**Institut National des Langues et Civilisations Orientales**

Département Textes, Informatique, Multilinguisme

# Generating Narrative Charts Using Aligned Data from Monolingual Comparable Corpora

## MASTER

### NATURAL LANGUAGE PROCESSING

*Speciality:*

*Multilingual Engineering*

by

**Genevieve BIENVENUE**

*Thesis Director:*

*Cyril Grouin*

*Supervisors:*

*Camille Guinaudeau*
*Hervé Bredin*

2015/2016

# Contents

# List of Figures



# List of Tables

# Abstract

The purpose of this research is to create a visual representation in the form of a narrative chart of the first television season of the *Game of Thrones* series. We do this in order to help clarify the character interactions and timeline in what is known as a very complex story. To accomplish this we perform various text-based alignment and extraction tasks on comparable corpora dealing with *Game of Thrones*. We use the most accurate sets of results from these methods to incrementally improve upon our narrative chart baseline. The final chart includes characters with speaking and nonspeaking roles aligned by scenes in an episode. The chart also includes location information to show where those scenes take place and lists of important words to show at a glance the most important characters and concepts in that episode.

***Key words*** *: Corpus Alignment, Information Extraction, Comparable Corpora, Narrative Chart, Dynamic Time Warping, Game of Thrones*

# Introduction

*Game of Thrones* is an immensely popular fantasy book series, written by author George R. R. Martin. It has been adapted into an award-winning television series by David Benioff and D. B. Weiss. Fans of the books and show have created an extensive online wealth of information that ranges from character biographies, to episode recaps, to chapter summaries, and beyond. The story is famously complex, mostly due to the sheer number of characters - each of which have their own alliances, agenda, and loyalties. Many people find the television series too confusing because of the manifold character interactions, many of whom have similar names or might appear only once yet be vital to another part of the story.

The main objective of this study is to create a visual representation of the first season and book of *Game of Thrones*. We would like this visual representation to resemble narrative charts similar to the narrative charts from the popular webcomic xkcd as seen in Figure 0.1, but instead of creating them via manual effort, we endeavor to form them using as much automatically generated and extracted information as possible.

We strive for as much automation as possible in the production of our graph for diverse reasons. Primarily we are interested in time consumed and the ability to generalize the task. Each production of an entirely manually created narrative chart takes a significant amount of time. The first creation of a narrative chart by writing scripts and establishing automation methods also takes a significant amount of time but due to the nature of automation, every other chart created on a different corpus or for a different purpose takes considerably less time. For this gain in time we unfortunately have to sacrifice accuracy, as our automatic methods are not as precise as manual methods. However, this ability to create graphs more quickly is one of the benefits of automation vs. manual effort. With automation, we would also be able to create narrative charts on a wider scale to show to a populace whose interests vary more than the individuals creating the graphs manually (e.g., Romantic-Comedy movies, Science Fiction television series, political scandals on the news, etc.).

The narrative charts from xkcd seen in Figure 0.1, as described in the top text, show character interactions over time in a series of popular movies. Different lines converging indicates that the characters those lines represent, are together at that point in time. In The Lord of the Rings and Star Wars charts, location information is also provided, shown in the light grey oblongs over certain sections.

The multimedia corpus from which we would like to create our graph contains a variety of sub-corpora. It includes: books, video, audio, subtitles, synopses, manual transcripts, annotations, summaries, and character biographies.We need to organize these corpora in a way that can be molded into a coherent narrative chart. To achieve this, we will be using various alignment methods on various parts of the corpora, as well as information extraction and information retrieval. When we talk about alignment, we are referring to arranging sections of text from comparable corpora that contain similarities and correspond to each other. These sections of texts can

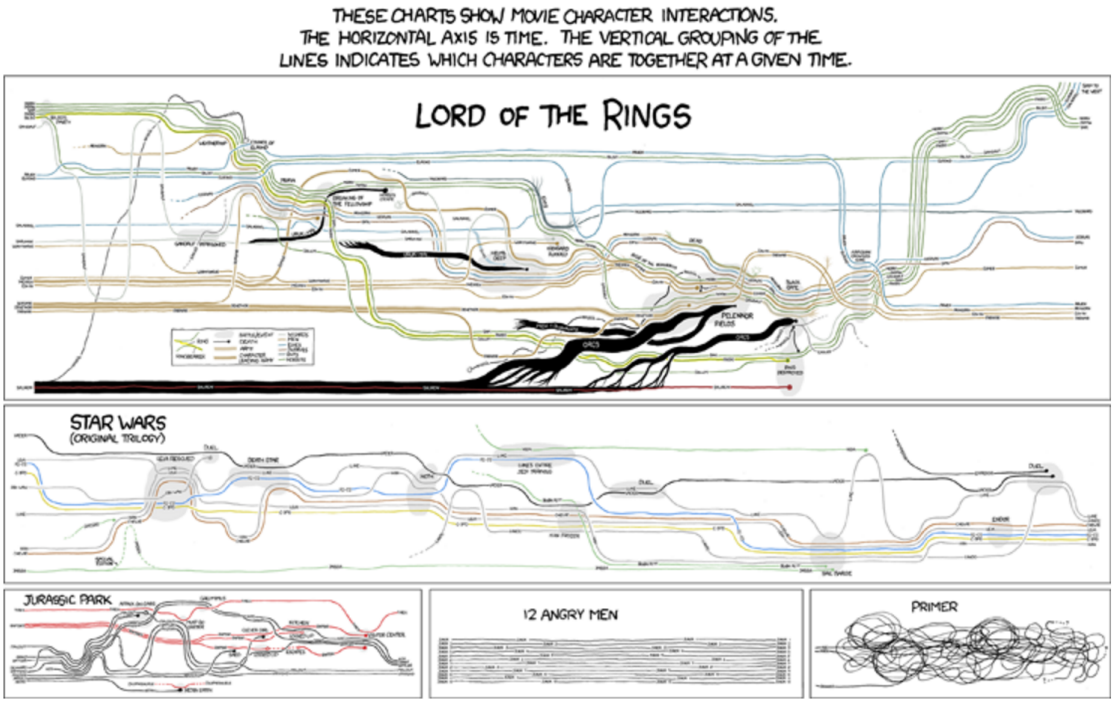



Figure 0.1: xkcd Narrative Charts

correspond to each other in a variety of ways: they can be talking about the same event or action, describing the same character or location, etc.

The various alignments that we perform in this study are the following:

- **characters & episodes** where we align characters into the episodes in which they appear
- **characters & scenes** where we align characters into the scenes in which they appear
- **subtitles & transcripts** where we align what a character says according to the subtitles and what a character actually says as heard by a person watching the series
- **transcripts & scenes** where we align dialogue with the scene in which it belongs
- **scenes & chapters** where we try to align the scenes from the television series with the chapters from the first book

Figure 0.2 shows a representation of which parts of our corpus are aligned with which other parts. The different colors of each node, as well as their labels, show the different sections of the *Game of Thrones* corpus that we use and the arrows show which sections will be aligned. From these alignments, we will extract all the pertinent information to form a narrative chart. We will create an initial graph with the data from our first alignment and then try to progressively improve the graph's appearance and accuracy with each subsequent alignment.

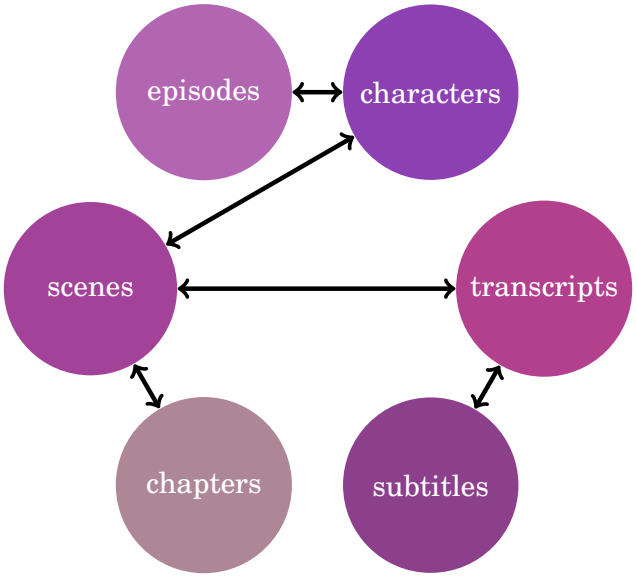



Figure 0.2: Alignment Representation

These narrative charts are interesting in that they allow us to create a richer text-based environment for deeper understanding of the subject. For example, a narrative chart showing the sequence of character interactions in an episode of *Game of Thrones* could help a confused viewer better understand at a glance what happened in the episode and prepare them to watch the subsequent episode without becoming lost in the storyline. In addition to the benefits of a graphic representation, the aligned information could be useful for better responding to text based queries. For example, one use could be to better search through video for concepts rather than direct dialogue (e.g., every time a character dies, scenes containing a direwolf, etc.).

