not be the same person I am now and this thesis would never be finished.

Special thanks to my friend Asia for her kindness and calmness facing the immeasurable number of my daily complaints. You have helped me more than you realize.

Finally, I am grateful to those who actually bothered to ask what my thesis is about instead of asking me when is it going to be done.

Olena Mashkina

Graz, 10.09.2020

Abstract

Automatically separating text into coherent segments sharing the same topic is a nontrivial task in research area of Natural Language Processing. Over the course of time text segmentation approaches were improved by applying existing knowledge from various science fields including linguistics, statistics and graph theory. At the same time obtaining a corpus of textual data varying in structure and vocabulary is problematic. Currently emerging application of neural network models in Natural Language Processing shows promise, which particularly can be seen on an example of Open Information Extraction. However the influence of knowledge obtained by an Open Information Extraction system on a text segmentation task remains unknown.

This thesis introduces text segmentation pipeline supported by word embeddings and Open Information Extraction. Additionally, a fictional text corpus consisting of two parts, novels and subtitles, is presented. Given a baseline text segmentation algorithm, the effect of replacing word tokens with word embeddings is examined. Consequently, neural Open Information Extraction is applied to the corpus and the information contained in the extractions is transformed into word token weighting used on top of the baseline text segmentation algorithm.

The evaluation shows that application of the pipeline to the corpus increased the performance for more than a half of novels and less than a half of subtitle files in comparison to the baseline text segmentation algorithm. Similar results are observed in a preliminary step in which word tokens were substituted by their word embedding representations. Taking into account complex structural features of the corpus, this work demonstrates that text segmentation may benefit from incorporating knowledge provided by an Open Information Extraction system.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Research Question

An author is responsible for giving their text a structure comprehensive for human readers. However, with a constantly emerging amount of unstructured textual information on the Internet and in print automatic text segmentation gained significant role in Natural Language Processing (NLP) research field. Text segmentation is a nontrivial task due to ambiguous topic boundaries, differences in writing style, vocabulary and text length. Text segmentation of automatically generated text, for instance automatic video captions, also poses a challenge. Application of text segmentation not only provides visual separation of unstructured text to the reader but also serves as a prerequisite task in other NLP applications, for instance, information retrieval, summarization and text understanding. An NLP task of Open Information Extraction (OIE) allows expressing key information contained in a text by extracting n-ary propositions. This thesis assumes that enriching linear text segmentation process with OIE knowledge holds potential for positively influencing the text segmentation results. As far as the author of this thesis is aware, the task of linear text segmentation improvement with the help of OIE has not been approached before.

Therefore, the main research questions this thesis aims to answer are:

- **Related to Natural Language Processing:**
  How does a modification of a linear text segmentation method by adding knowledge generated by Open Information Extraction (OIE) influence the performance of this method?
- **Related to the created dataset:**
  How does the performance of presented in this work segmentation pipeline compare for different fictional narrative text corpora (novels and subtitles)?

To establish a benchmark performance for a linear text segmentation an existing algorithm is used. More formally, given a text segmented into a set of sentences $\{sentence_1, \dots, \}$, method presented in this thesis may be expressed as a function composition. First, functions necessary to prepare the text for linear text segmentation are executed, $lt \circ ts : S \longrightarrow T$, where ts is a text segmentation function and lt is a lemmatization and tokenization function.

$$\begin{aligned} &ts : S \longrightarrow W,\ S = \{sentence_1, \dots, \}\ and\, W = \{word_1, \dots, \} \\ &lt : W \longrightarrow T,\ T = \{token_1, \dots, \} \end{aligned}$$

Then, after $lt \circ ts : S \longrightarrow T$ is performed by linear text segmentation algorithm part of the pipeline consisting of function composition $tt_3 \circ tt_2 \circ tt_1 : T \longrightarrow SB$ is executed.

$$\begin{aligned} &tt_1 : T \longrightarrow TS,\ TS = \{token\_sequence_1, \dots\} \\ &tt_2 : TS \longrightarrow B,\ B = \{block_1, \dots\} \\ &tt_3 : B \longrightarrow SB,\ SB = \{segment\_boundary_1, \dots\} \end{aligned}$$

To answer the first research question the linear text segmentation algorithm is modified which may be represented in form of compositions $f_1 \circ lt \circ ts : S \longrightarrow WE$ followed by $f_3 \circ f_2 : S \longrightarrow W'$ and $f_6 \circ f_5 \circ f_4 : W', WE \longrightarrow SB$.

$$\begin{aligned} &f_1 : T \longrightarrow WE,\ WE = \{word\_embedding_1, \dots\} \\ &f_2 : S \longrightarrow OIE,\ OIE = \{oie\_proposition_1, \dots\} \\ &f_3 : OIE \longrightarrow W',\ W' = \{weight_1, \dots\} \\ &f_4 : W', WE \longrightarrow WES,\ WES = \{weighted\_embedding\_sequence_1, \dots\} \\ &f_5 : WES \longrightarrow WEB,\ WEB = \{weighted\_embedding\_block_1, \dots\} \\ &f_6 : WEB \longrightarrow SB \end{aligned}$$

1.2. Outline

This thesis describes the application of OIE based on Recurrent Neural Networks (RNN) to the task of linear text segmentation given a corpus consisting of novels and subtitles to film adaptations of these novels. The remainder of this thesis is structured as follows.

Chapter 2 provides an overview of existing research and literature related to the topics covered in this thesis. Section 2.1 introduces basic concepts required to understand the contributions of this work to the NLP research field. Section 2.2 describes recent work in research areas related to this thesis in more detail. First, the advancement of linear text segmentation is presented in Subsection 2.2.1. Consequently, current state of knowledge on OIE is described in Subsection 2.2.2. Finally, application examples of literary fiction and film subtitles are given in Subsection 2.2.3.

Chapter 3 provides information about the dataset and technical implementation. First, in Section 3.1 a detailed presentation of the dataset is given followed by motivation for choosing this type of data in Subsection 3.1.3. Second, details about the preprocessing of the dataset are provided in Section 3.2. Third, TextTiling linear text segmentation algorithm is explained in Section 3.3. Then, Recurrent Neural Network Open Information Extraction (RnnOIE) algorithm used for OIE extraction is introduced in Section 3.4. Finally, Section 3.5 describes how in this work OIE results are integrated into linear text segmentation algorithm.

Chapter 4 demonstrates the performance of linear text segmentation approach presented in this work. First, text segmentation measures used for the evaluation are explained in Section 4.1. Afterwards, evaluation process including parameter search is described in detail in Section 4.2. Finally, achieved results are shown and discussed in Section 4.3.

In conclusion, Chapter 5 outlines possible further research directions and suggestions for future work.

2. Related Work

2.1. Background

This section gives an overview of the concepts relevant for this thesis. First, NLP specific concepts are introduced. Linear text segmentation is described in Subsection 2.1.1. Main properties along with the formal definition of OIE are presented in Section 2.1.2. Subsection 2.1.3 provides a description of word embeddings and introduces distributional hypothesis the word embeddings are based on. Afterwards, non-technical terms related to fiction and film are introduced. The qualities of dystopian fiction are described in Subsection 2.1.4. Subsection 2.1.5 defines the difference between related terms *subtitles* and *captions* along with an explanation which type of subtitles are used in this work. Finally, Subsection 2.1.6 provides insight into a meaning of a term ”film adaptation”.

2.1.1. Linear Text Segmentation

The goal of ***linear text segmentation***, which is sometimes called topic segmentation, is to automatically locate a transition from one topic to another in a text [17, 37, 59]. In order to achieve this goal a linear text segmentation system separates given input into a sequence of neighboring textual segments. Each of such segments contains a certain number of passages such as paragraphs or sentences and is characterized by a single homogeneous topic. Linear text segmentation algorithm used in this thesis is called TextTiling [26], a detailed description of it can be found in Section 3.3.

2.1.2. Open Information Extraction

Considering a Web corpus as an input to an already existing traditional closed Information Extraction (IE) systems certain challenges became apparent [18]. First, traditional IE relies on knowing the nature of an input domain beforehand. However, considering the versatility of the Web it is hard to determine a domain before the extraction. Second, IE expects predefined number and type of relations to be extracted by the system in advance. For Web it is not known what types of relationships are typical for the data under consideration and the number of relations is not only unknown, but also large. Third, for IE it is possible to define new additional relations, which leads to a repetition of the extraction process as a consequence. A system operating on a corpora as large as the Web has to be capable of performing extractions in a single pass.

In an attempt to overcome aforementioned challenges ***Open Information Extraction (OIE)*** as a more flexible alternation of IE was introduced [6]. Accordingly, three main properties of an OIE system may be defined as follows:

1. Domain independence. There is no predefined domain on which OIE has to rely for performing the extraction of relations.
2. Unsupervised extraction (automation). An extractor is able to handle diverse data without explicit instructions about which relations it will encounter. Detecting the relations is one of the tasks of the extractor.
3. Scalability (efficiency). It is possible to extract all relations from a large amount of heterogeneous unstructured data in one pass.

Formally an OIE system can be described as a function f_{oie} which takes a collection of sentences $s = \{sentence_1, \ldots, \}$ (a textual document) as an input and returns a set of n-ary tuples for each sentence in a collection *s* as an output [65]. Each n-tuple contains at least two arguments connected by a semantic relation between them, also called a predicate $\{$*argument 1, predicate, argument 2*$\}$. There is no limit to the number of relational n-tuples extracted from a single sentence. The n-tuples should represent propositions clearly expressed in the sentence, however generation of n-tuples representing implied propositions is optional [53]. Table 3.4 shows an example of OIE extraction by an extractor used in this work, RnnOIE [54].

2.1.3. Word Embeddings

Word embedding generation is a class of methods for transforming original textual data into a vector space based on prediction from the linguistic context [1, p. 331]. Each word in a vocabulary of a corpus is assigned a single real-valued vector which is learned by the chosen word embedding method. Word embedding vectors are dense and low-dimentional as opposed to sparse high-dimentional one-hot style representation of a vocabulary [44].

Word embedding is able to capture a set of contexts in which a word appears and therefore can be considered an approximation of this word's meaning [5, 67]. A word sharing the same neighbors with another word, but not necessarily co-occuring with it suggests these two words have similar meaning. This quality is based on distributional hypothesis from linguistic theory which states that words occurring in similar context tend to have similar meaning [25].

Widely used algorithms to generate word embeddings are GloVe [48], fastText [12, 31] and Word2Vec [42]. This work uses neural word embedding algorithm Word2Vec as described in Section 3.5.

2.1.4. Dystopian Fiction

Though there exists no agreed classification or a clear definition of dystopian literature genre, it is agreed upon that dystopian fiction is characterized by depiction and critique of political and social tendencies over the period of past few decades prior to the time of publication [62, p. 6].

Dystopian fiction characterizes a future society in an unwanted, critical state [60, 51]. Government or other type of power has gained total control of the population [60]. An individual is portrayed as weak and oppressed, alienated from others, the only aim of their existence being the fulfillment of their duties and social responsibilities. Free communication between individuals is restricted which may be linguistically emphasized by the introduction of a new language. Development of a new language or code of speech is used to enable the reader to gain deeper understanding of the described social situation. For example, in novel ”Nineteen Eighty-Four” a new code of speech called ”Newspeak” is used to influence or gain power over people [62,

p. 81]. Social commentary on concerns or hopes about technology and its effect on the individuals and society as a whole is frequently given [7, p. 145].

Novels ”We” by E. Zamyatin, ”Brave New World” by A. Huxley and ”Nineteen Eighty-Four” by G. Orwell are widely accepted as examples of classic dystopian novels [62, p. 16]. Contemporary dystopian fiction is partially targeting young adult readers, for example ”Feed” by M. T. Anderson and The Hunger Games trilogy by S. Collins [51], [7, p. 145].

2.1.5. Subtitles and Captions

The terms *subtitle* and *caption* will be defined from language learning perspective. The main assumption is that a viewer of an audiovisual media has different knowledge level in two languages [40]. It is expected that one of the languages is the first language of the viewer, yet it is not required. The audio track to the video is in the language in which the viewer has lower language understanding proficiency. The viewer sees a transcription of the currently spoken in the video speech displayed on the screen. If this text is in the language the viewer understands better, the text is called *subtitle*. If this text is in the language the viewer understands worse, and correspondingly is in the same language as the audio, the text is called *caption*.

Deaf or hard of hearing viewers are another type of audience for which the captions are created [28]. Hearing-impaired individuals need more contextual input in addition to a plain text transcript in order to fully understand what is happening in the video. Therefore captions displayed to such viewers may include descriptions of non-verbal information audible in the current video sequence. Examples of non-speech elements include speaker differentiation, background noises, audience reaction, sound effects, existence of music and so on.

There are two types of captions: closed captions and open captions [29]. Open captions are embedded into the video file and cannot be hidden. Closed captions (CC) are separate and the viewer can decide if the display of captions should be activated or not.

The audio tracks to the film adaptations are not used as a part of the dataset in this thesis. Hence it is not possible to call used transcripts *subtitles* as defined in the language learning context. The non-speech elements, if present, are filtered out as described in the Subsection 3.2.2. Lacking these features used transcripts do not qualify as captions if used in the context regarding viewers who are deaf or hard of hearing. Nonetheless, the textual transcripts used in this thesis will be further on referred to as *subtitles* for the purposes of readability.

2.1.6. Film Adaptations

A ***film adaptation*** is a conversion of a literary source (a novel, short story, etc.) into a film [45, p. 53]. The screenplay to the film is adopted rather than original, based on the already existing storyline and ideas. Adaptation implies change and differences due to the transition from one art medium to another [11, p. 5]. For example, the interior understanding of a character provided by the linguistic medium cannot be translated into a visual medium in an equally intelligible way. However, film adaptation may be promoted in such a way that the similarity to the original art piece is a part of the appeal for the film [39, p. 3], fidelity to the literary source being a widely discussed topic.

2.2. State of the Art

2.2.1. Linear Text Segmentation

A wide range of different approaches to address challenges of linear text segmentation was developed over the years. Methods based on existing knowledge from various fields, for example, linguistics, statistics, graph theory, etc. were used, recent years marked by an emerging interest in application of word embeddings and neural networks. Table 2.1 gives an overview of prominent linear text segmentation approaches along with their key features.

Most text segmentation methods can be categorized into two classes: similarity-based and generative models [37]. The ***similarity-based models*** are centered around a theory that the textual data in the same topic segment bears more resemblance to itself than the data in the preceding or consequent segment. This assumption is based on lexical cohesion theory which states that text segments with similar vocabulary are likely to be found in one coherent topic segment [24]. Text segmentation algorithms TextTiling [26], C99 [13] and LCseg [20] are examples of similarity-based models. Though GraphSeg [21] is a graph-based algorithm, it utilizes lexical similarity and therefore may also be considered similarity-based. The main assumption behind ***generative models*** is that a textual input consists of a hidden sequence of topics and it is possible to differentiate between the topic segments based on the characteristic probability distribution of words [37, 52]. Text segmentation algorithms U00 [61], Sun et al. [55], BayesSeg [17] and TopicTiling [50] are examples of generative models.

TextTiling [26] aims to determine whether there is a topic shift between two neighboring text segments based on term repetition. The algorithm starts with the division of text into pseudo-sentences of predefined length which are grouped together into blocks. Then the score determination step is executed in which the algorithm compares every two adjacent blocks in the input in a moving window fashion. Each block is treated as a bag of words for computing a score representing the amount of cohesion between the two blocks at hand. Low amount of cohesion signifies higher probability of a topic shift. Finally, the boundary detection is performed by filtering the potential segmentation points with the cohesion level lower than a predefined threshold. A detailed description of TextTiling algorithm can be found in Section 3.3.

C99 algorithm [13] is based on detecting segmentation boundaries by applying divisive hierarchical clustering to a rank matrix. First, the vector-space representations of sentences are used to compute a similarity matrix consisting of the pairwise cosine similarities between each pair of the sentences in the input document. Then the similarity values are converted to ranks to achieve local normalization. Finally, hierarchical clustering is used to determine the position of topic boundaries. The inside density consisting of a normalized sum of the rank values in a segment is computed for coherent text segments. The text is recursively split into smaller segments, split point being a potential boundary maximizing the density value, until the quality of segmentation stops showing significant improvements in comparison to the average segmentation quality results.

U00 algorithm [61] uses dynamic programming to find a minimum cost segmentation. Given a textual input, a gap between two adjacent words is considered a potential topic boundary. A graph is constructed with gaps as vertices and ordered connections between gaps as edges. This step is followed by cost calculation for each edge. The algorithm determines a minimum cost path in a graph from the first gap to the last gap by means of dynamic programming. As a result a segmentation is obtained in which separated text segments correspond to edges in a graph.

LCseg [20] is based on term repetition and uses lexical chains to compute the segmentation. First, a lexical chain containing all repetitions of a term in the entire document is constructed. Afterwards, a weighting scheme based on a derivation of term frequency-inverse document frequency (tf-idf) metric is applied. Longer chains are assigned lower weight than the shorter ones and chains with less repeated terms receive lower score. Similar to TextTiling a moving window of predefined size is used to scan the input. A lexical cohesion function is achieved by computing the lexical cohesion score at each transition between two windows. Lexical cohesion score is composed of cosine similarity using lexical chains overlapping with the two windows instead of using word counts. Once the algorithm finds a local minimum with strong cohesion function values at the neighboring points to the left and to the right, the hypothesized segmentation probability is computed. Conclusively, the points with highest segmentation probabilities which are larger than a threshold are considered final segmentation boundaries.

An approach introduced by Sun et al. [55] relies on mutual information and weighted mutual information to produce a segmentation and topic alignment between multiple documents. Text segmentation is seen as an optimization problem of finding such a segmentation and alignment that the loss of mutual information or weighted mutual information is minimal. The term weights are used as means of soft classification to decrease the effect of general or document-specific stop words and increase the influence of cue phrases on the segmentation process. The sentences sharing a topic are clustered together and the clusters are consequently used for alignment between documents of the dataset. The weighting of a term or a term cluster is based on entropy among documents or segments of documents.

BayesSeg [17] approach is based on using Bayesian framework to identify lexical cohesion. The words in every topic segment are considered to be draws from a multinomial language model corresponding to this topic segment. The probability mass of a segment concentrated on a small subset of words signifies the high likelihood of the language model of this segment. An objective function value of term frequency obtained from the probability perspective is maximized. Dynamic programming is applied to compute the final maximum likelihood segmentation.

A variation of the algorithm allows the integration of auxiliary information sources, for example cue phrases, without the need for additional labeling. By using this method it is possible to bias the sample selection towards the known cue phrases. In this case the draws are made from a special language model shared by all topics and documents in the dataset. A designed sampling-based inference technique is applied to produce a final segmentation.

TopicTiling [50] is a supervised text segmentation algorithm which uses the Bayesian Inference method of Latent Dirichlet Allocation (LDA). TopicTiling algorithm is based on a simplified version of TextTiling, however it uses topic IDs detected by the Inference method of LDA rather than term vectors for word representation. First, the documents in the training dataset are annotated with topic IDs. Due to the probabilistic nature of LDA an assignment of topic distribution in different runs may be inconsistent, therefore the most frequent topic ID assigned to the word is used as the word's topic ID. Then, topic IDs assigned to words are used to compute a score at each gap between two consecutive sentences. The final segmentation either returns a specified number of boundaries in case the number of segments was given as input or all boundaries with values larger than a certain threshold.

GraphSeg [21] is an unsupervised algorithm based on a semantic relatedness graphs. First, semantic relatedness between sentences is measured. Hence each word in one sentence is connected to each word in another sentence building a complete bipartite graph. The weights of the bipartite graph are represented by the cosine similarity measure between the word embedding vectors of each word pair. Hungarian method is applied to the graph in order to find a matching in which the sum of the edge weights is minimal. In the next step, given a set of word pairs provided by the Hungarian algorithm, the semantic relatedness measure between the sentences is computed. The measure is based on the cosine similarity and information content of the word pairs. Afterwards, each sentence in the text is considered a node in a relatedness graph and semantic relatedness measure is calculated for each sentence pair. An edge connects two sentences given the semantic relatedness measure between them is larger than a threshold. Finally, text segmentation is completed by locating maximal cliques of adjacent sentences in the constructed relatedness graph with the help of Bron-Kerbosch algorithm.

Sector [4] is an algorithm based on neural networks which performs text segmentation and topic label assignment using bi-directional Long Short-Term Memory (bi-LSTM) neural network. First, the sentence encoding based on Bloom filter compression method for sparse vectors is executed. Then, the assignment of a latent topic to each sentence is performed with the help of a two layer bi-LSTM neural network followed by an additional embedding layer with $tanh$ activation function. Consequently, the output layer with $softmax$ activation function for single topic labeling or $sigmoid$ activation function for multi-topic labeling decodes the potential text segment labels. Finally, the information from output layers and topic embeddings is used in order to perform final segmentation and topic classification.

2.2.2. Open Information Extraction

According to Niklaus et al. [46] OIE systems may be classified into categories of learning-based, rule-based, clause-based and systems capturing inter-proposition relationships. Recently multiple systems based on neural networks were introduced in an attempt to solve the task of OIE. Table 2.2 lists key features of OIE approaches described in this Subsection.

TextRunner [66] is a pioneering learning-based system in OIE. TextRunner uses limited automatically labeled data for training, therefore it is considered a self-supervised machine learning algorithm. First, given a small set of sentences a parser based on heuristic constrains automatically labels sentences with extractions. Independently, part of speech (POS) tags are assigned to sentences and noun phrases in a single pass over all documents in a corpus. Second, a sequence-labeling model is used to create a relational phrase extractor. Then, given a sentence as an input the candidate argument pairs consisting of noun phrases are identified. The relational phrase extractor decides whether to label the words between the parts of an argument pair as a relational phrase. Finally, an unsupervised probabilistic model is used to perform coreference resolution.

ReVerb [19] is a rule-based OIE system based on TextRunner. The main disadvantages of existing OIE systems at the time were uninformative and incoherent extractions. Therefore ReVerb introduced syntactic and lexical constraints in an attempt to reduce these unwanted extraction qualities. The extraction approach is centered around relation phrase, which means first specifying the relations satisfying the constraints and then finding corresponding arguments. Syntactic constraints are handcrafted Regular Expressions based on POS tags. Lexical constraints are based on assumption that a trustworthy relational phrase is encountered in connection to a considerable number of clearly defined argument pairs in a large corpora.

ClausIE [16] is a clause-based OIE system. ClausIE converts the input sentences into simplified independent clauses and uses clause constituents to generate the propositions. First, a dependency tree of a sentence representing its grammatical structure is generated. Second, the clauses in the input sentence are identified and the dependency relations are mapped to clause constituents. Consequently, a type of clause under consideration is determined based on the grammatical function of clause constituents. Finally, a conclusion is made about which constituents or combinations of constituents result in a proposition and a corresponding proposition is generated.

Stanford OIE [3] is an another example of clause-based OIE system. First, a classifier is learned by traversing a dependency tree and making a prediction about whether an edge produces an independent clause or not at each step. Thus the sentences are split into short utterances with the help of a learned classifier. Then, natural logic (a proof system based on human language syntax) is applied to the extracted utterances to maximally reduce them without loosing essential content. Finally, a small set of hand-crafted rules is used to split the shortened utterances into predicate-argument OIE triples.

Recurrent Neural Network OIE (RnnOIE) [54] is an OIE system based on neural network architecture. In this approach OIE extraction is addressed as sequence labeling problem. Given labeled dataset, supervised learning of a model by means of an bi-LSTM transducer architecture is performed. Multiple extraction tuples are encoded for a single sentence. A predicate is assigned multiple arguments making a tuple n-ary rather than binary. Custom BIO labels adapted from Semantic Role Labeling (SRL) systems are used to signify argument position and capture semantic meaning. An example of RnnOIE extraction along with BIO labeling can be seen in Table 3.4. Figure 3.6 depicts the architecture of RnnOIE system. More detailed description of RnnOIE can be found in Section 3.4.

Cui et al. [14] follow neural approach to generate binary OIE extractions. OIE extraction is seen as sequence-to-sequence generation problem with input sentence defined as the input sequence and the extracted OIE tuples with certain placeholders interpreted as output sequence. An RNN with an attention-based encoder-decoder Long Short-Term Memory (LSTM) architecture is applied to the input. An encoder encodes the input sequence in form of a context vector which is in turn used by the decoder to generate the output sequence of variable length.

2.2.3. Application of Literary Fiction and Film Subtitles

There are various ways in which NLP techniques are used to analyze fictional texts and film subtitles. One of NLP applications is representation of interpersonal relationships in a work of fiction by means of creating a fictional/social character network [34, 47, 58]. A fictional character network aims at detecting characters and representing information about them as well as discovering connections between the characters and expressing according relationships. Named Entity Resolution (NER), coreference resolution and anaphora resolution may be used in order to detect and identify the characters[15]. Fictional character networks may be used for assessment of the validity of literary theories, deciding on the level of historicity and realism, classification of fiction works, role detection, summarization, storyline detection and story segmentation [34]. Additionally, the corpora may be used to identify various key narrative features. For example, it may be applied to recreate the order and location of the events which took place [43], detect a mood of a film [56] or discover relevant scenes of a film without the need for analyzing the visual input [2, 27]. Finally, fictional literary texts are used to produce character summarization [69], chapter summarization [35] and novel summarization [30] due to the text length and narrative form.

Subtitles are used as a corpus of unstructured dialogue without annotation. Hence subtitles are segmented to identify whether consecutive sentences are part of the same dialogue turn [38] and may be annotated with a scene or speaker tag [68].

Manual translation performed by a professional human translator is a time-consuming task, therefore subtitles and literary fiction corpora are used as a component of machine translation [63, 64]. For example, machine translation systems based on existing subtitle datasets are used for translating subtitles into different languages [23, 41]. Another example is performing automatic translation of novels with the use of models trained on literary fiction as a supporting tool in a translation process [57].

algorithm	publication year	supervised	similarity-based	generative	key features
TextTiling [26]	1997	no[1]	✓		lexical co-occurance
C99 [13]	2000	no[2]	✓		divisive hierarchical clustering, ranking matrix
U00 [61]	2001	no[2]		✓	minimum cost segmentation, dynamic programming
LCSeg [20]	2003	no[2]	✓		TextTiling-based, lexical chains
Sun et al. [55]	2007	no[3]		✓	mutual information, dynamic programming
BayesSeg [17]	2008	no[3]		✓	Bayesian framework, incorporating cue phrases, dynamic programming
TopicTiling [50]	2012	yes		✓	LDA-based
GraphSeg [21]	2016	no	✓		semantic relatedness graph, word embeddings
Sector [4]	2019	no			LSTM neural network, topic labeling

Table 2.1.: Classification and key features of notable linear text segmentation algorithms. One could argue that the algorithms with superscript may be considered semi-supervised as they expect additional information about the input.
[1] TextTiling expects the paragraph structure of the input text is given.
[2] There exists a variation of the algorithm which expects the number of topic segments is given.
[3] In the presented evaluation of the algorithm in the paper it is assumed the number of topic segments is known.

OIE system	publication year	key features
TextRunner [66]	2007	• learning-based • shallow syntactic analysis (POS tags, NP chunks) • labeling training examples with a set of handcrafted patterns
ReVerb [19]	2011	• rule-based • shallow syntactic analysis (POS tags, NP chunks) • lexical and semantic constraints
ClausIE [16]	2013	• clause-based • dependency parsing • detecting clauses and identifying clause types • no training required
Stanford OIE [3]	2015	• clause-based • dependency parsing • minimization of extracted clauses • extraction based on handcrafted patterns
RnnOIE [54]	2018	• neural-based • bi-LSTM transducer for supervised model training • extracts n-ary relational tuples • OIE as a sequence labeling problem
Cui et al. [14]	2018	• neural-based • encoder-decoder LSTM RNN for supervised model training • extracts binary relational tuples • OIE as a sequence-to-sequence generation problem

Table 2.2.: Key features of notable OIE systems.

3. Method

This Chapter provides necessary information about the dataset creation and preprocessing along with detailed description of NLP methods and algorithms used in this thesis. First, a general overview of the dataset is given in Section 3.1. The dataset consists of two parts, Novels and Subtitles. The Novels part of the dataset is described in Subsection 3.1.1 followed by the Subtitles part in Subsection 3.1.2. The motivation for using this type of dataset is explained in Subsection 3.1.3. Second, the preprocessing steps performed on the dataset are presented in Section 3.2 in which Subsection 3.2.1 is dedicated to Novels dataset part and Subsection 3.2.2 to Subtitles dataset part. Finally, the parts of the linear text segmentation algorithm TextTiling are presented in Section 3.3. Neural OIE model used for the generation of propositions in this work is described in Section 3.4. The manner in which aforementioned OIE and linear text segmentation methods are connected is shown in Section 3.5.

3.1. Dataset Description

The dataset for this thesis contains two parts. The first part of the dataset includes novels as described in Subsection 3.1.1, the second part consists of subtitles to film adaptations of these novels presented in Subsection 3.1.2. The language of the dataset is English. The motivation behind the dataset choice is explained in Subsection 3.1.3.

3.1.1. Dataset Part 1: Novels

The Novels dataset includes 11 plain text files, see Table 3.1 for an overview.

	Novel title	Author	Original language	Publication year
1	Nineteen Eighty-Four: A Novel	G. Orwell	English	1949
2	Brave New World	A. Huxley	English	1932
3	We	Y. Zamyatin	Russian	written in 1920, translated into English in 1924
4	The Handmaid's Tale	M. Atwood	English	1985
5	Do Androids Dream of Electric Sheep?	P. K. Dick	English	1968
6	The Hunger Games	S. Collins	English	2008
7	Catching Fire	S. Collins	English	2009
8	Mockingjay	S. Collins	English	2010
9	The Giver	L. Lowry	English	1993
10	The Maze Runner	J. Dashner	English	2009
11	Ready Player One	E. Cline	English	2011

Table 3.1.: Overview of Novels dataset part. All novels satisfy two criteria. First, the literature genre of a novel is dystopian fiction. Second, there exists a screen adaptation of the novel which can be used in the Subtitles dataset. Only fictional text of the novels in plain text format is used as input for text segmentation task.

The files include the text constituting the novel and therefore do not contain non-fictional parts of the book such as dedication, acknowledgments, table of contents, etc. All novels share the same literature genre of dystopian fiction.

A novel “We” by Y. Zamyatin was originally written in Russian and then translated into English. “The Handmaid's Tale” was written by Canadian author M. Atwood. ”Nineteen eighty-four” and ”Brave New World” were written by English authors G. Orwell and A. Huxley respectively. Remaining seven novels were written by authors from USA. Though the majority of novels in the dataset were created by different authors, three novels were written by S. Collins and compose The Hunger Games trilogy.

The novels vary in publication date, the oldest one being “We” by Y. Zamyatin written in Russian in 1920, published in English in 1924 and the newest one being “Ready Player One” by E. Cline published in 2011.

In order to enable the evaluation of linear text segmentation algorithm, it is essential that a novel contains chapters, see Chapter 3.2.1 for more detailed explanation. The novel “Fahrenheit 451” by R. Bradbury, for example, could not be included into the dataset since despite being structurally separated into three parts, the parts do not include chapters.

3.1.2. Dataset Part 2: Subtitles

The Subtitles dataset consists of 13 SubRip subtitle files, see Table 3.2 for an overview. These files are going to be referred to as SRT files further on and are subtitle files in .srt format.

Table 3.3 shows an example of two consequent subtitle sequences following SRT format structure. A single sequence consists of four components, which are a sequence number counter, beginning and end time codes for subtitle display on screen, sequence text and a blank line. Beginning time code defines when the subtitle text starts to be visible on screen. It is followed by a symbolic arrow ”-->” leading to the end time code, which is the time when the sequence text disappears from the screen. Time codes use milliseconds precision in hh:mm:ss,sss format. Subtitle text can include more than one line.

The reason for an unequal number of novels and subtitles is due to the film adaptation aspects. The Subtitles dataset contains subtitles to two different screen adaptations of the novel “Nineteen Eighty-Four” by G. Orwell, one produced in 1956 and another one in 1984. The screen adaptation of a single book “Mockingjay” by S. Collins consists of two parts, resulting into two separate subtitle files. Therefore the Subtitles dataset has two items more than the Novels dataset.

The film ”We” was produced in Germany and therefore the subtitles are a translation from German into English. Both screen adaptations of the book ”Nineteen Eighty-Four: A Novel” from year 1956 and 1984 were produced in the UK. Remaining screen adaptations originate from USA.