This paper details the execution of the various methods used to attain the objectives for this thesis, including the alignment and extraction tasks performed and the graphs created. In Chapter 1 we discuss other studies whose alignment and extraction work is related to ours. The details of the corpus we use are examined in Chapter 2. Chapter 3 elaborates on the tools and methods used in this study, including the different alignments and extractions, and their evaluations for accuracy. Narrative chart results are shown in Chapter 4, and we follow with a discussion of the study and the results obtained in Chapter 5. A summary and possible expansions to this study are found in the Conclusion.

# Part I

# Context

CHAPTER 1

# RELATED WORK

**Contents**



## 1.1 Introduction

There are two main subtasks that need to be undertaken in order for us to create a narrative chart using the *Game of Thrones* corpus. The first is alignment between the sections of the corpora and the second is information extraction, so we can take pertinent information from the alignment results to put in the corpora. Our methods are built upon methods in current literature, however, unlike many of the studies, we focus entirely on the text aspect of our multimedia corpus. This both simplifies and complicates our task. On one hand, we do not need to worry about how to parse audio and video sequences to extract information from them. On the other hand, working with a text-only corpus increases the difficulty of determining whether a character is present or merely mentioned in a scene, and of distinguishing multiple characters that share a single name or single characters given multiple names. In this chapter, we discuss the works this study builds upon. We also look at some of the solutions these studies propose for commonly arising obstacles to alignment and extraction tasks.

## 1.2 Alignment Approaches

Using alignment methods to group multimedia corpora centered around television programs is a relatively recent area of interest in natural language processing research. [Gibbon, 2002], [Roy et al., 2014], and [Tapaswi et al., 2015] all used alignment methods in their studies. One of the main distinctions between alignment approaches is whether the corpora follow the same chronological sequence, as different methods will have worse to zero performance if used on the incorrect type of corpora.

Our work primarily builds upon the methods shown in [Roy et al., 2014]. They proposed three alignments: manual transcripts & subtitles, automatic transcripts & subtitles, and episode outlines & manual transcripts, for television shows *Game of Thrones* and *The Big Bang Theory*. To perform these alignments, they used the

same Dynamic Time Warping method we propose in this study (see Section 3.2.1), and they also used TF-IDF to improve their alignments. Due to the constraints given by Dynamic Time Warping, their corpora is limited to corpora that al follow the same chronological sequence.

[Tapaswi et al., 2015] also used *Game of Thrones*, as well as Harry Potter, in their alignment tasks. They aligned video scenes with book chapters, which means that, unlike [Roy et al., 2014], they used corpora not bound to the same chronological sequence. They used Dynamic Programming and face detection methods to find scene segmentations and align characters within those scenes. They then charted number of character occurrences from the scenes and chapters to get the similarity between them for alignment. They also make an important distinction from other studies in that they allow sections of their corpora to not be aligned rather than forcing an alignment. This allows them to account for scenes from the television series which do not appear in the book and vice versa.

[Gibbon, 2002] created an alignment between closed captions and transcripts in order to chronologically align transcripts to television news programs. Since their corpora follow the same chronological progression, they were able to use dynamic programming methods in their alignment. They distinguished their method from others by using a two step process, wherein they first aligned words from the closed captions and transcripts using word edit distance, a count of how many changes are necessary to turn one of the words into the other (e.g., "cat" and "hat" have an edit distance of 1 because only one letter changes), then used their findings to align at the sentence level based on the number of words in common.

Alignment methods are not always researched in order to better align television series. Alignment has often been studied for plagiarism detection improvement, as in [Sanchez-Perez et al., 2014]. Alignment can be useful in plagiarism detection because two sentences with content similar enough to be aligned could also contain copied or plagiarized text. TF-IDF is used to create sentence vectors in [Sanchez-Perez et al., 2014] to determine if two sentences are similar and potentially plagiarized. This is done by taking the cosine similarity of two sentence vectors, and determining those sentences are suspect if the cosine similarity is above a defined threshold. Like in [Gibbon, 2002], the number of words in common between the sentences is then used to determine plagiarism.

Alignment has also been used to create corpora for learning text-to-text rewriting rules, as in [Barzilay and Elhadad, 2003]. They addressed the issue of not having chronologically synchronous texts to align with Dynamic Programming by first using a clustering method to align paragraphs of text with similar topics. They postulated that paragraphs with similar topics are more likely to contain sentences that can be aligned with one another. Once aligned, they used a Dynamic Programming algorithm locally on the sentences in the aligned paragraphs using lexical similarity to determine if two sentences were aligned within the paragraph.

Our work incorporates many of these alignment techniques, notably by using Dynamic Time Warping methods and counts of tokens in common between documents. Our work focuses on text-based alignment methods whereas many of these studies mentioned include audio or video treatments. Where many of these studies focus on methods of alignment for better search results or for creating corpora to use in other natural language processing techniques, we focus on alignment for the purpose of creating a visual representation. This representation allows us to draw out a comprehensible story from a mass of text, for simple and automatic clarification of the

subject.

## 1.3 Information Extraction Approaches

Detecting certain kinds of information in a text is a commonly addressed problem in the field of natural language processing, with no perfect solution. We focus on the retrieval of two types of information from text: characters and locations.

### 1.3.1 Character Extraction

As discussed in [Vala et al., 2015], character identification is a more complex task than many assume, and cannot simply be resolved with a standard Named Entity Recognition tool. While these tools, when using specifically trained models, can identify the vast majority of characters, they can just as easily leave out minor characters whose references for example can be confused with a description of their function (e.g., "Mr. Bennet's coachman"). Recognition tools are also incapable of relating two names belonging to the same character together (e.g., "Mr. Darcy", "Fitzwilliam Darcy"). [Vala et al., 2015] proposed a technique to identify the various names of each character in a text. They started by using the Stanford NER tool to find characters, then applied a sequence of filters, mainly based on word context, to link the names.

[Roy et al., 2015] also proposed a method for detecting characters, though their study focused on finding the name of a speaker for a line of dialogue in a text. They postulated that a speaker can be identified using the content of their speech. They used four lexical approaches, including TF-IDF, to create speaker models to use in determining who said what. Both [Vala et al., 2015] and [Roy et al., 2015] are useful but in different ways. [Vala et al., 2015] concentrated on finding the various names of a single character in a text and [Roy et al., 2015] worked to abstract speakers whose names are not contained in the dialogues used in their experiments.

### 1.3.2 Location Extraction

The second type of information we are concerned with is location information. [Lingad et al., 2013] tested four different Named Entity Recognition tools, Stanford NER, OpenNLP, Yahoo!PlaceMaker and TwitterNLP, to determine their accuracy in extracting geographical locations and points-of-interest from Twitter data. They found that Stanford NER outperformed the other tools both with the out-of-the-box model and with models retrained specifically for their data.

Once NER tools extract location data from text, we still need to find a way to relate these extracted locations to other known locations. This problem is similar to the problem described for character extraction, where one character can have many names. Both [Scheider and Purves, 2013] and [Leidner et al., 2003] hypothesized about methods to determine a place's location relative to other locations given the location's context in the text (e.g., "New York, USA" vs. "York, UK").

## 1.4 Conclusion

In this chapter we discussed some of the most pertinent studies related to our thesis. We looked at different methods used in the current literature used for aligning text and extracting information, which our study uses and elaborates on. We saw that

Dynamic Programming techniques were commonly used to align chronologically synchronous corpora but more varied methods were tested with asynchronous corpora. The number of words in common seems to also be a commonly used metric for deciding if two texts are similar or not. We saw that Named Entity Recognition tools are a common solution to information extraction problems, though they introduce or at least fail to simplify other complexities such as relating multiple names to the same character. We will address some of these issues and describe our implementations of some of these methods in Chapter 3.

CHAPTER 2

# CORPUS

**Contents**



## 2.1 Introduction

This chapter gives information on the various aspects of our *Game of Thrones* corpus. We give details about the books, video, audio, subtitles, synopses, manual transcripts, annotations, summaries, and character biographies, and explain which of these are used in our experiments and to what extent.

## 2.2 Corpora

Our multimedia corpus can be seen as a compilation of smaller corpora all dealing with the same subject: *Game of Thrones*. We call these comparable corpora. Comparable corpora can typically be in different languages, some are included in this corpus in the audio and subtitles, but our study only deals with the sections in English.