	Film title	Director	Original language	Release year
1	Nineteen Eighty-Four	M. Anderson	English	1956
2	Nineteen Eighty-Four	M. Radford	English	1984
3	Brave New World	L. Libman, L. Williams	English	1998
4	We	V. Jasný	German	1982
5	The Handmaid's Tale	V. Schlöndorff	English	1990
6	Blade Runner	R. Scott	English	1982
7	The Hunger Games	G. Ross	English	2012
8	The Hunger Games: Catching Fire	F. Lawrence	English	2013
9	The Hunger Games: Mockingjay - Part 1	F. Lawrence	English	2014
10	The Hunger Games: Mockingjay - Part 2	F. Lawrence	English	2015
11	The Giver	P. Noyce	English	2014
12	The Maze Runner	W. Ball	English	2014
13	Ready Player One	S. Spielberg	English	2018

Table 3.2.: Overview of Subtitles dataset part. Every film is a screen adaptation of a novel included in the Novel dataset part. All film subtitles are SRT files.

The release year of the films does not vary as significantly as the one of novels. The oldest film is ”Nineteen Eighty-Four” directed by M. Anderson in 1956. The newest film is ”Ready Player One” directed by S. Spielberg in 2018.

3.1.3. Motivation for Choosing Dataset

Initially only subtitles were seen as a dataset. However, using solely subtitles introduces the issue of limited data for word embedding training. The quality of word embedding model is dependent on the size of the training corpus, the larger the corpus the higher the word embeddings quality [36]. The combination of subtitles and novels expands the number of words significantly. See Figure 3.1

```
1
00:02:17,440 --> 00:02:20,375
Your task seems impossible to me.

2
00:02:20,476 --> 00:02:22,501
The situation is critical at the moment.
I still have to try.
```

Table 3.3.: An example of two consequent subtitle sequences in an SRT file. A single subtitle sequence consists of its numeric counter, time at which the sequence text should be displayed at the screen and time at which it should be hidden, sequence text and a blank line signifying the end of the current sequence. Time codes are separated by an arrow --> and use millisecond precision.

for the comparison between the overall number of words in a novel and in a corresponding subtitle file. The longest novel is ”Ready Player One” by E. Cline containing 140 721 words, the shortest novel is ”The Giver” by L. Lowry with 44 790 words. The longest subtitle file is ”Ready Player One” with 10 863 words, the shortest subtitle file is ”Blade Runner”, a film adaptation of a novel ”Do Androids Dream of Electric Sheep?”, with 4 303 words.

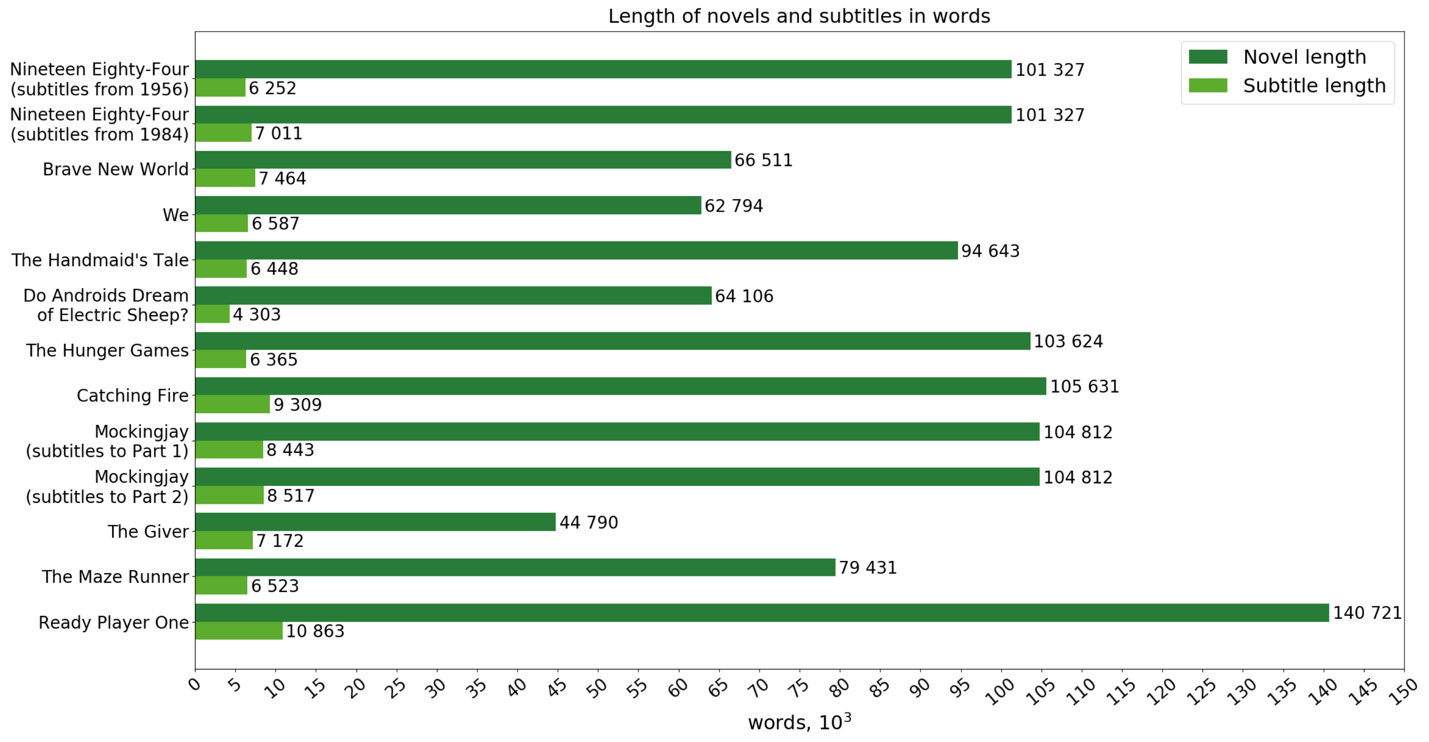


Figure 3.1.: The comparison between the overall number of words in a novel (dark green) and corresponding subtitle (green). The number includes non unique words as well as stop words.

Dataset used for this thesis holds properties which cannot be found in expository or descriptive texts. Fictional narrative texts use synonyms rather than word repetitions. Dystopian literature and films often contain made up terms which are not limited to only proper names. Outside of fiction context such vocabulary may be compared to words with typos or slang words which may not be a part of a formal vocabulary, but are not uncommon in casual communication.

Moreover, within creative writing certain liberties in expression of thoughts and opinions as well as critique of culture, society and political situation may be introduced. In case of dystopian setting already existing words may obtain fictional ironic meaning. For example, "Ministry of Love" in the novel "Nineteen Eighty-Four" by G. Orwell is more related to misery than to positive feeling of affection and "peacekeepers" in the Hunger Games trilogy by S. Collins are an instrument of repression.

The length of paragraphs and sentences is irregular and not easily predictable. Both novels and subtitles make heavy use of dialogues. Dialogues introduce rapid switches between characters and topics as well as punctuation challenges such as incomplete sentences ending with ellipsis.

Given described theoretical background a set of algorithms is chosen to perform the task at hand. This chapter provides a detailed overview of these algorithms as well as information about selection, preprocessing and transformation of input data. Figure 3.2 shows the pipeline of performed work.

3.2. Dataset Preprocessing

In order to evaluate the linear text segmentation performance it is necessary to generate ground truth against which the results are compared. The ground truth for the linear text segmentation task at hand should consist of topic shift positions in the input.

Relevant information about dataset files is computed once and is cached in form of binary files for future usage by TextTiling or RnnOIE. These binary files include chapters and paragraphs of novels, scenes of subtitles, word tokens and token weights computed by RnnOIE. Statistical data about the dataset file such as length of the sentences and paragraphs as well as vocabulary size is also stored.

To calculate WindowDiff and P_k metric it is necessary to know if a token begins a new topic segment, therefore this information is calculated and stored for each token as well. More detailed information about the preprocessing of novels can be found in Subsection 3.2.1. Subsection 3.2.2 describes the preprocessing process for subtitles.

3.2.1. Preprocessing of Novels

The narrative element of a novel, which is expected to satisfy all qualities of a topic segment according to definition in Chapter 3.3, is a chapter. The naive approach of separating a text into paragraphs and declaring them topic segments is not feasible due to the fact that novels include multiple dialogues. Formatting of a dialogue requires that a new paragraph starts as soon as speaker change occurs, which does not correspond to the idea of a single segment fully enclosing a certain topic.

For Novels dataset Regular Expressions are used to find and remove structural patterns not related to topic shifts. These Regular Expressions can be found in Appendix A.1. Remaining textual data is separated into text segments, each of them representing a chapter. The beginning of such text segment is considered to be a topic shift position and is used for the evaluation of linear text segmentation algorithm as described in Chapter 4.

All the headlines signifying the beginning of a novel part and its numeration , for example ”Part 1”, are removed. The title of a part, if present, represents an idea shared by more than one chapter. Therefore it cannot be appended as an additional textual input to one unique block of text and is taken out.

The headers containing the word ”chapter” along with its ordinal number are removed. Though the book ”We” by Y. Zamyatin is separated into sequences labeled ”records” by author, they are treated as equal to chapters as in terms of structure they represent the same concept. Another text separation techniques such as epilogue or dinkus (three or more asterisks in a row) are also considered to be equivalent to a chapter in terms of topic shift and are treated in the same manner as chapters. The title of a chapter, if present, is viewed as the first sentence in a chapter, this way it is considered to be a piece of the chapter text.

The novel ”The Handmaid’s Tale” by M. Atwood includes a special case of novel structure not shared by other novels from the dataset. One chapter or more can be preceded by a title in order to create a separate section and signify a broader theme. Sections which closely relate to the main character are titled ”Night”, the other sections named for example ”Waiting Room” or ”Household” tell more general story. Though such sections are not the same as parts from the literary perspective, they are treated in the same way for the purpose of text processing, which means the ordinal number and title of the section are removed throughout the text file.

3.2.2. Preprocessing of Subtitles

Subtitles used in this thesis essentially are transcripts of dialogue in the film. On one hand subtitles in this form do not have any structural anchors created by author similar to paragraphs or chapters in novels. On the other hand a film consists of sequential non-overlapping scenes. A change from one scene to another is happening if the situation or a unit of a dialogue has come to an end. This largely corresponds with the definition of a subtopic segment as defined in Chapter 3.3. An assumption is made that if there occurs a pause in action during which no dialogue is spoken then a scene switch has taken place. This way a subtopic segment starts either at the beginning of the subtitle text or after a pause which lasts five seconds or longer.

In Subtitles dataset the textual information not relevant to the subtopic is removed. Regular Expressions are used to filter out occasionally used SRT formatting tags. These Regular Expressions can be found in Appendix A.2. Nonverbal information not audible in the audio track intended for deaf or hard of hearing people is separated out. This includes speaker identification and symbols signifying the beginning and ending of music. The song lyrics are left in place if they are directly connected to the plot. For example the lyrics of a song ”The Hanging Tree” in The Hunger Games trilogy is a part of the narration, but a song ”Julia” by band Eurythmics featured in the film ”1984” released in 1984 is not. The filtering is done manually in cases when using Regular Expressions to find the uniform formatting pattern to describe the non-speech elements is error-prone.

3.3. TextTiling Algorithm

The linear text segmentation algorithm used in this thesis is based on a paper "TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages" by M. Hearst [26].

TextTiling is one of the early similarity-based linear text segmentation algorithms. However, it is not uncommon to use approaches closely resembling TextTiling as a base for new text segmentation methods, for example LCSeg [20], TSF [32] and TopicTiling [50] utilize TextTiling approach to build their ideas upon it. The ingenuous design of TextTiling algorithm leaves room for experimentation simultaneously providing robust linguistic foundation. These qualities make this algorithm an exceptional candidate for NLP enhancement presented in this thesis without the need for creating a completely new algorithm from scratch.

This section describes the original TextTiling algorithm along with changes or enhancements performed as a part of this work. One of such alterations is the application of TextTiling to the dataset consisting of fictional text, see Section 3.1 for more information about choosing the dataset and detailed dataset description. Initially, fictional texts were not taken into consideration as potential input type for TextTiling.

Original TextTiling algorithm takes expository texts with minimal styling, such as headings, as an input. It is required that the input text consists of paragraphs. An expository text contains one or more main topic discussions, which can be overlapping. However it is assumed that such text can be presented as a sequence of non-overlapping continuous subtopic segments.

A subtopic segment has following properties:

- represents one single subject of discussion
- does not overlap with other subtopic segments
- consists of a continuous text segment
- includes one or more paragraphs

The goal of the TextTiling algorithm is to find the precise location of transition from one subtopic segment to another, which is called a ***subtopic shift***. The algorithm consists of thee parts: tokenization, score calculation and boundary identification. It is required to provide parameter values for token-sequence size and block size to perform linear text segmentation with the help of TextTiling.

3.3.1. Tokenization

First, the whole input text is converted into a single sequence of tokens. The process of tokenization includes conversion of words to lower case and lemmatization. Afterwards this single sequence is separated into consecutive multiple ***token-sequences*** of size w (predefined parameter of the TextTiling algorithm). The parameter w is supposed to approximate the length of a sentence. The motivation for using parameter w is that real sentences may vary significantly in length and therefore have a potential to produce incomparable scores. Stop words are filtered out from token-sequences.

In this thesis the text is separated into tokens based on the RnnOIE tokenization rules, see Section 3.5 for more details. The stop word list used for this thesis is an extended version of `nltk.corpus` stop words list. It can be seen in Appendix B.

3.3.2. Score Calculation

In the original paper it is shown that the blocks score calculation method produces better results than vocabulary introduction method. Therefore in this thesis the blocks score calculation is used and only this method will be described in detail.

Lexical Score

A ***block*** is obtained by taking k consequent token-sequences. Predefined parameter k approximates average number of sentences in a paragraph in a similar fashion as the parameter w approximates sentence length. It is worth mentioning that the size of a single block in tokens can be less than a number obtained by multiplying token-sequence size by block size because the stop words are filtered out from token-sequences.

The change of subtopic in text is presumed to be accompanied by a change in vocabulary. Therefore computation of cosine similarity between blocks (lexical score) is done by observing how different is the vocabulary of the adjacent blocks. A transition from $tokensequence_i$ to the $tokensequence_{i+1}$ is called a ***token-sequence gap***. A lexical score is calculated at each token-sequence gap. Therefore given n

token-sequences, there are $n - 1$ token-sequence gaps and correspondingly $n - 1$ lexical scores.

Following steps are necessary to calculate the lexical score at a token-sequence gap. First, two blocks consisting of k token-sequences each are formed: $block_1 = \{tokensequence_i, \ldots, tokensequence_{i+k}\}$ and $block_2 = \{tokensequence_{i+k+1}, \ldots, tokensequence_{2k+1}\}$, here the numeration begins with 1 and i is an incrementing index. Second, a common vocabulary consisting of all tokens occurring in both blocks is calculated. A weight of a token $w_{t,b1}$ is the frequency of a vocabulary token t within a $block_1$. In conclusion, the lexical score at the token-sequence gap between blocks $block_1$ and $block_2$ is measured by their cosine similarity according to the formula 3.1:

$$score = \frac{\sum_t w_{t,b1} w_{t,b2}}{\sqrt{\sum_t w_{t,b1}^2 w_{t,b2}^2}} \tag{3.1}$$

After the lexical score is calculated, the counter i is incremented by one. This way the comparison between the blocks is done using the principle of moving window with a step of one token-sequence allowing to calculate the lexical similarity at each token-sequence gap.

Though one could assume that there is no subtopic shift for at least some arbitrary number of token-sequences in expository text, there is no certainty that this assumption is true for fictional literature and subtitles. Therefore to calculate a lexical score at each token-sequence gap a number of comparisons at the beginning and at the end of computation includes blocks with less token-sequences than k. The first comparison takes place between a block consisting out of one token-sequence and a block of size k, the second comparison is between a block out of two token-sequences and a block of size k and so on. The number of such smaller blocks at the beginning and end of calculation depends on the value of parameter k.

Depth Score

As can be seen in the Figure 3.3, lexical scores graph has high points called ***peaks*** and low points called ***valleys***. High peak tops on both sides of a token-sequence gap create a deep valley at this token-sequence gap. Such a valley signals strong change in the vocabulary, because the lexical similarity of the blocks first decreases and then increases. In order to determine whether a subtopic change has taken place a ***depth score*** is calculated at each token-sequence gap. To calculate the depth score of a token-sequence gap_j the lexical scores of the gaps to the left of it $gaps\_left = score(gap_{j-1}), \dots, score(gap_0)$ and of the gaps to the right of it $gaps\_right = score(gap_{j+1}), \dots, score(gap_m)$ are observed. Here $m + 1$ is the overall number of token-sequence gaps. To find the highest neighboring peak to the left the values of $gaps\_left$ are scanned one by one from $score(gap_{j-1})$ to $score(gap_0)$ until the first peak at gap_l which satisfies the condition 3.2 is found.

$$\begin{aligned} &\exists gap_l : score(gap_l) \in gaps\_left\ and \\ &score(gap_l) > score(gap_j)\ and \\ &score(gap_l) > score(gap_{l-1}) \end{aligned} \tag{3.2}$$

The highest neighboring peak to the right is computed in an analogous way with ascending gap index and should satisfy the condition 3.3.

$$\begin{aligned} &\exists gap_r : score(gap_r) \in gaps\_right\ and \\ &score(gap_r) > score(gap_j)\ and \\ &score(gap_r) > score(gap_{r+1}) \end{aligned} \tag{3.3}$$

After both gap_l and gap_r have been identified, the depth score of gap_j is calculated using the formula 3.4.

$$\begin{aligned} depth\_score(gap_j) = &score(gap_l) - score(gap_j) + \\ &score(gap_r) - score(gap_j) \end{aligned} \tag{3.4}$$

Score Smoothing

Peaks in the lexical scores which are directly followed or preceded by higher peaks are interfering with the process of finding a highest neighboring peak for computing depth scores. See Figure 3.4 for an example. To avoid such behavior average smoothing of lexical scores is introduced. Fist, smoothing width size s and number of smoothing rounds r are chosen. Then for each token-sequence gap gap following steps are repeated r times:

1. find lexical the score at gap
2. find the lexical scores of the $s/2$ gaps to the left of gap
3. find the lexical scores of the $s/2$ gaps to the right of gap
4. calculate the average of the lexical scores from steps 1, 2, 3 and assign its value to gap

To enable the calculation of lexical score smoothing for the first and last token-sequence gap in this work $s/2$ zeros are added at the beginning and at the end of the lexical scores list (zero padding).

For the dataset used in this work smoothing improves the evaluation results in some cases, but it fails to consequently improve the results for all files in the dataset. For this reason no smoothing was used in this thesis.

3.3.3. Boundary Identification

The depth scores with values over a certain threshold are considered potential segment boundaries. In case the depth scores are distributed normally the suggested threshold value is calculated by either subtraction of standard deviation from mean called the liberal measure LC or subtraction of half of the value of standard deviation from mean called conservative measure HC.

The original paper makes an assumption that depth scores are distributed normally, however this does not hold for the dataset used in this thesis. The distribution of depth scores of novels and films is tested with help of `shapiro` and `normaltest` from `scipy.stats` package. Depending on the input parameters the distribution in some cases fits gamma probability distribution as can be seen in Figure 3.5, nonetheless it is not true for all cases. Therefore the threshold is set to the median value of depth score data.

Another assumption is made that final segment boundaries should be rather sparse than dense. Therefore only the depth scores located at least three token-sequences apart from each other should be taken into consideration.

It is possible that detected potential subtopics are not located exactly at the beginning of a paragraph. In such case the location of a subtopic boundary is moved to the original text paragraph structure. Given a token-sequence gap that has been identified as a potential subtopic shift first the closest paragraph break to the left of it is examined. This paragraph break is marked as a subtopic shift if it has not been selected as such previously. In case it has already been selected, the closest paragraph break to the right of the token-sequence gap is considered in the same way. In case both closest neighboring paragraph breaks have been identified as subtopic boundaries, the second closest paragraph break to the left of the token-sequence gap is taken into consideration. This paragraph break is required to be at least three token-sequences away from the previously observed paragraph break on the left-hand side. The same steps are performed on the right-hand side if the second closest paragraph break to the left has been marked as a valid subtopic shift in previous steps. If there was no success in approximation of found subtopic boundary to the paragraph structure after observing four closest neighboring paragraph breaks this subtopic boundary is discarded. A final list of detected boundaries consists out of all paragraph breaks which were identified as subtopic shifts.

The Subtitles dataset is missing the paragraph structure by definition, therefore approximation to original paragraph structure cannot be implemented and is skipped during code execution for this type of input.

3.4. Recurrent Neural Network Open Information Extraction (RnnOIE)

In order to perform OIE for this thesis the version of Recurrent Neural Network Open Information Extraction (RnnOIE) implemented by Allen Institute for Artificial Intelligence (AllenAI) is used. A `Predictor` class is imported from Python library `allennlp.predictors.predictor` and is instantiated by a model provided by AllenAI [1]. Original RnnOIE is defined in the paper “Supervised Open Information Extraction” by Stanovsky et al. [54]. Though the AllenAI version of RnnOIE contains slight differences to original RnnOIE in labeling notation, these dissimilarities do not influence the functioning of the model. Therefore though the reimplemented RnnOIE model will be described, it would be referred to as RnnOIE further on for better readability.

By applying RnnOIE a predicate-argument structure of a sentence is obtained. Given a sentence $s = \{word_0, \ldots, word_n\}$ RnnOIE extracts a set of correspond-

[1] https://s3-us-west-2.amazonaws.com/allennlp/models/openie-model.2018-08-20.tar.gz

ing tuples. Each tuple represents a proposition stated in a given sentence s. A ***subspan*** x_i of a sentence s consists of one or more adjacent words from it $x_i = \{word_k, \ldots, word_{k+y}\}$. A single extracted tuple $t_j = \{x_0, \ldots, x_m\}$ consists of at least two subspans of s. One subspan of a tuple is marked as a predicate, while all the other subspans are marked as arguments. Tuples of a sentence can overlap with each other, therefore multiple tuples can share the same predicate. A predicate may include modal auxiliary verbs and embedded predicates and therefore it is possible for a predicate to consist out of more than one word.

RnnOIE uses custom BIO tagging. Beginning (B) is assigned to the words at the beginning of a subspan. Inside (I) is attributed to other words inside of the subspan. Outside (O) is a tag for words of the sentence which are not a part of a tuple under consideration. An example of an RnnOIE extraction with BIO tagging can be seen in the Table 3.4.

Input sentence:
The sun went down and the dark-gray clouds changed color.
Extracted tuples:
1) [ARG0: The sun] [V: went] [ARG1: down]

The$_{\text{B-ARG0}}$ sun$_{\text{I-ARG0}}$ went$_{\text{B-V}}$ down$_{\text{B-ARG1}}$ and$_{\text{O}}$ the$_{\text{O}}$ dark$_{\text{O}}$ -$_{\text{O}}$ gray$_{\text{O}}$ clouds$_{\text{O}}$ changed$_{\text{O}}$ color$_{\text{O}}$
2) [ARG0: the dark - gray clouds] [V: changed] [ARG1: color]

The$_{\text{O}}$ sun$_{\text{O}}$ went$_{\text{O}}$ down$_{\text{O}}$ and$_{\text{O}}$ the$_{\text{B-ARG0}}$ dark$_{\text{I-ARG0}}$ -$_{\text{I-ARG0}}$ gray$_{\text{I-ARG0}}$ clouds$_{\text{I-ARG0}}$ changed$_{\text{B-V}}$ color$_{\text{B-ARG1}}$

Table 3.4.: An example of RnnOIE extraction tags. RnnOIE produces two tuples for the input sentence. First tuple consists out of predicate "went" as signified by a label V (verb) and two arguments ARG0 and ARG1, each being a subspan of original sentence. Second tuple contains predicate "changed" and two corresponding arguments. Under the dashed lines the labels for single tokens are shown. According to BIO-tagging additional letter B (beginning), I (inside) or O (outside) are added to predicate and argument information. In the second tuple a token "the" is at the beginning of ARG0 and therefore is tagged B-ARG0, "dark" is inside of an argument and is tagged I-ARG0, "sun" is outside of a predicate or any argument in this tuple and is tagged "O".

RnnOIE is a model obtained by supervised training of a RNN with a bi-LSTM architecture. First, feature vectors, which are going to be used as an input to the neural network, are prepared. A feature vector is computed for each word in the sentence according to the formula 3.5, where $\oplus$ stands for concatenation.

$$\begin{aligned} feature\_vector(word_i) =& emb(word_i) \oplus emb(POS(word_i)) \oplus \\ & emb(p) \oplus emb(POS(p)) \end{aligned} \tag{3.5}$$

Given a sentence $s = \{word_0, \ldots, word_n\}$ each $word_i, i \in 0 \ldots n$ is mapped to a vector which is a corresponding pre-trained 300-dimentional GloVe word embedding $emb(word_i)$. A POS of $word_i$ is encoded as a five-dimentional word embedding $emb(POS(word_i))$. A tuple t contains a single word which can be identified as predicate's syntactic head p. Therefore a feature vector of w_i is constructed by concatenating word embedding of w_i, word embedding of POS of w_i as well as word embedding of predicate's syntactic head and word embedding of its POS. Then the input embeddings are transformed into contextualized output embeddings by bi-directional deep LSTM transducer [22]. These output embeddings are passed as an input into softmax function. Finally, the softmax function produces independent probability distribution over possible tags for each word. The architecture of the network can be seen in Figure 3.6.

3.5. Combining RnnOIE and TextTiling

On one hand, TextTiling is based on a textual representation of the input. For the purpose of enhancing linear text segmentation with context information word embeddings are used.

This work evaluates the effect of substituting words with word embeddings vectors on the performance of given linear text segmentation algorithm. Replacing a word with a word embedding makes it possible that words with similar meaning are represented by similar vectors [42].

On the other hand, the original TextTiling approach makes use of lexical co-occurrence. This way the algorithm relies on semantic proximity based on word repetition without taking the syntactic meaning of words in the sentence into account. In order to evaluate the benefit of using such information in division of a fictional text into meaningful units this work concentrates on enriching linear text segmentation process with OIE representations of the same input.

In case of TextTiling with word embeddings, input tokens are replaced by their vector representations at the step of splitting input into token sequences. In this work Word2Vec word embedding algorithm [42] from the module `gensim.models` is used.

Usage of pre-trained word embeddings such as provided by Google [2] proved to be impractical for the dataset used in this thesis. Fictional texts, especially of dystopian genre, contain terms or sometimes even languages made up by the author which are not a part of English vocabulary used to train such word embedding models. A novel and its film adaptation share the fictional terms and proper names. Hence the Word2Vec model is first trained on the sentences from a novel and then the same model is additionally trained on subtitle sentences to slightly expand training corpus size. Stop words are not included in the training set to create a stronger pull between vectors of relevant words. Following parameters were used to generate the model: the length of the word embedding is 300, the size of the dynamic context window size is five, minimum number of words considered during training is one, used training algorithm is skip-gram. The training phase of the Word2Vec algorithm is not fully deterministic. Therefore generated model is stored in a file and used for evaluation with different TextTiling parameters to avoid using different vector representations of the same input for evaluation.

After the word embeddings are generated they are applied to TextTiling. Given a block b of size n such that $b = \{v_{t_1}, \ldots, v_{t_n}\}$, where v_{t_1} stands for a Word2Vec vector of a token with an index one in this block, vector representation of this block v_b is computed according to the formula $v_b = \sum_{i=1}^{n} v_i$. The vector length of different blocks stays constant for a single algorithm run and therefore cosine similarity is calculated between block vectors of the same length.

TextTiling and RnnOIE are combined by applying weights to word embeddings. Given a sentence as an input, RnnOIE extracts propositions stated in the sentence in form of n-ary tuples. It is possible that a word token is not a part of any tuple or that multiple tuples share the same word token. In this work it is assumed that the appearance or absence of a word token in an OIE tuple corresponds to the strength of syntactic meaning of this word token in the given sentence. Therefore an overall number of occurrences of a certain word token in all extracted tuples of a single sentence is seen as token's weight. Stop words listed in Appendix B.1 are automatically assigned a weight of zero because they carry little semantic

[2]https://code.google.com/archive/p/word2vec/

meaning. By applying a weight to a word's vector its direction stays the same, but the magnitude is increased or decreased.

Figure C.1 shows an example of RnnOIE extraction in JSON format given the input sentence "The sun went down and the dark-gray clouds changed color." The key "words" contains a list of extracted tokens for given sentence. RnnOIE uses a custom tokenizer based on `spaCy` tokenizer which follows slightly different rules than the tokenizers initially used in this work (`word_tokenize` or `TweetTokenizer` from `nltk.tokenize`). Using different tokenizers leads to unequal number of extracted tokens which introduces difficulties for matching tokens with their weights produced by RnnOIE. For example, as can be seen in Figure C.1, a word "dark-gray" is separated by RnnOIE tokenizer into three tokens "dark", "-", "gray", however both `TweetTokenizer` and `nltk.tokenizer` extract only one token "dark-gray". In order to reduce potential errors custom RnnOIE tokenizer is used throughout this work.

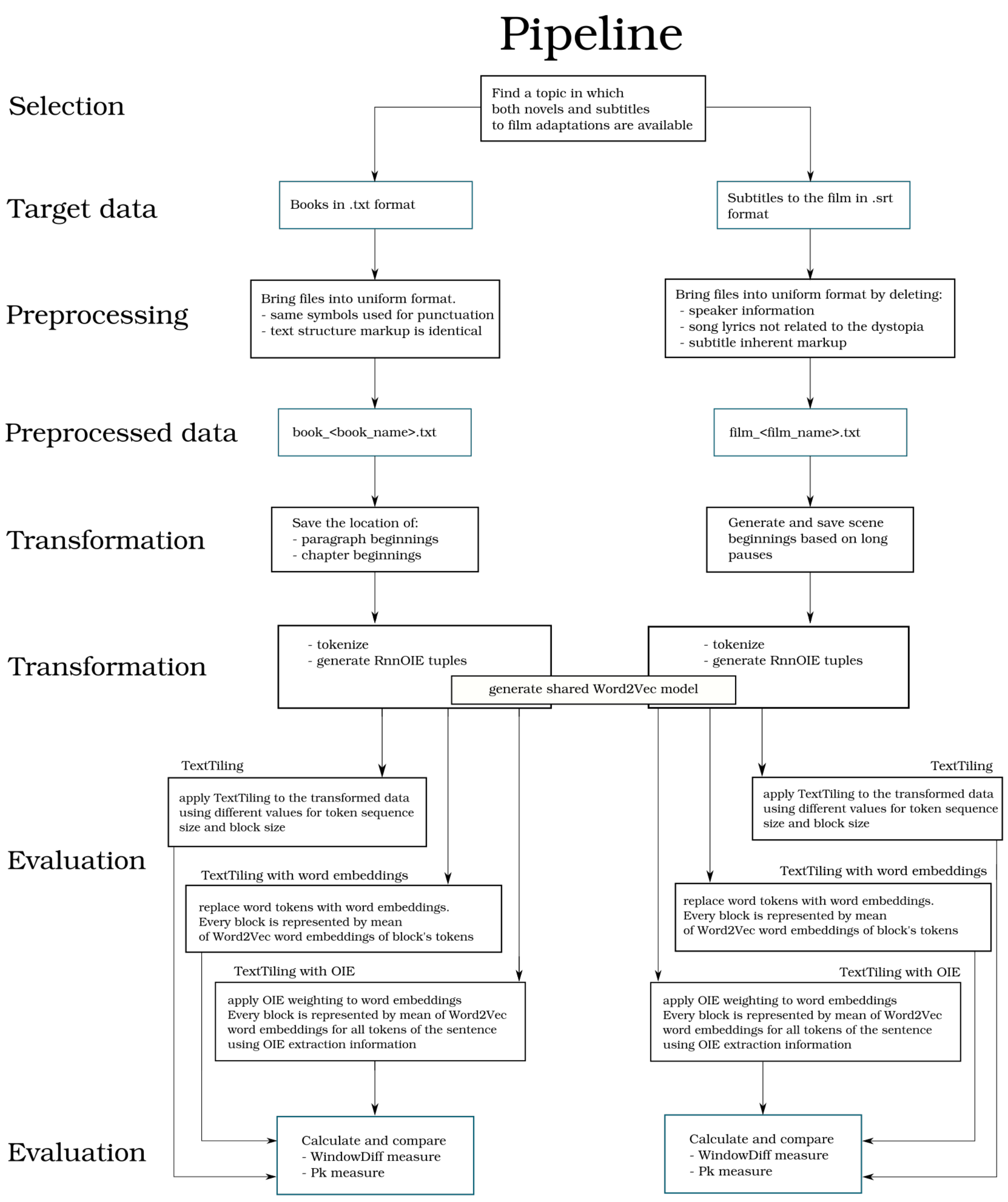

Pipeline
Selection
Find a topic in which both novels and subtitles to film adaptations are available
Target data
Books in .txt format
Subtitles to the film in .srt format
Preprocessing
Bring files into uniform format.
- same symbols used for punctuation
- text structure markup is identical
Bring files into uniform format by deleting:
- speaker information
- song lyrics not related to the dystopia
- subtitle inherent markup
Preprocessed data
book_<book_name>.txt
film_<film_name>.txt
Transformation
Save the location of:
- paragraph beginnings
- chapter beginnings
Generate and save scene beginnings based on long pauses
Transformation
- tokenize
- generate RnnOIE tuples
generate shared Word2Vec model
Evaluation
TextTiling
apply TextTiling to the transformed data using different values for token sequence size and block size
TextTiling with word embeddings
replace word tokens with word embeddings. Every block is represented by mean of Word2Vec word embeddings of block's tokens
TextTiling with OIE
apply OIE weighting to word embeddings Every block is represented by mean of Word2Vec word embeddings for all tokens of the sentence using OIE extraction information
Evaluation
Calculate and compare
- WindowDiff measure
- Pk measure

Figure 3.2.: Pipeline of performed work for the novels (left branch) and films (right branch).