### 2.2.1 Books

The book series is called *A Song of Ice and Fire* and is currently unfinished. As of the writing of this thesis, five novels have been published out of a planned total of seven. The corpus contains these five published books in plain text format, although this study only concerns itself with the first of the books: *A Game of Thrones*. This book contains 73 chapters and 293,655 total words. The chapters are used in the scene recap & chapter alignments in Section 3.3.5. An extraction of the most significant words per chapter, as calculated with TF-IDF, was also performed in Section 3.4.2.

### 2.2.2 Video

The corpus contains the video of the first season of the adapted television series. This video was extracted from the DVDs and provided by LIMSI. However, as our study deals with the processing of text information, the use of this video is outside the scope of this thesis.

### 2.2.3 Audio

The corpus contains the audio to the first season of *Game of Thrones*. The audio is in six different languages. This audio has also been provided by LIMSI and was also extracted from the DVDs. It is not used in our study.

### 2.2.4 Subtitles

The subtitles included in the corpus were collected by a team from LIMSI from the DVD version using Optical Character Recognition (OCR). They are complete for all DVD released seasons of *Game of Thrones* in eighteen languages. Subtitles include a line number, temporal stamp indicating where in the video a person says something and a line of dialogue or sound effect. On occasion, a subtitle will include two lines of dialogue or sound effects from different sources, this is shown by the two lines of dialogue appearing on different lines and also starting with a hyphen "-". Two subtitles showing these structures are given in Figure 2.1. One speaker's line can appear in multiple subtitles as they only contain as much text as can easily fit on the screen. The subtitles are used in the transcripts & subtitles alignment in Section 3.3.2.

```
137
00:16:30,799 --> 00:16:33,188
- (Pup cries)
- The direwolf is the sigil of your house.

138
00:16:34,519 --> 00:16:36,635
They were meant to have them.
```

Figure 2.1: Subtitles Format

### 2.2.5 Synopses

There is a short synopsis for each episode in the first season detailing very briefly what happens in that episode. They range from 52 to 114 words and contain on average 75 words. Due to their shorter length and relative word sparsity, they were not included in our alignments but were added to our training corpus for the doc2vec model described in Section 3.3.4. Figure 2.2 shows the synopsis for episode 3.

```
Jon Snow attempts to find his footing at Castle Black amidst
hostility and suspicion from his fellow members of the Night's
Watch. Eddard Stark and his daughters arrive at King's Landing.
Catelyn renews an old friendship with a member of the court,
while Daenerys and Viserys journey onwards to the Dothraki
capital of Vaes Dothrak.
```

Figure 2.2: Synopsis Format

### 2.2.6 Manual Transcripts

The manual transcripts are similar to the subtitles but include speaker information. However, they have been stripped of all temporal markings and line numbers. The lines of the manual transcripts are separated by speaker, rather than by text able to fit on a screen. They were made by fans on the internet and formatted by LIMSI. They contain an accurate representation of what each speaker in the episode says along with who said it. However they do not include dialogue in other languages (e.g., Dothraki) or sound effect information as in the subtitles. An example of two lines of the transcript is shown in Figure 2.3. The transcripts are used in the transcripts & subtitles alignment in Section 3.3.2 and the transcripts & screen recaps alignment in Section 3.3.4.

```
JON_SNOW : Lord Stark? There are five pups. One for each of
the Stark children. The direwolf is the sigil of your House.
They were meant to have them.
EDDARD_STARK : You will train them yourselves. You will feed
them yourselves. And if they die, you will bury them yourselves.
```

Figure 2.3: Manual Transcripts Format

### 2.2.7 Scene Recaps

The scene recaps provide summaries of each episode chronologically by scene. These differ from the summaries in that each scene is described in order, rather than having a text that is focused on summarizing by larger concepts. The scene descriptions vary in length with the scene they are summarizing. These recaps were written by fans and taken from the *Game of Thrones* Wiki.[1] They exist for the first six episodes of the first season. Each episode's recap contains on average 4,453 words. The scene recaps are used in the transcripts & scene recaps alignment in Section 3.3.4, and the scene recaps & chapters alignment in Section 3.3.5. We also use them for extracting names and locations to improve the narrative chart in Section 3.4.1.

### 2.2.8 Annotations

There are three annotation sections in this corpus. The first is a chapter to scene alignment from the [Tapaswi et al., 2015] results. This is in a JSON format and contains information about which chapters in the first book correspond to which sections of video. For each chapter included, the data also includes the chapter number and title. For each video included, the data also includes its episode number, start and end times for the section of video found for the chapter. The second annotation section contains scene demarcations. For each episode in the first season, the start and end times are provided for each scene. Lastly, the third annotation section has location information. This location information is provided with episode number, scene number, start and end times and has multiple levels of location precision (*Winterfell* is more specific than *the North* which is more specific than *Westeros*). An example of the location information can be seen in Figure 2.4.

[1] http://gameofthrones.wikia.com/wiki/Category:Recaps

```
4 4 301.1 384.9 Westeros the North Winterfell exterieur
4 5 384.9 586.7 Westeros the North Castle black exterieur
4 6 586.7 707.7 Essos Dothraki sea Vaes dothrak exterieur
4 7 707.7 988 Essos Dothraki sea Vaes dothrak interieur
```

Figure 2.4: Location Information Format

### 2.2.9 Summaries

There are four types of summaries for each episode of the first season of *Game of Thrones*. The first of these types is a long summary, which describes in some detail the happenings at each location shown in the episode (average length: 1,747 words). The second type is the short summary, which briefly describes the main actions of the episode (average length: 108 words). The last two are identical medium length summaries but in different formats, one in a plain text format and the other in JSON, with accompanying metadata such as the location and the episode number (average length: 809 words). The text for each of these summaries was written by fans on the internet. These summaries, with the exception of the summary in JSON format, were also used in training the doc2vec model in Section 3.3.4.

### 2.2.10 Character Biographies

The character biographies were taken from the *Game of Thrones* Wiki[2] using a script in Python we wrote using BeautifulSoup.[3] They deal with the characters primarily as they are in the television series than as they are in the books, as sometimes there is a difference. Depending on how important the character is and how often they appear, these biographies contain more or less information. They contain a history of the character separated by season, pictures of the character, a list of appearances by episode, a comparison with the book representation of the character, family tree, quotes, aliases, actor's name from the series, etc. As the information has been added by numerous fans of the series, it is not necessarily consistent across all characters. The list of appearances by episode was used in the alignment of characters to episodes in Section 3.3.3.

## 2.3 Conclusion

In sum, our corpus is comprised of smaller comparable corpora, primarily in English and all treating the subject of *Game of Thrones*. We looked at the structure of these various parts and will, in later sections, see how these corpora can be aligned.

[2] http://gameofthrones.wikia.com/wiki/Category:Season_1_Characters?display=page&sort=mostvisited

[3] https://www.crummy.com/software/BeautifulSoup/

# Part II

# Experiments

CHAPTER 3

# METHODS

**Contents**



## 3.1 Introduction

In this chapter we explore the various alignments and extraction techniques we use to incrementally improve the accuracy and completeness of our narrative chart depicting the first season and book of *Game of Thrones*. First we elaborate on the tools and libraries we use to perform our alignments, extractions, and charts. We then describe the process for aligning characters & episodes, transcripts & subtitles, characters & scenes, transcripts & scene recaps, and scene recaps & chapters. We also describe our processes for extracting the pertinent information to create a narrative chart. Lastly, we show our narrative chart improving over time with each subsequent alignment and extraction, and provide initial evaluations for the accuracy of our alignments.

## 3.2 Tools and Libraries

To accomplish our alignment, extraction, and graphing tasks we use an assortment of programming tools and libraries which will be elaborated on in this section. With the exception of the use of the charting libraries for the narrative charts, all programs were written in Python for version 2.7.12.

### 3.2.1 Dynamic Time Warping

Dynamic Time Warping, introduced in [Sakoe and Chiba, 1978], allows two sequences of different lengths to be mapped together in a non-linear way. For instance, given the same story, the time spent on specific details will differ between narrators in their respective documents. If we want to map the documents with each other line by line, we cannot do it linearly, as the lines from each document are “out of sync”.

Dynamic Time Warping attempts to solve this problem by non-linearly mapping subsections across documents.