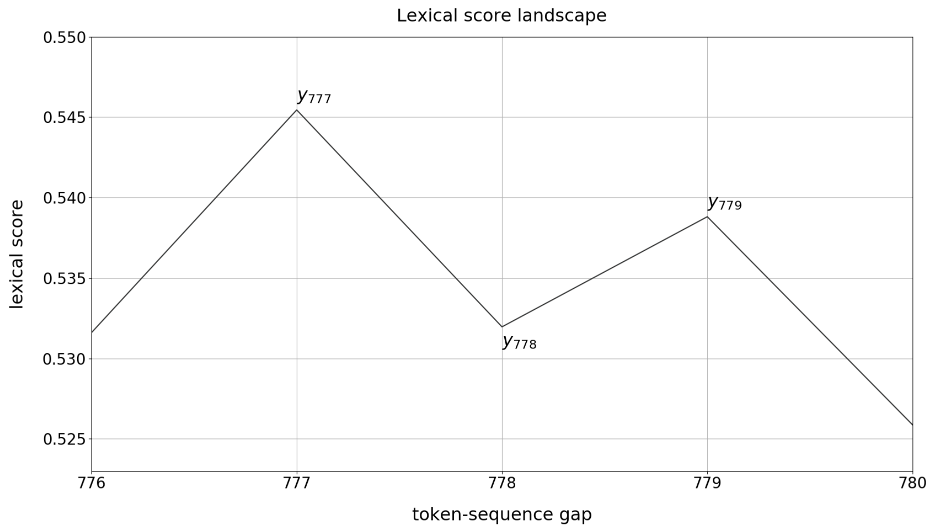


Figure 3.3.: The landscape of lexical scores for the novel ”The Maze Runner” by J. Dashner (k=13, w=49), the gaps between token-sequences 776 to 780 are shown. y_{778} represents a lexical score y at a token-sequence gap 778 and is called a valley because it has lower value than neighboring token-sequence gaps. Lexical scores y_{777} and y_{779} are called peaks.

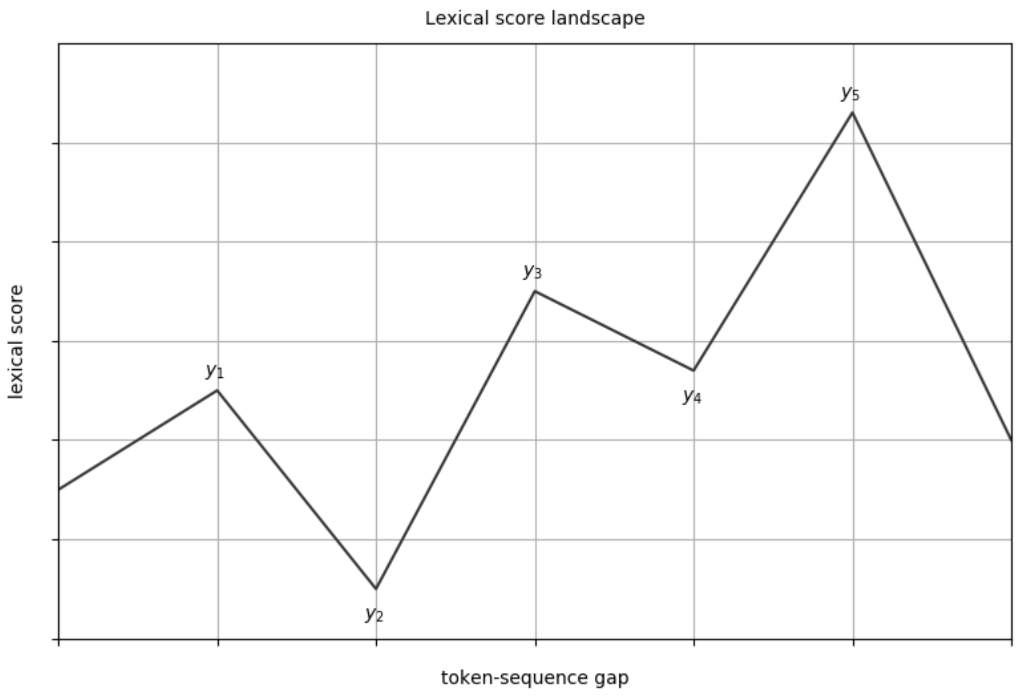


Figure 3.4.: The peak at y_3 is detected as a closest peak to the right of the valley at y_2. However it would be beneficial to mark y_5 as such because the lexical similarity value decreases only slightly in y_4. To avoid such behavior smoothing is introduced.

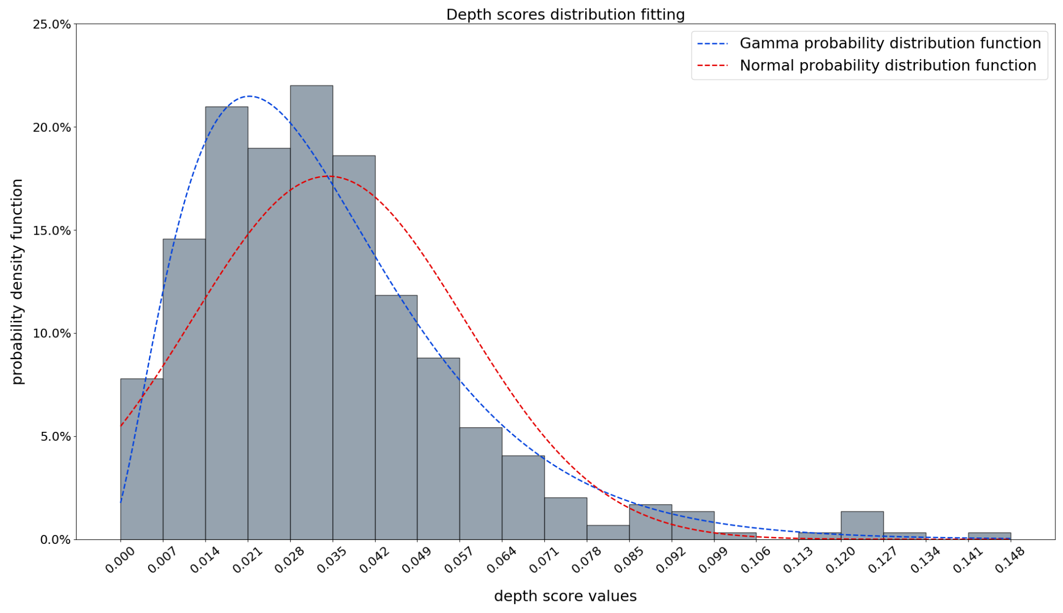


Figure 3.5.: An example of depth score value distribution for the novel ”The Giver” by L. Lowry with block size 12 and token-sequence size 107. The area (integral) under the histogram sums up to 100%. This example illustrates that an assumption made by original TextTiling paper about normal distribution of depth scores values does not hold for the dataset used in this work.

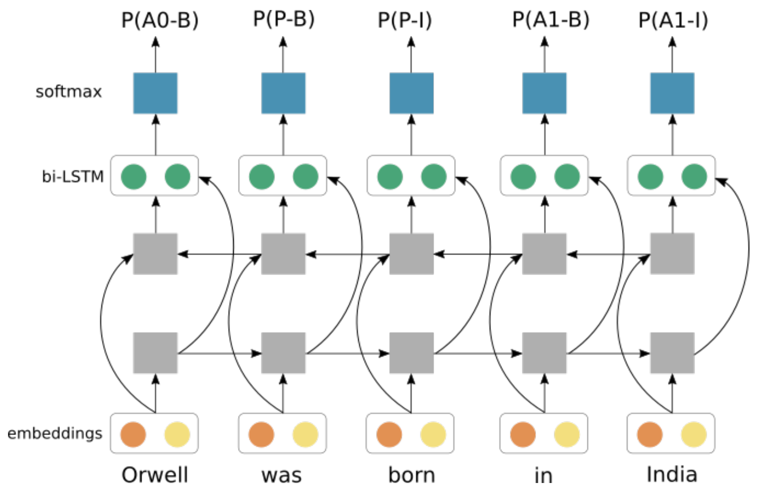


Figure 3.6.: Architecture of a bi-LSTM RNN used in RnnOIE, adapted from Stanovsky et al. [54]. Neural network input contains both current word's features (orange circles) and predicate's syntactic head features (yellow circles). These features consist out of word embedding and it's part of speech embedding. RnnOIE uses a bi-LSTM RNN with the softmax output layer to produce independent probability distributions over all various possible BIO tags.

4. Evaluation

4.1. Measures

For results evaluation two common metrics for text segmentation are used: $\mathrm{P_k}$ [8] and WD (WindowDiff) [49].

4.1.1. WindowDiff Measure

Evaluated algorithm provides a list of hypothetical boundaries which is compared to the list of reference segmentation boundaries. To calculate WindowDiff Measure a window of size k is moved over a document. The moving window size k is set to the half of the average segment size of the ground truth. In this work a boundary is signified by a token at the beginning of a new subtopic segment. Given an interval with k tokens $[i, i+k]$ the number of reference boundaries in this interval $b(ref_i, ref_{i+k})$ is compared to the number of hypothetical boundaries $b(hyp_i, hyp_{i+k})$, see Figure 4.1 for a visual example.

In case $b(ref_i, ref_{i+k})$ is not equal to $b(hyp_i, hyp_{i+k})$ a penalty is added to the algorithm's score. The size of the penalty is $|b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k})| > 0$. The score is normalized by the number of performed measurements therefore the possible values of WD measure are in $[0, 1]$ with zero being the best achieved value. If a boundary is off by a number of words less than k it is called a ***near miss*** and is still considered a correct boundary identification.

4.1.2. $\mathrm{P_k}$ Measure

$\mathrm{P_k}$ measure is a probability that k units (in case of this work k tokens) are incorrectly identified as being either in a same segment or different segments. See Figure 4.1 for

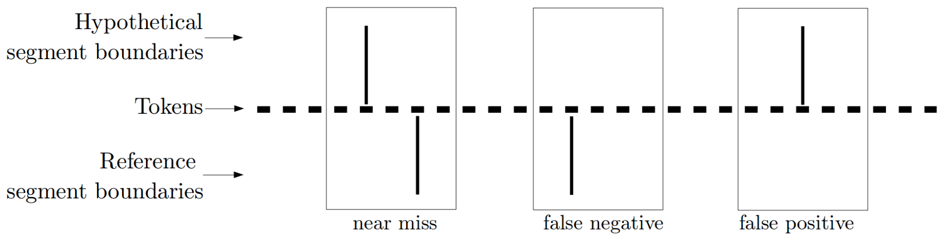


Figure 4.1.: An example of differences between hypothesized segmentation and reference segmentation given a window size k of five tokens. If a boundary is off by a few tokens it is called a near miss and is still considered a correct boundary identification. False negative is a boundary which is present in the reference segmentation, but was not detected by the algorithm. False positive is a boundary which was found by the algorithm despite its absence in the reference segmentation. False positives and false negatives are penalized by both WindowDiff and $\mathrm{P_k}$.

a visual example. The moving window size k is set to the half of the average segment size of the ground truth. Therefore an interval $[i, i+k]$ consisting of k tokens is observed with an assumption that any of the tokens may be a segment beginning. Inside of this interval the number of tokens $b(hyp_i, hyp_{i+k})$ which are considered by the algorithm under evaluation to start a new segment is compared to the analogous number $b(ref_i, ref_{i+k})$ from the reference segmentation. If $b(hyp_i, hyp_{i+k})$ is not equal to $b(ref_i, ref_{i+k})$ a penalty is added to the algorithm's score. In this case a detected subtopic boundary is either a false positive or a false negative. A near miss is considered a correct boundary identification. After the comparison is done the moving window is moved further by one token. At the end of execution the obtained score is normalized by the number of performed measurements. An algorithm which has managed to detect all boundaries correctly achieves a $\mathrm{P_k}$ score zero.

4.2. Results

For TextTiling algorithm the values of two parameters, token-sequence size w and block size k, should be defined prior to the run, see Subsection 3.3.1 and Subsection 3.3.2 for more information.

The first parameter w is an estimate of sentence length in tokens. Initially an assumption was made that its possible values lie in the interval between minimum sentence length min_sl and maximum sentence length max_sl for the evaluated file. TextTiling parameter k is an approximation of the number of sentences in a paragraph. Analogous to w the value of k was expected to lie in the interval between the minimum paragraph length min_pl and the maximum paragraph length max_pl.

However, the minimum length of a sentence in all files of the dataset is one. The minimum length of the paragraphs (novels) or scenes (subtitles) is equal or less than eight. Therefore using these minimum values as lower parameter value bounds could lead to the algorithm comparing unnecessary short vectors. In case of such comparison a low lexical score will be assigned frequently which would lead to a faulty subtopic boundary detection. To avoid such behavior the lower bound of the value interval is set to the median sentence length value $median\_sl$ and median paragraph length $median\_pl$.

$$\forall w \in [median\_sl, max\_sl], length\ in\ tokens \tag{4.1}$$

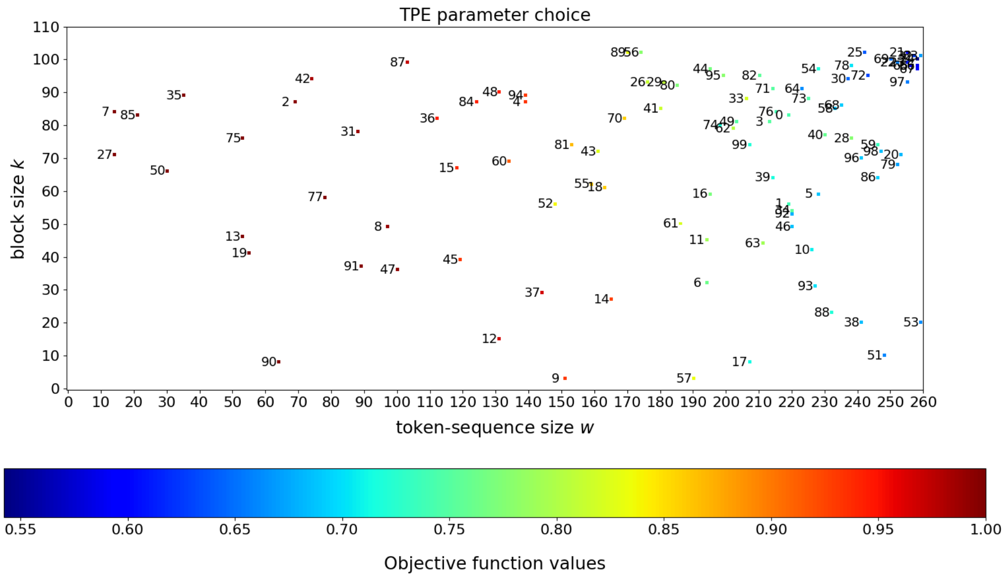
$$\forall k \in [median\_pl, max\_pl], length\ in\ sentences \tag{4.2}$$

To choose the parameter values, first, the objective function which should be minimized by the parameter search algorithms is defined. The objective function takes the token-sequence size and block size as input parameters. The function itself includes the execution of original TextTiling algorithm with input parameters on a given input file. After the TextTiling is performed the WindowDiff measure is applied to the TextTiling results. This way given the input parameters block size and token-sequence size the objective function returns the WindowDiff measure value.

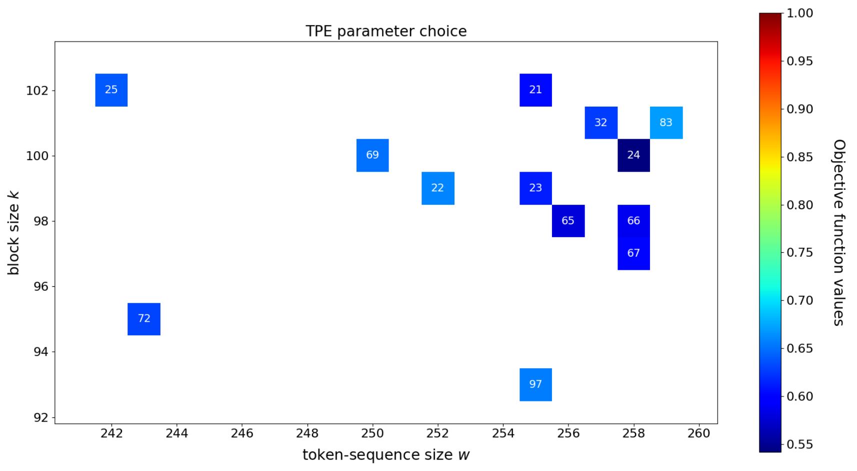
Second, two different parameter optimization algorithms were applied to the novel "Ninety-Eighty Four" as well as to the corresponding subtitles. The random search parameter optimization [9] was performed for 100 iterations. Then 100 iterations of Bayesian Optimization using the Tree-structured Parzen Estimator (TPE) algorithm [10] were performed on the same dataset items and the results were compared to each other. TPE algorithm delivered more reliable results in less time, therefore in this work the choice of a parameter pair (k_i,w_i) for an evaluation run i from the corresponding intervals is done with the help of TPE. The TPE algorithm from the module `hyperopt` is used.

Third, for each item in the dataset the value of objective function was calculated for 100 parameter pairs, $i \in [0, 99]$, chosen by the TPE algorithm and stored in a CSV file. Figure 4.2 shows the parameter search process on an example of the novel ”Brave New World” by A. Huxley.

Ten parameter pairs resulting in lowest WindowDiff values were identified for further use. TextTiling with word embeddings as well as TextTiling with OIE are executed given these parameter pairs as input. Here not only the WindowDiff measure, but also P_k measure is calculated. Tables 4.1 and 4.2 provide an overview of the three parameter pairs which produce the lowest WindowDiff and P_k measure values for each file.



(a)



(b)

Figure 4.2.: Parameter search order for the novel "Brave New World" with $k_i \in [2, 102]$ and $w_i \in [9, 259]$. A parameter pair (k_i,w_i) is chosen by TPE algorithm. Numbers next to the points correspond to the evaluation run number i, $i \in [0, 99]$. The color of the point represents the value of the objective function given (k_i,w_i) as input. The algorithm gradually aims at the areas where the probability of the minimal value of the objective function is the highest. The top right corner area of the Figure (a) where lowest values of objective function are located is shown in Figure (b).

	Film title	TextTiling	TextTiling with word embeddings	TextTiling with OIE	TextTiling	TextTiling with word embeddings	TextTiling with OIE	TextTiling	TextTiling with word embeddings	TextTiling with OIE
1	Nineteen Eighty-Four (1956)	k=39, w=22			k=34, w=21			k=29, w=22		
		WD=0.412 P_k=0.387	WD=0.423 P_k=0.401	**WD=0.389** **P_k=0.375**	WD=0.403 P_k=0.387	WD=0.418 P_k=0.407	WD=0.405 P_k=0.386	WD=0.409 P_k=0.388	WD=0.416 P_k=0.397	WD=0.413 P_k=0.394
2	Nineteen Eighty-Four (1984)	k=29, w=22			k=34, w=21			k=14, w=21		
		WD=0.41 P_k=0.393	**WD=0.395** **P_k=0.376**	WD=0.405 P_k=0.389	WD=0.401 P_k=0.381	WD=0.431 P_k=0.411	WD=0.405 P_k=0.386	WD=0.443 P_k=0.426	WD=0.408 P_k=0.389	WD=0.429 P_k=0.415
3	Brave New World	k=32, w=41			k=36, w=41			k=25, w=38		
		WD=0.395 P_k=0.372	WD=0.393 P_k=0.37	**WD=0.381** **P_k=0.357**	WD=0.394 P_k=0.371	WD=0.382 P_k=0.359	WD=0.409 P_k=0.39	WD=0.383 P_k=0.367	WD=0.404 P_k=0.39	WD=0.395 P_k=0.375
4	We	k=50, w=48			k=43, w=51			k=45, w=48		
		WD=0.393 P_k=0.385	WD=0.413 P_k=0.408	WD=0.445 P_k=0.445	WD=0.394 **P_k=0.383**	WD=0.412 P_k=0.409	WD=0.444 P_k=0.438	WD=0.396 P_k=0.39	WD=0.423 P_k=0.422	WD=0.433 P_k=0.429
5	The Handmaid's Tale	k=56, w=44			k=42, w=41			k=55, w=40		
		WD=0.349 P_k=0.334	WD=0.367 P_k=0.352	WD=0.369 P_k=0.36	WD=0.356 P_k=0.343	WD=0.35 **P_k=0.329**	WD=0.362 P_k=0.357	WD=0.35 P_k=0.34	WD=0.386 P_k=0.379	WD=0.38 P_k=0.37
6	Blade Runner	k=18, w=34			k=22, w=23			k=28, w=34		
		WD=0.336 **P_k=0.325**	WD=0.36 P_k=0.351	WD=0.354 P_k=0.343	WD=0.35 P_k=0.331	WD=0.343 P_k=0.326	WD=0.363 P_k=0.348	WD=0.347 P_k=0.344	WD=0.349 P_k=0.339	WD=0.352 P_k=0.339
7	The Hunger Games	k=46, w=20			k=27, w=24			k=48, w=19		
		WD=0.374 **P_k=0.359**	WD=0.411 P_k=0.4	WD=0.411 P_k=0.394	WD=0.383 P_k=0.362	WD=0.409 P_k=0.4	WD=0.406 P_k=0.394	WD=0.386 P_k=0.366	WD=0.42 P_k=0.411	WD=0.413 P_k=0.396
8	The Hunger Games: Catching Fire	k=57, w=40			k=51, w=40			k=61, w=40		
		WD=0.32 **P_k=0.305**	WD=0.334 P_k=0.324	WD=0.342 P_k=0.327	WD=0.327 P_k=0.312	WD=0.337 P_k=0.329	WD=0.329 P_k=0.315	WD=0.328 P_k=0.317	WD=0.35 P_k=0.343	WD=0.338 P_k=0.332
9	The Hunger Games: Mockingjay - Part 1	k=13, w=97			k=8, w=98			k=12, w=110		
		WD=0.354 P_k=0.35	**WD=0.3353** P_k=0.331	WD=0.349 P_k=0.347	WD=0.351 P_k=0.342	WD=0.3354 **P_k=0.327**	WD=0.36 P_k=0.355	WD=0.354 P_k=0.351	WD=0.343 P_k=0.338	WD=0.354 P_k=0.346
10	The Hunger Games: Mockingjay - Part 2	k=21, w=97			k=20, w=97			k=13, w=108		
		WD=0.326 P_k=0.322	**WD=0.315** P_k=0.312	WD=0.329 P_k=0.326	WD=0.33 P_k=0.327	WD=0.316 **P_k=0.311**	WD=0.325 P_k=0.321	WD=0.332 P_k=0.33	WD=0.334 P_k=0.331	WD=0.317 P_k=0.312
11	The Giver	k=82, w=28			k=78, w=28			k=70, w=28		
		WD=0.427 P_k=0.388	**WD=0.408** **P_k=0.372**	WD=0.442 P_k=0.414	WD=0.409 P_k=0.376	WD=0.43 P_k=0.399	WD=0.432 P_k=0.414	WD=0.414 P_k=0.376	WD=0.419 P_k=0.387	WD=0.485 P_k=0.463
12	The Maze Runner	k=43, w=27			k=5, w=28			k=44, w=27		
		WD=0.357 **P_k=0.338**	WD=0.373 P_k=0.36	WD=0.38 P_k=0.367	WD=0.359 P_k=0.34	WD=0.388 P_k=0.377	WD=0.402 P_k=0.392	WD=0.36 P_k=0.347	WD=0.375 P_k=0.364	WD=0.402 P_k=0.394
13	Ready Player One	k=95, w=34			k=66, w=39			k=73, w=39		
		WD=0.365 P_k=0.346	**WD=0.3547** P_k=0.339	WD=0.378 P_k=0.369	WD=0.366 P_k=0.345	WD=0.3554 **P_k=0.334**	WD=0.365 P_k=0.349	WD=0.369 P_k=0.351	WD=0.356 P_k=0.345	WD=0.373 P_k=0.362

Table 4.1.: An overview of three parameter pairs which result in the lowest WindowDiff or P_k values for each file in the subtitle dataset part. The same parameter pair (block size k and token-sequence size w) is used to execute original TextTiling algorithm, TextTiling with word embeddings as well as TextTiling with OIE. The lowest achieved measure value is marked in bold.

	Novel title	TextTiling	TextTiling with word embeddings	TextTiling with OIE	TextTiling	TextTiling with word embeddings	TextTiling with OIE	TextTiling	TextTiling with word embeddings	TextTiling with OIE
1	Nineteen Eighty-Four: A Novel	k=24, w=136			k=31, w=135			k=14, w=131		
		WD=0.961 P_k=0.531	WD=0.917 P_k=0.514	**WD=0.854** P_k=0.515	WD=0.97 P_k=0.528	WD=0.874 P_k=0.529	WD=0.882 P_k=0.564	WD=0.97 P_k=0.528	WD=0.891 P_k=0.519	WD=0.909 **P_k=0.509**
2	Brave New World	k=100, w=258			k=99, w=255			k=98, w=256		
		WD=0.542 **P_k=0.424**	WD=0.623 P_k=0.503	WD=0.685 P_k=0.541	WD=0.613 P_k=0.507	WD=0.544 P_k=0.432	WD=0.671 P_k=0.522	WD=0.579 P_k=0.456	WD=0.672 P_k=0.5	WD=0.668 P_k=0.532
3	We	k=22, w=88			k=31, w=88			k=29, w=81		
		WD=0.783 P_k=0.521	**WD=0.703** P_k=0.499	WD=0.739 P_k=0.5	WD=0.756 P_k=0.508	WD=0.738 P_k=0.519	WD=0.724 P_k=0.51	WD=0.789 P_k=0.515	WD=0.817 P_k=0.522	WD=0.76 **P_k=0.468**
4	The Handmaid's Tale	k=14, w=112			k=11, w=109			k=16, w=112		
		WD=0.788 P_k=0.525	WD=0.721 P_k=0.485	**WD=0.709** P_k=0.518	WD=0.77 P_k=0.497	WD=0.747 P_k=0.514	WD=0.711 **P_k=0.475**	WD=0.768 P_k=0.503	WD=0.716 P_k=0.49	WD=0.725 P_k=0.513
5	Do Androids Dream of Electric Sheep?	k=20, w=70			k=10, w=71			k=17, w=68		
		WD=0.99 P_k=0.523	**WD=0.953** P_k=0.508	WD=0.983 P_k=0.517	WD=0.988 P_k=0.523	WD=0.959 **P_k=0.502**	WD=0.972 P_k=0.507	WD=0.996 P_k=0.523	WD=0.959 P_k=0.515	WD=0.99 P_k=0.515
6	The Hunger Games	k=39, w=54			k=40, w=52			k=5, w=48		
		WD=0.996 P_k=0.5094	**WD=0.981** P_k=0.512	WD=0.9814 P_k=0.515	WD=0.995 P_k=0.51	WD=0.989 **P_k=0.509**	WD=0.986 P_k=0.517	WD=1.0 P_k=0.5094	WD=1.0 P_k=0.5094	WD=1.0 P_k=0.5094
7	Catching Fire	k=11, w=44			k=3, w=51			k=8, w=56		
		WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509	WD=1.0 P_k=0.509
8	Mockingjay	k=19, w=46			k=29, w=36			k=12, w=43		
		WD=1.0 P_k=0.524	**WD=0.989** P_k=0.528	WD=1.0 P_k=0.524	WD=1.0 P_k=0.524	WD=0.994 P_k=0.524	WD=0.993 P_k=0.527	WD=1.0 P_k=0.524	WD=1.0 P_k=0.524	WD=1.0 P_k=0.524
9	The Giver	k=12, w=107			k=11, w=106			k=14, w=101		
		WD=0.723 P_k=0.529	**WD=0.69** **P_k=0.468**	WD=0.739 P_k=0.481	WD=0.719 P_k=0.52	WD=0.711 P_k=0.542	WD=0.728 P_k=0.534	WD=0.748 P_k=0.481	WD=0.764 P_k=0.518	WD=0.772 P_k=0.521
10	The Maze Runner	k=13, w=49			k=9, w=52			k=7, w=50		
		WD=0.989 P_k=0.511	WD=0.947 P_k=0.503	**WD=0.934** **P_k=0.498**	WD=0.987 P_k=0.511	WD=0.956 P_k=0.513	WD=0.962 P_k=0.509	WD=0.988 P_k=0.511	WD=0.964 P_k=0.516	WD=0.969 P_k=0.51
11	Ready Player One	k=8, w=71			k=8, w=60			k=8, w=58		
		WD=0.998 P_k=0.51	WD=0.992 P_k=0.513	**WD=0.981** P_k=0.516	WD=0.998 P_k=0.511	WD=0.99 P_k=0.51	WD=0.984 P_k=0.509	WD=0.997 P_k=0.51	WD=0.99 P_k=0.51	WD=0.986 **P_k=0.508**

Table 4.2.: An overview of three parameter pairs which result in the lowest WindowDiff or P_k values for each file in the novel dataset part. The same parameter pair (block size k and token-sequence size w) is used to execute original TextTiling algorithm, TextTiling with word embeddings as well as TextTiling with OIE. In case a unique lowest achieved measure value is obtained it is marked in bold.

4.3. Discussion

As can be seen in Tables 4.1 and 4.2 it is not possible to generalize a TextTiling parameter space which would satisfy all input files equally well. The parameters yielding the best results may differ significantly and should be chosen based on the data they are applied to.

The value of P_k metric is lower than the one of WindowDiff for the same parameters. The reason for this is the condition which determines if a penalty is given by the measures. If there is at least one reference boundary in the window and one or more boundary in hypothetical boundaries for the same window, P_k does not penalize and as a result does not take the number of detected false positives into account. Hence WindowDiff penalizes the false positives more than P_k which results in an overall lower score and higher measure value.

The leftmost three columns of the Tables 4.1 and 4.2 represent a parameter pair using which the top values for the WindowDiff measure were achieved. Using this parameter pair for 6 out of 13 subtitles and for 9 out of 11 novels replacing words by embedding vectors in TextTiling algorithm decreases the WindowDiff measure which signifies the increase in the algorithm's performance. The reason for this may be that word embedding model includes the context of a text for which it was generated. The original TextTiling algorithm (TT) studies a number of word tokens which two consequent blocks of text have in common. The more similar is the vocabulary of two blocks, the higher their lexical similarity. The meaning of these word tokens and their significance in the rest of the text is not taken into consideration. Word embeddings offer a similar representation for words used in a similar way [1]. Applying TextTiling with word embeddings (TWE) to the input and therefore replacing the word tokens with word embeddings of these tokens extends the TT representation introducing more context information. Word embeddings have an ability to capture related words [33], which also may improve the linear text segmentation results for fictional texts with diverse vocabulary.

The substitution of word tokens with word embeddings changes the landscape of lexical scores and depth scores by altering the location of peaks and valleys. Word embeddings are real-valued vectors single elements of which have the order of magnitude less than -1. The high similarity between the vectors frequently results in values of lexical scores close, but not equal, to one. As a consequence the magnitude of depth scores is close, but not equal, to zero. To compare the landscape of depth scores of the TT approach to the TWE approach, the TWE depth scores and cut-off threshold are scaled up. The scaling ratio is calculated by dividing the cut-off threshold of TT by the cut-off threshold of TWE. This way both methods share the same value of the TT cut-off threshold and the differences in the depth score values are easier to compare visually. As can be seen in the Figure 4.3 on an example from the novel ”The Maze Runner” by J. Dashner the landscape of the depth scores for TWE is similar to the one of TT, however the importance of token-sequence gaps is magnified or reduced depending on which approach is used.