To do so, we compute the distance matrix from comparing each subsection of a document, in our case spoken lines, with every other subsection from another document. This distance matrix is then used to create a third path that decreases the distance between the two documents. This path is found by using a distance measure, adaptable by the user, that is pertinent to the sequences. We extract various types of features to use in our distance measures to align the corpora such as tokens, nouns, lemmas, etc. The distance measure used for this algorithm is specified in each alignment section that makes use of it.

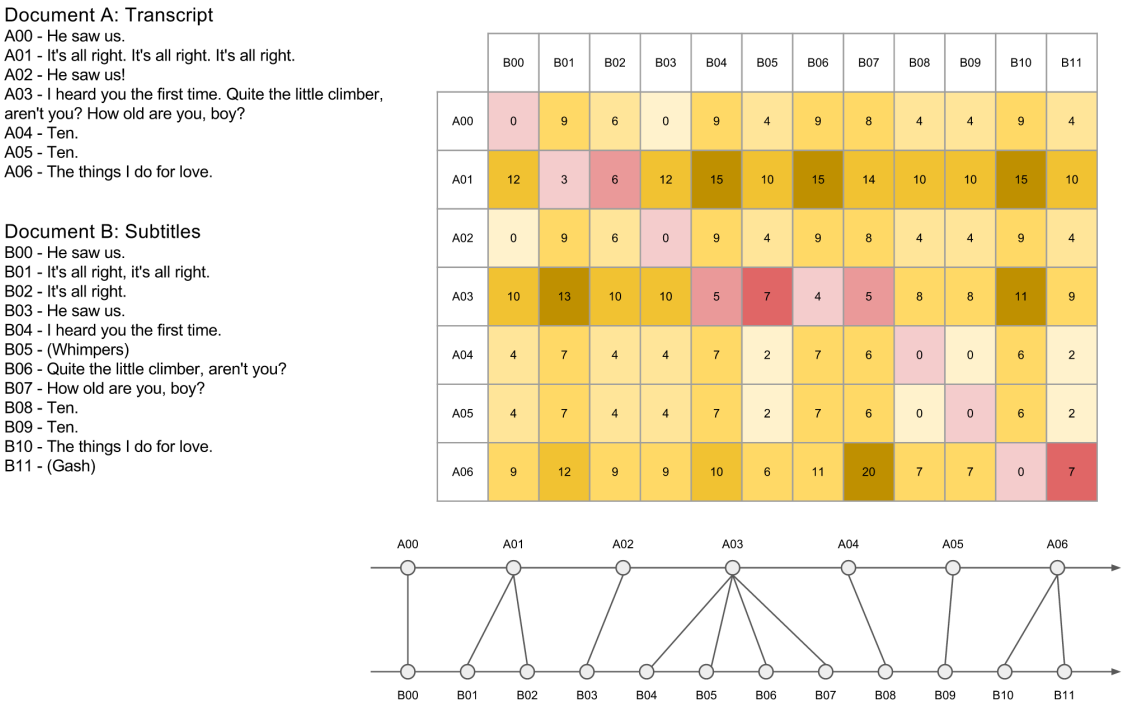



Figure 3.1: Dynamic Time Warping Example

Figure 3.1 shows an example of transcripts and subtitles being mapped together with the Dynamic Time Warping method. Distances between transcripts from document A and subtitles from document B are shown in the distance matrix on the right. Shades of yellow are used to show approximate distances (paler yellow means the lines are more similar and darker yellow means they are more different). The red shows the path created that minimizes the differences between the transcripts and subtitles. The graphic on the bottom shows how the red path is used to align the two sets of documents. Each subtitle on the lower line is paired with at least one of the transcripts on the upper line.

The Dynamic Time Warping path found must adhere to certain restrictions, those inherent to the algorithm and those specified by the user. There are three primary restrictions built in:

1. The path must start at (0, 0) and end at ($m$, $n$) ($m$ being length of sequence 1 and $n$ being length of sequence 2)

2. The path can only go forwards and cannot double back in time (i.e., the path can only go to the right, downwards, or diagonal down to the right)

3. The path can only advance one step at a time

Depending on the implementation, users can also specify whether they want to constrain the path's ability to go in a certain direction (e.g., the path cannot advance vertically), how far away from the diagonal the path can go, or the maximum incline of the path on the graph.

The algorithm uses backtracking to find the optimal path with the distance measure provided. This means that the path is not constructed during the distance calculations and can only be traced once all the distances have been calculated. Due to this, the path does not always go through the point containing the smallest distance, but it does find the best, least-overall-distance path considering all distances between the two sequences.

We used the pyannote library[1] implementation of this algorithm. The pyannote library contains a series of algorithm implementations for processing multimedia corpora.

### 3.2.2 Feature Extraction

To compare and find the distance between or similarity of the documents in our corpora, we need to extract features, or properties, to use in our assorted distance measures. We selected a number of different features from our corpora including tokens, lemmas, various parts of speech, word vectors, document vectors, and TF-IDF scores.

#### Tokens

The simplest form of feature extraction we used was an extraction of all tokens. We define a token for this study as a sequence of letters, numbers, or punctuation delimited by whitespace. They were found using the tokenizer feature of the Spacy library.[2] Spacy is an open-source library and Python API for common natural language processing tasks.

#### Lemmas

Another feature we extracted with the help of Spacy was lemmas. Lemmas are the canonical/base form of a word, without inflectional suffixes. Spacy strips words of these suffixes using WordNet data and an extension to cover pronouns. For example, “walked" and “wolves" become “walk" and “wolf". The definition of a lemma is further expounded in [Mair and Hundt, 2000].

#### Parts of Speech

We also use part-of-speech tags as a filtering method for which tokens to extract. Part-of-speech tags are identifier tokens that represent a word class (e.g., noun, verb, adjective, etc.) for a token. We tagged our tokens with Spacy's part-of-speech tagger, which uses the Penn Treebank tag set. The Penn Treebank tag set contains a wide

[1] https://github.com/pyannote
[2] https://spacy.io/

variety of part-of-speech tags including for example: 'VB' (base form of a verb), 'UH' (interjection), and 'NNS' (plural noun). A full explanation of part-of-speech tagging and Penn Treebank is found in [Santorini, 1990].

### Word2vec

Word2vec, introduced in [Mikolov et al., 2013], is a group of machine learning models that use shallow learning to create a vector space of words where each word is assigned to a different vector. Each value in the vector corresponds to different features of the words found in context (i.e., words before and after in a sentence) of the word throughout the training corpus. Words tend to have similar vector values to their synonyms or related words, because the vector values are based on the words' context and similar words tend to appear in similar contexts. For example, we can see how "prince" and "princess" would attain similar vectors if our training corpus contained the phrases: "The prince drank the poison", "The princess drank the wine", "The prince ordered his guards away", and "The princess ordered her maidens away". Both "prince" and "princess" appear in very similar word contexts. This method of creating word vectors also has an interesting side effect. Due to the relations between the word vectors, mathematical operations on the vectors can lead to interesting semantic results, for example: "woman" - "man" == "princess" - "prince".

We use the Spacy implementation of word2vec as well as Spacy's built-in word vectors, which were trained on data from Wikipedia using a dependency-based model from [Levy and Goldberg, 2014], in our experiments. We hypothesized that using word vectors instead of tokens in our distance measures could allow us to find more similarity between documents containing the same event in different but similar words. For example, this would allow us to compare scene recaps and book chapters even though the writing style and vocabulary used in the scene recaps, written by fans on the internet, differs from those of the books, written by George R.R. Martin.

### Doc2vec

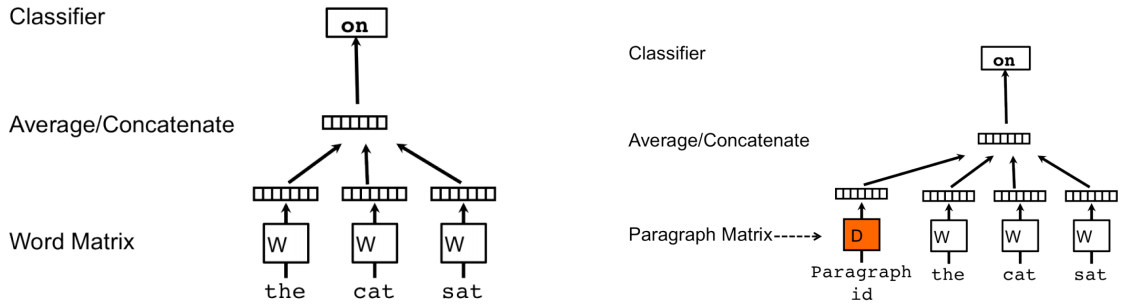
Doc2vec is a similar concept to word2vec. The two each contain word vectors calculated using word contexts, but doc2vec has an additional vector per paragraph that is calculated at the same time as the word vectors during training. This vector goes by many names (e.g., paragraph vector, document vector, sentence vector, etc.), but all describe a vector that is calculated using the same methods as in word2vec but on a group of words rather than an individual word. The doc2vec algorithm was proposed in [Le and Mikolov, 2014] and according to them, paragraph vectors improve performance over word vectors when comparing documents rather than individual words.

Since we also compare documents in our corpora, we use doc2vec to extract document vectors for use in one of our distance measures. For this we used the Gensim[3] implementation which is based on the aforementioned paper. Gensim is a Python library for semantic modeling and machine learning.

Figures 3.2a and 3.2b from [Le and Mikolov, 2014] show a representation of the difference between training a word2vec and a doc2vec model. It shows that they are identical but for an additional paragraph matrix that serves as the paragraph id in doc2vec. The vector representations of the words and the paragraph matrix are

[3] https://radimrehurek.com/gensim/

determined using the same methods. The paragraph id however, since it is trained using the words in the paragraph, keeps a sense of word order. This allows us to distinguish the phrases "the dragons ate the sheep" and "the sheep ate the dragons" as different, whereas in a word2vec implementation comparing individual words in a document, the two phrases would be identical.




(a) Word2vec representation

(b) Doc2vec representation

Figure 3.2: Doc2vec vs. Word2vec Representations

### TF-IDF

TF-IDF stands for term frequency–inverse document frequency. It is a method of finding important tokens in a document amongst many documents. The important tokens in a document are tokens that occur relatively frequently in that document and do not appear in a relative number of the rest of the documents. So for example, given the three documents:

- "Eddard of the House Stark"
- "the banners of House Stark and House Baratheon"
- "Robert of House Baratheon"

"Eddard" would be important in document 1, "banners" and "and" in document 2, and "Robert" in document 3. The other tokens ("Baratheon", "of", "the", "House", and "Stark") occur in too great a percentage of the other documents to be considered important to a single document.

The TF-IDF is calculated by taking the product of the term frequency and the inverse document frequency. The term frequency is the number $n$ of times a term $t$ occurs in a document $d$ divided by the number of words in the document $n_d$. The inverse document frequency is calculated by taking the log of the total number of documents $N$ over the document frequency $df$ of term $t$ (the number of documents in which the term appears at least once). Figure 3.3 shows these formulae in mathematical notation. TF-IDF is explained in greater statistical detail in [Jones, 1972].

We use TF-IDF in our alignments to try to group documents that share key tokens. This comes from the idea that even if many characters, places, and themes are recurring, there is probably a defining event in a document that singles it out from others. Hopefully documents from each of the corpora we are trying to align will contain tokens pertaining to this key event so they can be paired using the TF-IDF scores of these tokens.

$$tf_{t,d} = \frac{n_{t,d}}{n_d}$$

$$idf_t = \log \frac{N}{df_t}$$

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t$$

Figure 3.3: Formulae to Calculate TF-IDF

### 3.2.3 Named Entity Recognition

One of the main tools we use for extracting information from our alignments is Named Entity Recognition (NER). Named Entities are elements in a text which belong to one or more predefined categories (e.g., person, location, organization, date, etc.) and Recognition refers to our ability to extract these entities from a text and class them into the appropriate category. For example, in the sentence "Stark will never consent to leave Winterfell", ideally an NER tool will recognize "Stark" as a person and "Winterfell" as a location.

We tested both the Stanford NER and Spacy tools with our data. The outputs of the two programs were similar, and neither was completely accurate. Both Stanford NER and Spacy use machine learning techniques in their Named Entity Recognizers. Stanford NER uses a Conditional Random Field sequence model and Spacy uses a method similar to the one described in [Ratinov and Roth, 2009] but with the averaged perceptron.

### 3.2.4 Text Normalization

The experiments were tested with both preprocessed and non-preprocessed text. The preprocessing was performed in a Python function and deleted punctuation, made all the text lowercase, and expanded contractions (e.g., "they're" becomes "they are"). The results were generally better with preprocessed text and we report our results from the experiments using preprocessed text.

### 3.2.5 Charting Libraries

Two charting libraries were used to create the narrative charts. The first, Tapestry, was created by Nashville-based programming bloggers, the websages, for personal use though they open sourced their code on Github.[4] It uses a program in Javascript to parse and graphically illustrate character and episode data formatted in XML and JSON files. Tapestry was discarded after the first narrative chart in favor of another library that produces a tidier chart, more similar to the xkcd charts. This second library is called D3-layout-narrative and was created by an employee at the Australian Broadcasting Corporation, who also open sourced their code on Github.[5] It also uses a Javascript program to parse and illustrate data from a JSON file.

The other charts used throughout this paper to illustrate various paths and alignments were created in Python using Matplotlib, a popular Python plotting library.

[4] https://github.com/websages/tapestry
[5] https://github.com/abcnews/d3-layout-narrative

## 3.3 Alignments

In this section, we describe the methods used to perform our various alignments on the comparable corpora using the tools and libraries described in Section 3.2. We also present evaluations for the accuracy of the alignments performed in the form of F-measure comparison tables.

First we calculate precision as the percentage of alignments found that are correct, then we compute recall, which is the percentage of correct alignments found. Finally, the F-measure for each episode is calculated by taking the harmonic mean of the precision and recall. The F-measure ranges from 0 (worst) to 1 (best).

The three measures were calculated using the following formulae:

$$precision = \frac{\text{number of correct alignments found}}{\text{all alignments found}}$$

$$recall = \frac{\text{number of correct alignments found}}{\text{all expected alignments}}$$

$$F\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall}$$

### 3.3.1 Characters & Episodes Alignment

To create a baseline for our narrative chart, we needed to assemble the *Game of Thrones* data into the format understood by the charting library, which is a list of people per unit of time. We first decided to test an alignment of characters per episode in which they appeared. To gather this information, we wrote a web scraper in Python to browse the *Game of Thrones* Wiki using the BeautifulSoup web parser library. To obtain all of the character pages, our script identifies each character page from a list of the characters who appear in season 1, taken from the *Game of Thrones* Wiki[6].

The web scraper then visits each character page to download the character's biography. The HTML dump is then parsed with BeautifulSoup selectors to extract the desired information, namely the list of episodes in which each character appears. This information is finally formatted into the XML and JSON formats understood by the Tapestry library which will be used to generate the narrative chart. This format is shown in Figure 3.4.

Episodes each have an id, a name, a start time, a duration, and a list of the ids of all the characters that appear in that episode. Characters each have an id, a name, and a group. The group property is meant for grouping by family loyalty or other division, but is not explored in this study and could be interesting to pursue further at a later date.

Unfortunately this narrative chart did not help clarify the *Game of Thrones* character interactions, as seen in the Results chapter (Section 4). We hypothesise that

[6] http://gameofthrones.wikia.com/wiki/Category:Season_1_Characters?display=page

```
"episodes": [
    { "id": 1,
      "name": "Winter is Coming (episode)"
      "start": 10,
      "duration": 30,
      "chars": [3, 9, 10, 12, 13, 18, 19, 23, 25, 30, 41, ...]
    },
    ...

<characters>
    <character group="0" id="0" name="Aemon" />
    <character group="0" id="1" name="Allo" />
    <character group="0" id="2" name="Armeca" />
    ...
```

Figure 3.4: Data Format for Character & Episode Alignment in Tapestry

some other division of time throughout the series is necessary to improve the granularity of the graph. We decided to divide the task into more manageable units and make a chart per episode with the characters aligned by scenes in which they appear rather than by episodes. This will improve the clarity of the graph, bringing it closer to its purpose of showing information in an as transparent and meaningful way as possible. The process for this alignment is examined in the following sections.

### 3.3.2 Transcripts & Subtitles Alignment

In order to improve the graph and create an alignment of characters to scenes we first needed to know when each character speaks during the episode. Using the scene segmentations in the corpus acquired from the book2movie, [Tapaswi et al., 2015], results, and the timestamp included in the subtitle format, we are able to place the speaker into a particular scene. As stated in their article, the authors find scene boundaries using a dynamic programming technique suggested in [Tapaswi et al., 2014], which is said to not be 100% accurate but with no mention of precisely how accurate the result are. Nevertheless, manual verification revealed that the results were accurate within the range of human error and it was decided that it would be enough to yield exploitable results.