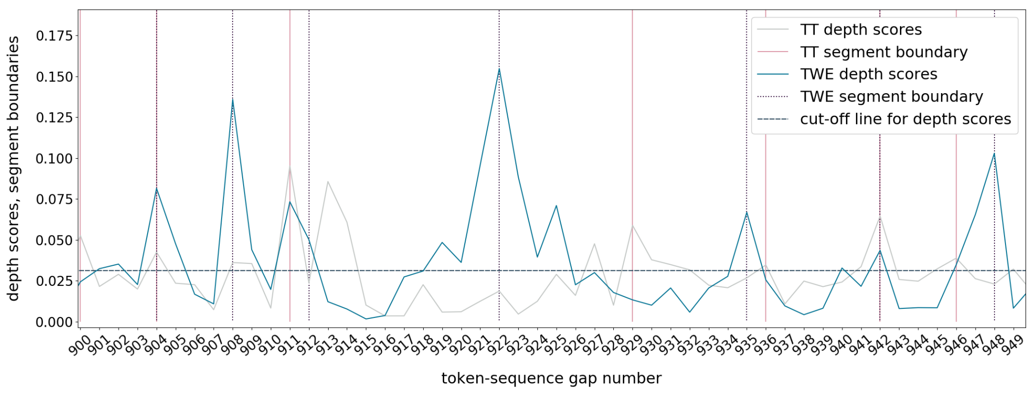


Figure 4.3.: The landscape of depth scores for the novel ”The Maze Runner” by J. Dashner (k=13, w=49), the gaps between token-sequences 900 to 949 are shown. TWE depth scores were scaled up to allow better visual comparison. Token-sequence gaps 904 and 942 are identified as a subtopic boundary by both TT and TWE. Token-sequence gaps 908, 922, 948 are considered subtopic boundaries by TWE, but not by TT and other way around in case of the gaps 929 and 946. For the token-sequence gaps 911 and 912 as well as 935 and 936 the algorithms disagree on the boundary by one token-sequence gap.

The disagreement between the values of P_k and WindowDiff measures for subtitles is low compared to the novels. This difference signifies that a number of false positives detected by the algorithms for subtitle files is also lower. This may be due to the nature of ground truth for subtitles. Finding an unambiguous subtopic boundary is a challenging task even for linguistic experts [26]. In this work a more generic approach for creation of ground truth is used as described in Subsections 3.2.1 and 3.2.2. Automatically generated subtopic breaks for subtitle files are more frequent compared to the novels. The subtopic breaks in the ground truth are located close to each other in many cases. This may correspond better to how scene transitions are distributed in fictional texts than separation into chapters.

Figure 4.4a shows the reference boundaries as well as boundaries identified by the algorithms presented in this work on an example of the subtitles to the film ”Ninety Eighty-Four” produced in 1956. For a comparison an example of reference boundaries of novels can be seen in Figure 4.4b.

As mentioned above, the separation into chapters as the smallest subtopic segments may be too coarse for novels. The choice of chapter beginnings as segment start markers expects that one chapter represents one subtopic segment and there are no subject switches inside of a chapter, yet this is not always the case. This can be illustrated on an example of the novel ”Catching Fire” by S. Collins. The measure values for the novel are large independent of the parameters or chosen method. The reason for this behavior may be that the novel mostly consists of dialogue with rather short descriptions from a narrator's perspective. Each dialogue line is treated by the algorithm as a new paragraph, allowing it to place more subtopic shift markers as it would do in case of a descriptive text with long paragraphs. In a dialogue more than one party is expressing their opinion with their own specific vocabulary, therefore the vocabulary changes frequently. Nevertheless it is expected that a big vocabulary switch happens only at the beginning of a chapter resulting in a large number of false positives as can be seen in the Figure 4.4b.

In reference to novels the block size parameter space is linked to the paragraph length and therefore independent from the chapter length, wherein the chapter boundaries are representing the ground truth. The subtitles do not contain chapters or paragraphs, therefore the scene boundaries as defined in the Subsection 3.2.2 play the role of both paragraph breaks and subtopic breaks (ground truth). Limiting the block size to be the maximum size of paragraphs (novels) or scenes (subtitles) allows the length of a single block to exceed the median length of the text segment in a reference significantly.

In case of the novel ”Brave New World” by A. Huxley both block size and token-sequence size are large compared to the parameters of other novels in Table 4.2. The size of a single block is 25 800 word tokens or less taking into consideration that the removal of stop words decreases the block size, wherein the median length of a chapter is 3 682 word tokens (stop words included). Hence the size of a block may exceed the text reference by 7.0 times. Due to the large size of the token-sequence the word embedding vector corresponding to this sequence looses accuracy of its context meaning which may cause TT to produce better results than TWE.

For subtitles a similar picture can be observed: the lower the ratio between maximum block size and median scene size, the better TWE performance. For ”Nineteen Eighty-Four”(1984) the ratio is 29.00, for ”Nineteen Eighty-Four”(1956) the ratio is 30.11, for ”Mockingjay”(part 1) the ratio is 36.03, for ”Brave New World” the ratio is 37.49, for ”The Giver” the ratio is 38.92. The ratios for the rest of the subtitles range from 48.42 (”Blade Runner”) to 170.00 (”Ready Player One”). Nevertheless TWE performs better than TT despite the high ratio between maximum block size and median scene length for two subtitle files, ”Mockingjay” (part 2) with ratio 97.00 and ”Ready Player One” with ratio 170.00.

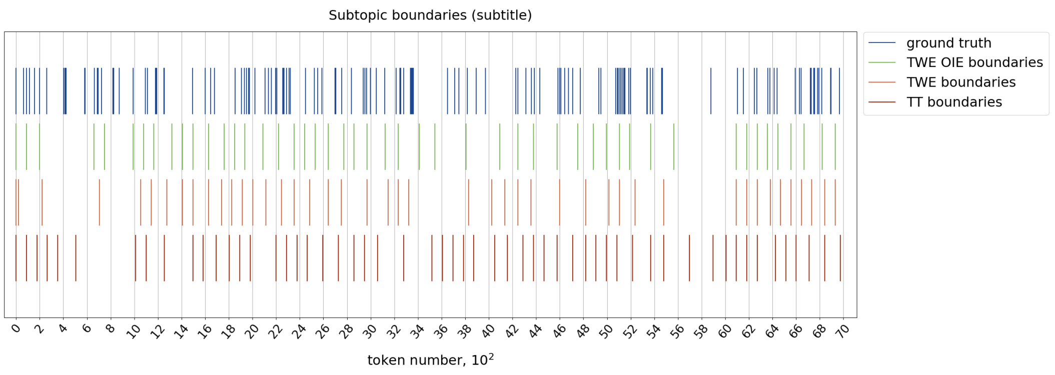
Despite much larger ratios the maximum size of the block for subtitles is notably lower than the one of novels. The largest value of maximum block size for subtitles is 3 230 for ”Ready Player One”, whereas for novels aforementioned ”Brave New World” has the largest value of 25 800 word tokens. A large block containing a large number of token-sequences may lead to a consequence that a vector corresponding to such block in a step of lexical score calculation carries little unique information about the block.

Observing the parameter pair with the best achieved WindowDiff measure values in Tables 4.1 and 4.2, for 4 out of 13 subtitles and for 7 out of 11 novels applying TextTiling with word embeddings and OIE weights (TWE OIE) increases the performance in comparison to TT. Figure 4.5 illustrates the changes in the depth score landscape depending on the method. An example shown is taken from the subtitles to the film ”The Maze Runner”. To allow better visual comparison the depth score values of TWE and TWE OIE were scaled up as described above.

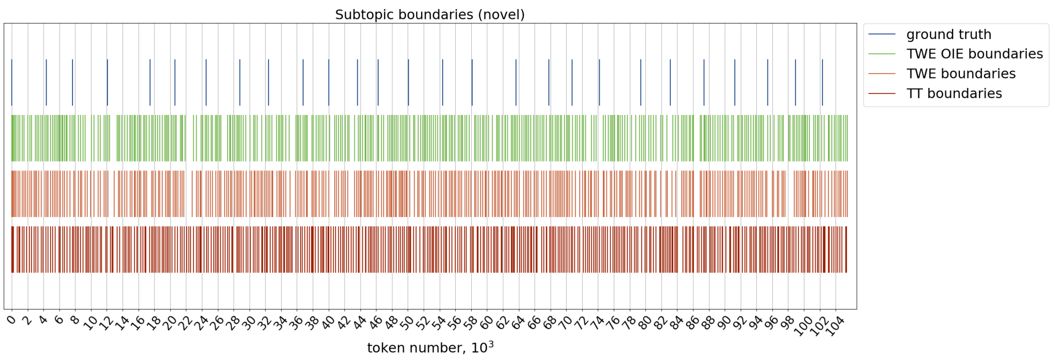
A question arises why does the application of TWE OIE produce better results than TWE in some cases as well as why does it appear to be a more effective method for novels than subtitles. A possible answer is the writing style of the input text. OIE tool used for TWE OIE, RnnOIE, extracts propositions from an unstructured text with one or more predicates as described in Section 3.4. However, predicates consisting of nouns as well as informal speech pose a challenge to RnnOIE [54]. An example of sentence posing a challenge is ”Because of my job” found in a novel ”Do Androids Dream of Electric Sheep?”. This short sentence appears to be a part of a spoken or written conversation between characters of the novel. In such case the grammatically correct construction of a sentence is neglected and the mind of a conversational partner is supposed to fill in the lack of linguistic structure. Given such sentences as input RnnOIE does not produce any tuples, the sentence is considered irrelevant by the pipeline, which in turn leads to zero weights for all word tokens of such sentence. This poses a problem when a large number of short informal sentences in the data is encountered which is the case for all the subtitles and some of the novels.

In subtitles the percentage of sentences under 5 words in an input file exceeds 35% percent for each file ranging from film ”Brave New World” with 37.95% to ”The Maze Runner” with 67.29%. In novels the percentage of such sentences is lower ranging from 15.47% for ”Nineteen Eighty-Four: A Novel” to 28.36% for ”The Giver”. Other novels with number of short sentences over 25% are ”Brave New World” (28.09%), ”Do Androids Dream of Electric Sheep?” (26.93%), ”We” (26.76%) and ”Mockingjay” (26.74%). For all these files the WindowDiff measure for TWE OIE is higher than for TWE.

During the analysis of the results it became apparent that some of the stop words have not been completely filtered out. Custom RnnOIE tokenizer used in this work separates contractions used with personal pronouns into two tokens, a contraction token keeping an apostrophe. For example, ”I’m” is separated into tokens ”I” and ”’m”. The same behavior is observed in case of possessive ”’s”, for instance, ”Dave’s” is separated into ”Dave” and ”’s”. Contractions are included in the `nltk.corpus` stop words list, however they do not include an apostrophe which means that for example ”s” is in the stop words list, but ”’s” is not. Hence the stop words list should be extended.



(a)



(b)

Figure 4.4.: An example of ground truth and subtopic boundaries detected by three algorithms. A word token which is positioned directly after the subtopic shift is marked by a vertical line on the y-axis. The colors of the vertical lines correspond to the method used to detect subtopic shifts.
Figure (a) shows detected subtopic boundaries for the subtitles to the film "Ninety Eighty-Four" (1956), k=39, w=22. There is a larger number of subtopic boundaries than in novels. Multiple subtopic breaks in ground truth are located close to each other.
Figure (b) depicts detected subtopic boundaries for the novel "Catching Fire" by S. Collins, k=11, w=44. All methods have difficulty identifying the switch from one chapter to another (ground truth) as a subtopic shift and introduce a large number of false positives.

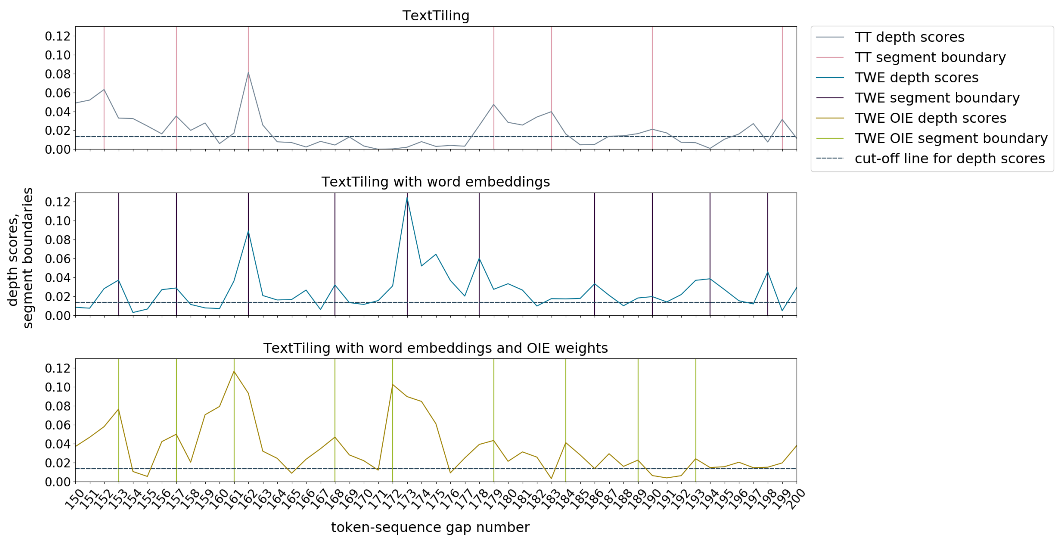


Figure 4.5.: A comparison of the landscape of depth scores and boundaries for three methods: TT, TWE and TWE OIE. Token-sequence gaps 150 to 200 of a subtitle file to a film "The Maze Runner" (k=43, w=27) are shown. TWE and TWE OIE depth scores were scaled up to allow better visual comparison.
All algorithms agree on a subtopic boundary at token-sequence gap 157. TT depth score landscape from token-sequences 162 to 179 contains a prolonged valley, however TWE and TWE OIE demonstrate multiple peaks. Both TWE and TWE OIE agree on the subtopic boundary at the token-sequence gap 168 and another subtopic boundary is detected at 173 by TWE and 172 by TWE OIE. TT identifies the subtopic boundary at token-sequence gap 183, in TWE landscape this peak is suppressed, however it appears again in TWE OIE landscape at a token-sequence gap 184. Two peaks in TWE after the token-sequence gap 193 are almost completely suppressed in TWE OIE.

5. Conclusions

The first research question this thesis aimed to answer was whether a modification of a linear text segmentation method by adding knowledge generated by Open Information Extraction influences the performance of this method. The text segmentation pipeline introduced in this work is referred to as TWE OIE. An intermediate step of TWE OIE, referred to as TWE, is a replacement of word tokens in a linear text segmentation algorithm called TextTiling with corresponding word embeddings. Given the pipeline run with the lowest achieved WindowDiff and P_k values the substitution resulted in lower WindowDiff and P_k values for 6 out of 13 subtitles and 9 out of 11 novels, which signified increase in performance. Application of a complete pipeline by using extractions produced by an OIE system RnnOIE to generate a weighting scheme for obtained word embeddings led to decrease of both WindowDiff and P_k values for 4 out of 13 subtitles and 7 out of 11 novels. Therefore it is concluded that the usage of word embeddings in combination with information obtained from RnnOIE system has a potential to improve the performance of TextTiling text segmentation algorithm. It should be mentioned that the dataset created for this work used automatically generated topic boundaries as ground truth. Other than that due to its fictional nature the dataset did not contain obvious subtopic boundaries. Thus performing evaluation on a corpus with more apparent subtopic switches may provide better results. The major challenges of introduced pipeline were related to the TextTiling parameter optimization, handling of long texts with considerably variable subtopic length and extracting propositions from grammatically incomplete sentences.

The second research question was whether there is a difference in performance of the introduced pipeline for parts of the created corpus (novels and subtitles). Overall the WindowDiff and P_k values for subtitles are significantly lower than measure values for novels independent of the used method, which may be caused by the nature of topic boundaries distribution in the subtitles part of the corpus. WindowDiff and P_k values of the pipeline are lower than the benchmark performance for 7 out of 11 novels and 4 out of 13 subtitle files for the pipeline run with the lowest measure values. Therefore TWE OIE appears to be a more effective method for novels than subtitles. The difference in performance may be attributed to the large number of short informal sentences encountered in the subtitles.

A further contribution of this thesis to the NLP research area is creation of a corpus consisting of 11 fictional novels and 13 subtitle files to the film adaptations of these novels. This type of data holds such unique properties as synonym usage, fictitious terms, creative writing style, variable length of text segments, irregular topic transitions and extensive use of dialogue.

5.1. Future Work

This thesis provides a good starting point for discussion and further research of applying OIE based on neural networks to the linear text segmentation and other NLP tasks. More extensive evaluation of performed work can be obtained by using the described approach on a modified version of current dataset or other datasets. Dataset used in this thesis consisted of texts from twentieth and twenty-first century with fictional narrative form. Hence it is important to investigate the influence of other text writing styles on the results. This can be achieved, for instance, by creating a dataset consisting of expository texts and subtitles to documentaries. Evaluation of results on an artificial dataset with unambiguous subtopic shifts would make it possible to generate baseline WindowDiff and P_k values for TWE and TWE OIE. Such baseline could provide a reference for result evaluation. The ground truth in this work was automatically generated and future work could include the creation of more accurate ground truth by linguistics experts.