The goal for this alignment was to match each subtitle, as seen in the upper left of Figure 3.5, (whose format consists of a line number, a timestamp and a spoken line), with a transcript, as seen in the upper right of Figure 3.5, (whose format consists of the name of the speaker and the spoken line). It is worth noting that the transcript usually contains the exact line, as heard by an annotator, while the subtitles can sometimes contain a shortened variation, according to screen space constraints.

Once matched, we combined the two items into a new format so that each subtitle had a speaker name associated to it. The format of the result is seen in the lower line of Figure 3.5. It consists of a timestamp in seconds, the speaker name and the spoken line. Using the line from the transcript rather than the line from the subtitle would make some cases more orthographically correct, but the transcript lines sometimes spreads over many subtitle lines for one speaker. It would be impractical to split the

transcript lines in a way that exactly coordinates each sub-line of the transcript to the subtitle timestamps.

```
14
00:03:24,280-->00:03:25,679   WAYMAR_ROYCE:
Do the dead frighten you?     Do the dead frighten you?

                                        ⇓

204.280 205.679 WAYMAR_ROYCE Do the dead frighten you?
```

Figure 3.5: Data Transformation for Transcripts & Subtitles Alignment

This alignment was performed using a script forked from the work of [Roy et al., 2014], and for this alignment the script was only slightly modified. It uses the Dynamic Time Warping module from the pyannote library to find the best overall match, taking all data into consideration, of each of the subtitles to the transcripts. This script, written in Python, proved essential, not only for this alignment, but also for the alignment of the transcripts with the scene recaps.

The script takes two sets of data, in our case: parsed subtitles and transcripts, and matches them according to the dynamic time warping algorithm constraints (described in Section 3.2.1), including user defined constraints, and a comparison function specified by the user which measures distance between sequences.

In this case, no extra constraints were specified. The distance is computed by counting common tokens in each line. Note that there is no text normalization or lemmatization used in this step. The algorithm takes the distance from each subtitle and transcript and creates a distance matrix, in which it then looks for the best path between the two sequences of texts. We established earlier that the best path is the one that minimizes the overall distance between the two sequences.

The evaluations in Table 3.1 were obtained by comparing the automatically generated results with a manual alignment. This evaluation process is further detailed in Chapter 4.

| | Precision | Recall | F-measure |
|---|---|---|---|
| Episode 1 | 0.811 | 0.809 | 0.810 |
| Episode 2 | 0.863 | 0.861 | 0.862 |
| Episode 3 | 0.899 | 0.898 | 0.899 |

Table 3.1: Transcripts & Subtitles Alignment Evaluation

The results are favorable but somewhat lower than what one might expect in this alignment, given that the subtitles are very similar if not identical to the transcript of what is actually said in the scene. We explain this drop in accuracy primarily by the occurrence of sound effects in the subtitles: The subtitles incorporate environmental noises such as "(Horse snorts)" and "(speaking Dothraki)" but the transcripts make no mention of them. This created a misalignment because one of the constraints is that every subtitle needs to be aligned with a transcript. However, the sound effects were left in the data because we recognized that they would be useful in later types of alignments such as a subtitle & chapter alignment. This study does not include theses alignments but they would be an avenue to explore in further studies. Other

errors were due to the occurrence of interjections such as "uh" or "oh", or a character's name being called which is missing from either the subtitles or transcripts.

### 3.3.3 Characters & Scenes Alignment

As mentioned earlier at the end of Section 3.3.1, to improve our current version of the narrative chart, we need to align characters not by the episodes in which they appear, but by scenes. To do this we first needed to create an alignment between the subtitles and the transcripts in order to know who talked when in the series, which we did in Section 3.3.2. We are now going to use this information to divide the result of the subtitles & transcripts alignment into the scenes in which they belong. To do this we are going to use the scene segmentations from the book2movie data.

At this stage it was a simple matter of using a script in Python to compare the name of a character speaking to a list of characters and associated ids (generated in alphabetical order), and associate that id with the correct scene based on the timestamp. The start time for the aligned subtitle-transcript was compared with each scene's start and end times, and the character id was placed in the scene when the timestamp fit chronologically between the scene's start and end.

One limitation of this particular method is that characters without speaking lines but who are nonetheless present in the scene are ignored. This issue will be further addressed in Section 3.4.1.

### 3.3.4 Transcripts & Scene Recaps Alignment

To enhance the narrative chart further, we will concentrate on adding more and varied information to it. To do this, we need to continue our alignments to gather this supplementary information. We try various different methods to achieve the best alignment possible in the section. This is because the best possible alignment will contain the most accurate information to add to our narrative chart. This section will elaborate on the ten different ways we attempted to align transcripts & scene recaps. For these alignments, we again used the Dynamic Time Warping script mentioned in earlier sections. The script was modified for each section to extract different features for each of the tests. The different features we extract are tokens, lemmas, word embeddings, document embeddings, and parts of speech. The basic algorithm is unchanged but we wrote in an added constraint on the direction that the path can take when trying to find the optimal line from the start of the transcripts and recaps to the end. This constraint was added so that multiple transcripts could be aligned to a single scene but a single transcript could not be aligned to multiple scenes. This means that all dialogue is contained within the scene, and no one can be said to talk over a scene transition. Two tables, Table A.1 and Table A.2 showing the results of all the evaluations in this section side by side can be found in the Annex.

#### Tokens in common

The first two ways we try to align transcripts (noted #1 and #2) with scene recaps are very similar. The features extracted from the two series of texts are tokens. The tokens, as mentioned above have been preprocessed but they have not been lemmatized or stemmed. The distance measure used for the Dynamic Time Warping algorithm is, like in Section 3.3.2, number of tokens in common. The difference between the two distance measures is the direction in which we calculate the number

of tokens in common. The first method subtracts the number of common tokens from the number of tokens in the scene recap to use the number of unique tokens in the scene. The second method subtracts the number of common tokens from the number of tokens in the transcript to use the number of unique tokens in the transcript. The latter option works considerably better (as we can see in Table 3.2) as the transcripts are considerably shorter than the scene recaps, and so have fewer tokens that can be shared across other transcripts.

Method #3 uses the same distance measure of tokens as the second method, but normalized. This normalization is performed by dividing the result of the second distance by the total number of tokens in the transcript. This attempts to diminish the bias of longer transcripts versus shorter transcripts, the idea being that longer transcripts have more opportunities to have tokens in common with the scene than shorter transcripts do. This normalization gives more accurate results in some episodes but not on average as seen in Table 3.2.

Figures 3.6, 3.7, and 3.8 contain the formulae used to calculate methods one, two and three. We define the words contained in the transcript, $t$, as $t = \{w_1^t, w_2^t, ..., w_n^t\}$ and the words in the scene recap, $s$, as $s = \{w_1^s, w_2^s, ..., w_m^s\}$.

$$d_1(t,s) = |s| - |t \cap s|$$

Figure 3.6: Method #1 Formula

$$d_2(t,s) = |t| - |t \cap s|$$

Figure 3.7: Method #2 Formula

$$d_3(t,s) = \frac{|t| - |t \cap s|}{|t|}$$

Figure 3.8: Method #3 Formula

| | Method #1 SR - (T + SR) | Method #2 T - (SR + T) | Method #3 T - (SR + T) (norm) |
|---|---|---|---|
| Episode 1 | 0.006 | 0.854 | 0.864 |
| Episode 2 | 0.003 | 0.679 | 0.754 |
| Episode 3 | 0.014 | 0.893 | 0.825 |
| Episode 4 | 0.009 | 0.920 | 0.868 |
| Episode 5 | 0.009 | 0.806 | 0.780 |
| Average | 0.008 | 0.830 | 0.818 |

Table 3.2: Transcripts & Scene Recaps Alignment Evaluation - Tokens in Common

Table 3.2 shows a comparison of F-measures for the alignments tested above for the first five episodes. We can see that the first distance measure method (unique tokens in scenes) is very ineffective. Distances measure methods #2 and #3 (unique tokens in transcripts and unique tokens in transcripts normalized) are much more productive.

**Lemmas in common**

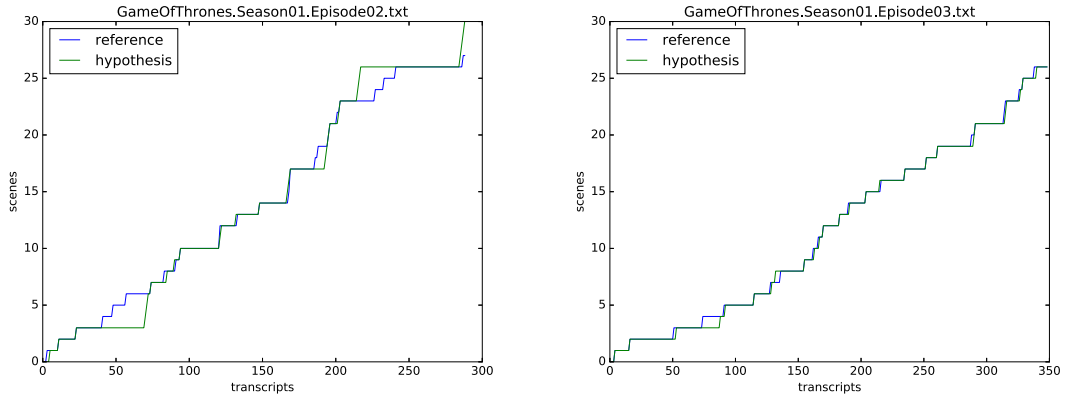
The next step was to try to compare more tokens by using the token's lemmas instead of the entire word. This would allow us to compare tokens with different suffixes like "walked" and "walking" that might appear in each of the texts but describe the same thing.

The same structure was kept for the distances as the structure we had in method two and three. Method #2' uses the same formula as is shown in Figure 3.7 and

method #3' uses the formula shown in Figure 3.8, however, we redefine $t$ and $s$ to contain lemmas instead of words ($t = \{l_1^t, l_2^t, ..., l_n^t\}$, $s = \{l_1^s, l_2^s, ..., l_m^s\}$). The number of unique lemmas in the transcripts was first used and then for our second experiment, we took this number and divided it by the total number of lemmas in the transcripts, as we did in the normalization section previously. Table 3.3 shows a comparison of F-measures of these two new alignment methods, (named #2' and #3'). We can see in this table that method four, the one without normalization performed better this time. This contrasts the previous experiment where the normalization was better. A reversal of leading values like this could indicate that the normalization perhaps does not have as much effect on the alignment as was previously thought.

| | Method #2' T - (SR + T) | Method #3' T - (SR + T) (norm) |
|---|---|---|
| Episode 1 | 0.867 | 0.851 |
| Episode 2 | 0.716 | 0.689 |
| Episode 3 | 0.839 | 0.862 |
| Episode 4 | 0.905 | 0.863 |
| Episode 5 | 0.806 | 0.764 |
| Average | 0.826 | 0.805 |

Table 3.3: Transcripts & Scene Recaps Alignment Evaluation - Lemmas in Common




(a) Episode 2

(b) Episode 3

Figure 3.9: Transcripts & Scene Recaps Alignment Path - Lemmas in Common

To aid in the visualization of these evaluation scores, we plotted the alignment path found with method #2' (lemmas in common) with the alignment path decided upon manually. The worst performing episode, episode 2, is shown on the left in Figure 3.9a and the best performing episode, episode 3, is on the right in Figure 3.9b. We can see that the lines mostly overlap in episode 3, showing a close correlation between the manual alignment and the automatic alignment, whereas in episode 2, the lines diverge and show at what point the algorithm did not correctly align the transcripts and scene recaps.

We hypothesize that these errors are a result of having a significant number of similar functional words in those sections. The original form of these functional

words (e.g., “is”, “was”, “are”), as used in Section 3.3.4, provides more of a distinction between the scenes and transcripts than when all are lemmatized (e.g., “be”, “be”, “be”). When more of the features are shared across scenes or transcripts in a sequence, the distinction between them is blurred, making it more difficult for the Dynamic Time Warping algorithm to find the correct path.

**Word2vec**

Two distance measures using word2vec were tested to align scene recaps and transcripts.

The first method using word2vec, method #4 uses a sum of all the word vectors in a document to compare its similarity to summations of word vectors of other documents. This is similar to the idea of doc2vec which is discussed in Section 3.3.4, although with a different implementation. The distance is calculated by first finding the vectors for each word in each document using the model included in the Spacy library. The vectors for each document are then added together to form one single vector. The Dynamic Time Warping Algorithm then takes the cosine similarity of these vectors (each transcript with each scene recap) to create its distance matrix.

For the second method using word2vec, method #5, we compute the word vector for each word in a document and then individually take the distance between them and individual word vectors from each other document. The mean of the smallest distance of the individual word vectors from each pair of documents was kept as the distance for that pair of documents. Then, using these distances, we can use the Dynamic Time Warping algorithm to determine the pairs of aligned documents on an episode scale.

Figures 3.10 and 3.11 contain the formulae used to calculate methods four and five. We define the word vectors contained in the transcript, $t$, as $t = \{v_1^t, v_2^t, ..., v_n^t\}$ and the word vectors in the scene recap, $s$, as $s = \{v_1^s, v_2^s, ..., v_m^s\}$. $d_{\cos}$ is the cosine distance.

$$d_4(t,s) = d_{\cos}\left(\frac{1}{|t|}\sum_{v\in t} v, \frac{1}{|s|}\sum_{v'\in s} v'\right)$$

Figure 3.10: Method #4 Formula

$$d_5(t,s) = \frac{1}{|t|}\sum_{v\in t}\left(\min_{v'\in s}\left(d_{\cos}(v,v')\right)\right)$$

Figure 3.11: Method #5 Formula

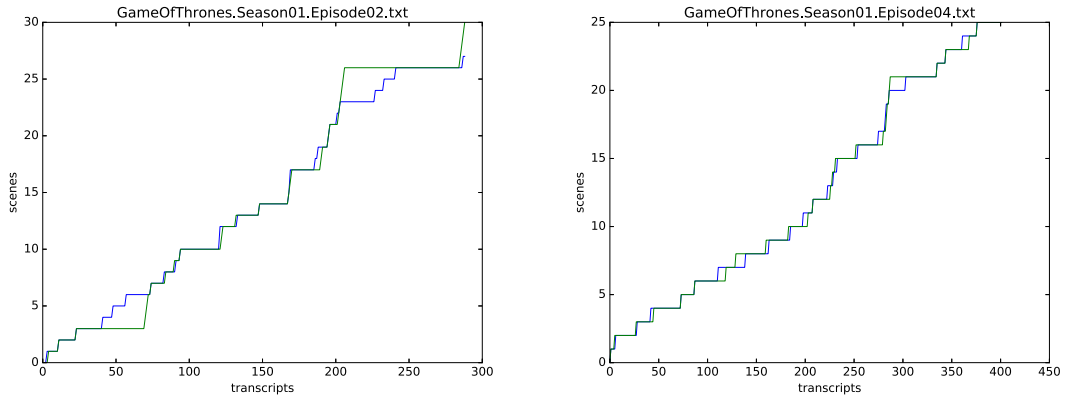
| | Method #4 Averaged | Method #5 Individual |
|---|---|---|
| Episode 1 | 0.589 | 0.851 |
| Episode 2 | 0.490 | 0.699 |
| Episode 3 | 0.593 | 0.808 |
| Episode 4 | 0.778 | 0.823 |
| Episode 5 | 0.198 | 0.757 |
| Average | 0.529 | 0.787 |

Table 3.4: Transcripts & Scene Recaps Alignment Evaluation - Word2vec

Table 3.4 shows a comparison of F-measures for the results of these two distance measures using word2vec. We can see that the distance measure using individual word vectors performed better than the one using averaged word vectors. There was a clear issue in the alignment of episode 5 in the former alignment. This is because

some transcripts only contained a name or other information not recognized by our word2vec model, so the result of that word was undefined. The Dynamic Time Warping algorithm needs all distances to be defined so to resolve this issue and continue the alignment, these undefined values were then given an arbitrary value (equal to the maximum distance) but these values had a negative effect in the Dynamic Time Warping algorithm and ruined the accuracy of the alignment.

As in the previous section, we plotted the automatic alignments for each episode along with a manual alignment to help see where the algorithm erred in its alignment. For this visualization we chose to show the worst and best alignment results from the better-performing word2vec distance measures, the one using individual word vectors rather than averaged word vectors. We can see in Figure 3.12a that episode 2 is still the episode with the worst alignment results. We can even see that there is a similar shape in the errors when compared to the path in Figure 3.9a. The exact reason could not be ascertained upon closer inspection of the data, especially in the first diversion around scene three. The diversion around scene 27 however is somewhat more clear. The scene recap for scene 27 is much longer than the recaps surrounding it and therefore contains more possibilities with with it can be matched. In addition the dialogue belonging to scenes 25 and 26 is quite short and contains the same characters as scene 27, so there is insufficient data to make a scene division decision in the transcripts. For this distance measure, episode 4 had the best results as seen in the way the two path lines mostly overlap, and confirmed in the results in Table 3.4.