On one hand, it is possible to define the parts of used dataset (novels and subtitles) as two independent datasets with separate word embedding models. On the other hand, a word embedding model may be trained on a single text document consisting of all files in the dataset merged into one. It is a question of additional research to investigate how strong is the influence of such separation or merger on the linear text segmentation outcome.

It became apparent that informal writing with short, not fully formulated sentences had an influence on the performance. Therefore a comparison of achieved results to performance of the pipeline on a dataset consisting of texts written in formal language poses an interesting question.

Moreover, modifications and improvements can be made to the used models and parameters. The parameter space used for the TextTiling algorithm parameters was based on the statistical information about the input file. It is possible to further fine-tune or redefine the boundaries of the parameter space, especially given the findings about the influence of the block size on the results described in Section 4.3. Parameter search could be performed for larger number of iterations for every modification of the original linear text segmentation algorithm. In this thesis the same parameters were used to generate a single word-embedding model for each linear text segmentation. The influence of parameter tuning for the Word2Vec model generation on the overall result could be examined in more detail. Further research may prove beneficial to the computation of sentence embeddings as well as token weight calculation based on OIE tuples.

In terms of the results application to other NLP fields, a promising direction for future work could be examining the usage of subtopic segments to perform an alignment of scenes in films to scenes in novels. Apart from looking at the datasets in English, a possible next step could be applying discussed techniques to other natural languages.

Bibliography

[1] C. C. Aggarwal. *Machine learning for text.* Springer, 2018.

[2] E. Amer and A. Nabil. A framework to automate the generation of movies' trailers using only subtitles. In *Proceedings of the 7th International Conference on Software and Information Engineering*, pages 126–130, 2018.

[3] G. Angeli, M. J. J. Premkumar, and C. D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, 2015.

[4] S. Arnold, R. Schneider, P. Cudré-Mauroux, F. A. Gers, and A. Löser. Sector: a neural model for coherent topic segmentation and classification. *Transactions of the Association for Computational Linguistics*, 7:169–184, 2019.

[5] R. Bamler and S. Mandt. Dynamic word embeddings. *arXiv preprint arXiv:1702.08359*, 2017.

[6] M. Banko, M. J. Cafarella, S. Soderl, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 2670–2676, 2007.

[7] B. Basu, K. R. Broad, and C. Hintz. *Contemporary dystopian fiction for young adults: Brave new teenagers.* Routledge, 2013.

[8] D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210, 1999.

[9] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[10] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123, 2013.

[11] G. Bluestone. *Novels into film*. Univ of California Press, 1968.

[12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[13] F. Y. Choi. Advances in domain independent linear text segmentation. *arXiv preprint cs/0003083*, 2000.

[14] L. Cui, F. Wei, and M. Zhou. Neural open information extraction. *arXiv preprint arXiv:1805.04270*, 2018.

[15] N. Dekker, T. Kuhn, and M. van Erp. Evaluating social network extraction for classic and modern fiction literature. *PeerJ Preprints*, 6:e27263v1, 2018.

[16] L. Del Corro and R. Gemulla. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366, 2013.

[17] J. Eisenstein and R. Barzilay. Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 334–343, 2008.

[18] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

[19] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1535–1545, 2011.

[20] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 562–569, 2003.

[21] G. Glavaš, F. Nanni, and S. P. Ponzetto. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Joint Conference on Lexical and Computational ing semantic relatedness graphs*, pages 125–130, 2016.

[22] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[23] P. Gupta, M. Sharma, K. Pitale, and K. Kumar. Problems with automating translation of movie/tv show subtitles. *arXiv preprint arXiv:1909.05362*, 2019.

[24] M. A. K. Halliday and R. Hasan. Cohesion in english, 1976.

[25] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[26] M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Journal of Computational linguistics*, 23(1):33–64, 1997. doi: 10.1021/acs.inorgchem.5b00134.

[27] M. Hesham, B. Hani, N. Fouad, and E. Amer. Smart trailer: Automatic generation of movie trailer using only subtitles. In *2018 First International Workshop on Deep and Representation Learning (IWDRL)*, pages 26–30. IEEE, 2018.

[28] R. Hong, M. Wang, X.-T. Yuan, M. Xu, J. Jiang, S. Yan, and T.-S. Chua. Video accessibility enhancement for hearing-impaired users. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 7S(1), Oct. 2011. doi: 10.1145/2037676.20.

[29] M. S. Jelinek Lewis and D. W. Jackson. Television literacy: Comprehension of program content using closed captions for the deaf. *The Journal of Deaf Studies and Deaf Education*, 6(1):43–53, Jan. 2001. doi: 10.1093/deafed/6.1.43.

[30] H. Jelodar, Y. Wang, C. Yuan, X. Feng, X. Jiang, Y. Li, and L. Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211, 2019.

[31] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[32] R. Kern and M. Granitzer. Efficient linear text segmentation based on information retrieval techniques. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, pages 167–171, 2009.

[33] D. Kiela, F. Hill, and S. Clark. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, 2015.

[34] V. Labatut and X. Bost. Extraction and analysis of fictional character networks: A survey. *ACM Computing Surveys (CSUR)*, 52(5):1–40, 2019.

[35] F. Ladhak, B. Li, Y. Al-Onaizan, and K. McKeown. Exploring content selection in summarization of novel chapters. *arXiv preprint arXiv:2005.01840*, 2020.

[36] S. Lai, K. Liu, S. He, and J. Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016.

[37] J. Li, B. Chiu, S. Shang, and L. Shao. Neural text segmentation and its application to sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[38] P. Lison and R. Meena. Automatic turn segmentation for movie & tv subtitles. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 245–252. IEEE, 2016.

[39] C. MacCabe, K. Murray, and R. Warner. *True to the Spirit: Film Adaptation and the Question of Fidelity.* Oxford University Press, 2011.

[40] P. Markham. Captioned videotapes and second-language listening word recognition. *Foreign Language Annals*, 32(3):321–328, 1999. doi: 10.1111/j.1944-9720.1999.tb01344.x.

[41] E. Matusov, P. Wilken, and Y. Georgakopoulou. Customizing neural machine translation for subtitling. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 82–93, 2019.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, pages 1–12, 2013.

[43] M. Mitri. Story analysis using natural language processing and interactive dashboards. *Journal of Computer Information Systems*, pages 1–11, 2020.

[44] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[45] J. Monaco. *How to read a film: Movies, media, and beyond.* Oxford University Press, 2009.

[46] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh. A survey on open information extraction. *arXiv preprint arXiv:1806.05599*, 2018.

[47] S.-B. Park, K.-J. Oh, and G.-S. Jo. Social network analysis in a movie using character-net. *Multimedia Tools and Applications*, 59(2):601–627, 2012.

[48] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[49] L. Pevzner. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[50] M. Riedl and C. Biemann. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 Student Research Workshop*, pages 37–42, 2012.

[51] F. Serafini and J. Blasingame. The changing face of the novel. *The Reading Teacher*, 66(2):145–148, 2012.

[52] R. Shaw. Automatically segmenting oral history transcripts. *arXiv preprint arXiv:1509.08842*, 2015.

[53] G. Stanovsky and I. Dagan. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305, 2016.

[54] G. Stanovsky, J. Michael, L. Zettlemoyer, and I. Dagan. Supervised open information extraction. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1 (Long Papers)(Section 4):885–895, 2018. doi: 10.18653/v1/N18-1081.

[55] B. Sun, P. Mitra, C. L. Giles, J. Yen, and H. Zha. Topic segmentation with shared topic detection and alignment of multiple documents. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–206, 2007.

[56] P. Svensson and Y. Taoudi. Labeling moods of movies by processing subtitles, 2019.

[57] A. Toral, M. Wieling, and A. Way. Post-editing effort of a novel with statistical and neural machine translation. *Frontiers in Digital Humanities*, 5:9, 2018.

[58] Q. D. Tran and J. E. Jung. Cocharnet: Extracting social networks using character co-occurrence in movies. *J. UCS*, 21(6):796–815, 2015.

[59] G. Tür, D. Hakkani-Tür, A. Stolcke, and E. Shriberg. Integrating prosodic and lexical cues for automatic topic segmentation. *Computational linguistics*, 27 (1):31–57, 2001.

[60] M. Turku. Reality versus fiction: The truth behind a utopian/dystopian novel. *"Hëna e Plotë" Bedër University*, page 54, 2016.

[61] M. Utiyama and H. Isahara. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 499–506, 2001.

[62] E. S. Valentine. *The Protagonist's Response to Power and Language in the Dystopian Novel.* PhD thesis, 1998.

[63] M. Volk, R. Sennrich, C. Hardmeier, and F. Tidström. Machine translation of tv subtitles for large scale production. 2010.

[64] L. Wang, X. Zhang, Z. Tu, A. Way, and Q. Liu. Automatic construction of discourse corpora for dialogue translation. *arXiv preprint arXiv:1605.06770*, 2016.

[65] F. Wu and D. S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 118–127, 2010.

[66] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, and S. Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, 2007.

[67] H. Zamani and W. B. Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514, 2017.

[68] L. Zhang and Q. Zhou. Automatically annotate tv series subtitles for dialogue corpus construction. In *2019 Asia-Pacific Signal and Information Processing*

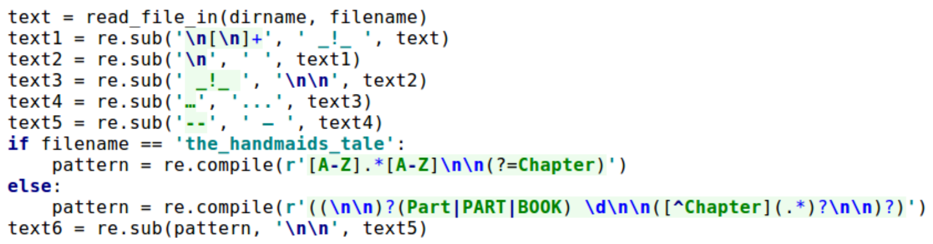
Association Annual Summit and Conference (APSIPA ASC), pages 1029–1035. IEEE, 2019.

[69] W. Zhang, J. C. K. Cheung, and J. Oren. Generating character descriptions for automatic summarization of fiction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7476–7483, 2019.

Appendix

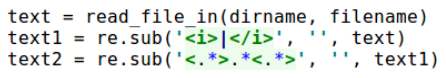
Appendix A.

Regular Expressions



```python
text = read_file_in(dirname, filename)
text1 = re.sub('\n[\n]+', ' _!_ ', text)
text2 = re.sub('\n', ' ', text1)
text3 = re.sub(' _!_ ', '\n\n', text2)
text4 = re.sub('…', '...', text3)
text5 = re.sub('--', ' – ', text4)
if filename == 'the_handmaids_tale':
    pattern = re.compile(r'[A-Z].*[A-Z]\n\n(?=Chapter)')
else:
    pattern = re.compile(r'((\n\n)?(Part|PART|BOOK) \d\n\n([^Chapter](.*)?\n\n)?)')
text6 = re.sub(pattern, '\n\n', text5)
```

Figure A.1.: Regular Expressions used to bring the input files from Novels dataset to a consistent format. Python's `re` module is used. Text of the book "The Handmaid's Tale" by M. Atwood requires special handling.



```python
text = read_file_in(dirname, filename)
text1 = re.sub('<i>|</i>', '', text)
text2 = re.sub('<.*>.*<.*>', '', text1)
```

Figure A.2.: Regular Expressions used to bring the input files from Subtitles dataset to a consistent format. Python's `re` module is used.

Appendix B.

Stop Words and Punctuation

```python
from nltk.corpus import stopwords
stopwords = stopwords.words('english')
stopwords.extend(['us', 'ah', 'um', 'mm', 'mmm', 'oh', 'hm', 'eh', 'uh', 'ha', 'aw', 'ugh', 'huh', 'uhm', 'hmm'])
```

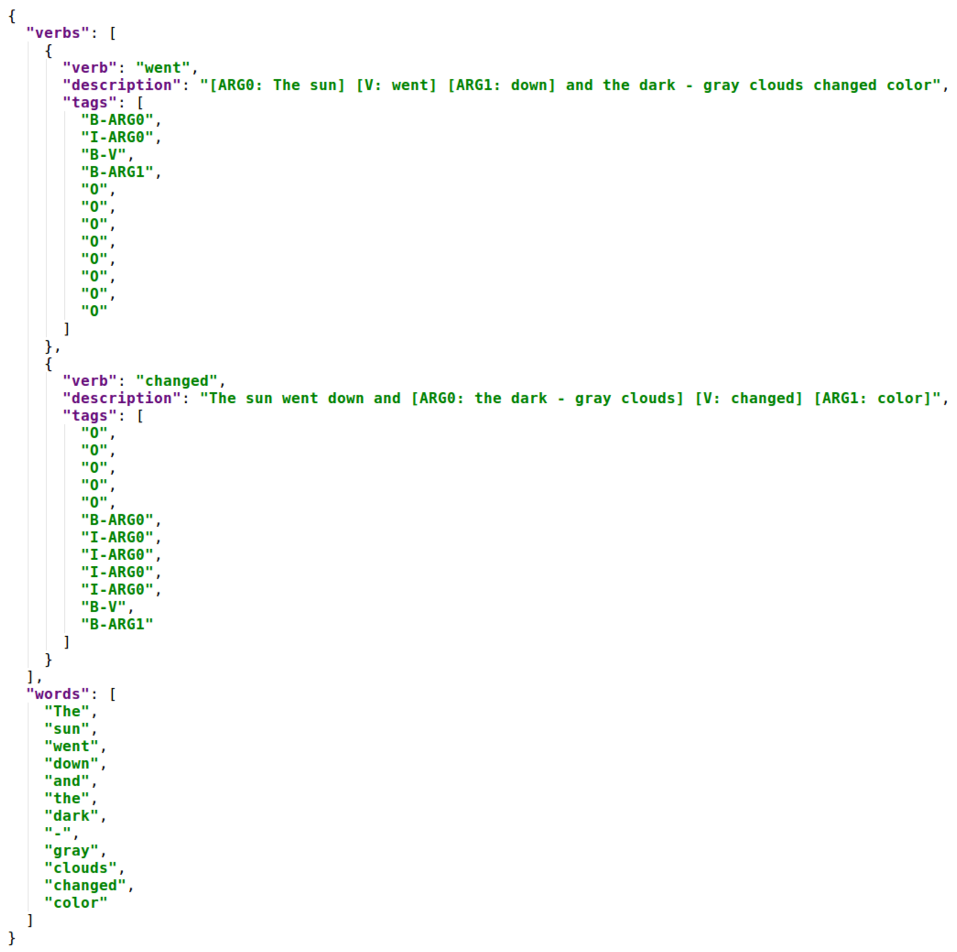
Figure B.1.: List of stop words used in this thesis.

```python
import string
punctuation = string.punctuation + '``' + "''" + '“' + '”' + '...' + '–' + '—' + '―' + '’' + '’' + '--' + \
              '«' + '»' + '°' + '‘' + '... ...' + '§' + ');' + '):' + '©' + '‡' + '†' + '£' + '..' + '>'
```

Figure B.2.: List of punctuation characters used in this thesis.

Appendix C.

RnnOIE Extraction



```
{
  "verbs": [
    {
      "verb": "went",
      "description": "[ARG0: The sun] [V: went] [ARG1: down] and the dark - gray clouds changed color",
      "tags": [
        "B-ARG0",
        "I-ARG0",
        "B-V",
        "B-ARG1",
        "O",
        "O",
        "O",
        "O",
        "O",
        "O",
        "O",
        "O"
      ]
    },
    {
      "verb": "changed",
      "description": "The sun went down and [ARG0: the dark - gray clouds] [V: changed] [ARG1: color]",
      "tags": [
        "O",
        "O",
        "O",
        "O",
        "O",
        "B-ARG0",
        "I-ARG0",
        "I-ARG0",
        "I-ARG0",
        "I-ARG0",
        "B-V",
        "B-ARG1"
      ]
    }
  ],
  "words": [
    "The",
    "sun",
    "went",
    "down",
    "and",
    "the",
    "dark",
    "-",
    "gray",
    "clouds",
    "changed",
    "color"
  ]
}
```

Figure C.1.: An example of RnnOIE extraction in JSON format. Two tuples were extracted, first defined by predicate ”went”, second by predicate ”changed”. The separation of the sentence according to predicate-argument structure can be seen under ”description”. Individual BIO tags for each of the words are listed under ”tags”. Extracted tokens of the sentence are listed under ”words”.