(a) Episode 2

(b) Episode 4

Figure 3.12: Transcripts & Scene Recaps Alignment Path - Word2vec

Figures 3.12a and 3.12b represent the difference between these two word2vec distance measures. The blue squares represent individual words in a scene document and the red dots are individual words in a transcript document. We can see in Figure 3.12a showing a representation of the averaged vectors as a line between averaged scene recap (turquoise triangle) to averaged transcript vectors (pink triangle), that the distance using the averaged vectors can easily be skewed by having a great number of common words, such as “the”, “of”, etc., in common but less easily takes into account a single occurrence of a very unique word. This could hamper alignments between documents that both contain many common words and even the same characters or places, but each also contain a relatively infrequently occurring key word or

phrase that could identify them as a pair. The triangles represent the result of the averaged vectors and the line connecting them shows the alignment made.

On the right, in Figure 3.13b showing the individual vector comparisons, the distance between a red dot is taken with each of the blue squares and the average of the smallest distance between each of the red dots and blue squares is kept as the alignment distance for those two documents. We can see that if the unique word is repeated in both the scene and the transcript (or, according to word2vec properties, a very similar unique word), we can give more weight in the alignment to this unique word in the document alignment. This allows us to compare key words in the alignment rather than using a generalization using all the words in the document.

(a) Averaged Vectors (b) Individual Vectors

Figure 3.13: Word2vec Alignment Representations

### Doc2vec

Unlike in Section 3.3.4, we did not use a pre-trained vector model. This is because, for doc2vec, the paragraph vector is trained at the same time as the word vectors in order to have the paragraph id be a vector containing word order. We used the default settings in Gensim to train our model as they are as close as possible to what was presented in [Le and Mikolov, 2014], except for the number of epochs which was increased from 5 to 10, and used everything in our *Game of Thrones* corpus that was available in plain text format, preprocessed, in the training corpus.

Our doc2vec distance was calculated using the same equation as method four, shown in Figure 3.10. To use this equation with doc2vec, we redefine $t$ and $s$, adding the paragraph vector $v^p$, as $t' = t \cup \{v^p\}$ and $s' = s \cup \{v^p\}$.

| | F-measure |
|---|---|
| Episode 1 | 0.658 |
| Episode 2 | 0.765 |
| Episode 3 | 0.819 |
| Episode 4 | 0.800 |
| Episode 5 | 0.801 |
| Average | 0.769 |

Table 3.5: Transcripts & Scene Recaps Alignment Evaluation - Doc2vec

The results show a definite improvement on our naive document vector implementation (the first word2vec alignment), but do not quite attain the performance level of the second word2vec alignment. Undoubtedly, this could be improved upon by experimenting with the training settings, the number of training epochs and the amount of training data.

### Part-of-Speech selection

In an attempt to improve the previous word2vec alignments, we decided to compare words based on their part of speech, rather than all words. We hypothesized that the words that would have the highest impact on the alignment only belong to a few linguistic categories, the others being too common throughout the documents to declare a difference from one document to the next. The parts of speech we chose to compare are: nouns, main verbs, adjectives and combinations thereof. Both the summed vectors and individual words alignments were redone with the difference being that before the documents are fed into the algorithm for distance measuring, they are first filtered to contain only the words with the part of speech desired. To do this all the words were first tagged with their part of speech using the Spacy library which performs within 1% of the state of the art on this task.

| Part-of-speech | F-measure |
|---|---|
| all words | 0.477 |
| nouns, adjs, verbs | 0.048 |
| nouns, verbs | 0.048 |
| nouns | 0.021 |
| verbs | 0.012 |
| Average | 0.121 |

Table 3.6: Transcripts & Scene Recaps Alignment Evaluation - Part-of-speech Selection

The results for this section were somewhat surprisingly low. We postulate that perhaps there were not enough words with the desired part of speech to accurately compare the relative placement of each document in the Dynamic Time Warping algorithm. To make up for this word deficiency, we try to compare more than one transcript at a time, as described in Section 3.3.4.

### Moving window

The last alignment method tested for transcripts and scene recaps was what we call a 'moving window'. We use this method to help improve the part-of-speech selection method. We call this method a moving window because we compare a different and subsequent selection of transcripts instead of one transcript at a time. For each current transcript we want to find the distance of with each scene, we also take the transcript that comes immediately before and the transcript that comes immediately after. Figure 3.14 shows a representation of this method. Each dot represents a transcript. The red dot is the current transcript and the box shows what transcripts are taken in context with it. This method is used to give us more tokens to work with and more context for each transcript as some of the transcripts can be quite short or very reliant on surrounding context to make sense.

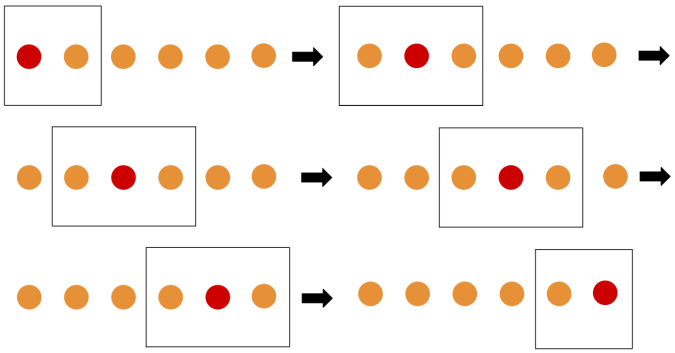


Figure 3.14: Moving Window Representation

We then filter the tokens as described in the previous section to compare only the nouns and verbs of each transcript and scene. Next, we take the word vectors of these nouns and verbs, and use the individual word vector method to find the cosine similarity of the two documents using the word vectors with the smallest distance of the word vectors calculated.

| | F-measure |
|---|---|
| Episode 1 | 0.852 |
| Episode 2 | 0.672 |
| Episode 3 | 0.805 |
| Episode 4 | 0.853 |
| Episode 5 | 0.774 |
| Average | 0.791 |

Table 3.7: Transcripts & Scene Recaps Alignment Evaluation - Moving Window

Table 3.7 shows considerable improvement over the part-of-speech selection alignment (we can compare an average F-measure of 0.121 for the part-of-speech selection with the average F-measure obtained with this method: 0.791) and also slightly improves our scores from both word2vec alignments and the doc2vec alignment (refer to Tables A.1 and A.2 for a side by side comparison). This implies that our other alignments probably suffered for a lack of text with which the method can use. In Section 3.3.5 we move on to a new alignment that suffers less from an insufficient quantity of text.

### 3.3.5 Scene Recaps & Chapters Alignment

This section, like Section 3.3.4, details various alignment methods to align two sections of the *Game of Thrones* corpus so that we can augment our narrative charts with more information. Here we chronicle four methods to align scene recaps from the episodes with chapters from the first book. Unlike our previous alignments, we cannot use the Dynamic Time Warping algorithm. This is because these two sets of documents do not necessarily follow each other in chronological order. An additional difficulty occurs in that significant sections of both the book and the episodes do not occur in each other. To overcome these obstacles, we use a simpler comparison method where the smallest distance between two sections indicates an alignment, and in

certain cases we introduce a cut-off value to specify that a section has no alignment. We say that one chapter could be aligned with multiple scenes but each scene only has one chapter it can be aligned with, if at all, since some scenes in the series do not occur in the books. Table A.3 in the Annex shows a comparison of all the evaluation methods used in this section side by side.

### Tokens in common

We create a baseline for this alignment by starting with a very simple method: tokens in common. Text is preprocessed and tokens are found in the same way as has been described in earlier sections. The a distance matrix is also created in the much the same way as earlier alignment methods, with the number of tokens not in common stored in a matrix for each scene and chapter. Then we take the lowest of these numbers and declare that chapter and scene aligned. For this alignment we did not have a cut-off value as the distance scores were all very close and no pattern in the values of correct and incorrect alignments was found.

| | F-measure |
|---|---|
| Episode 1 | 0.338 |
| Episode 2 | 0.355 |
| Episode 3 | 0.333 |
| Episode 4 | 0.615 |
| Episode 5 | 0.308 |
| Average | 0.389 |

Table 3.8: Chapters & Scene Recaps Alignment Evaluation - Tokens in Common

The results for this tokens-in-common method serve as our baseline for the chapter & scene alignments but are not terribly impressive as seen in Table 3.8. One of the main issues is that many scenes in the episodes do not appear in the books. Another great difficulty is that many of themes throughout the books and series are similar and use similar vocabulary. Without the constraint of chronological order, an early scene featuring a character can easily be aligned with a later chapter featuring that same character, creating a misalignment.

### Lemmas in common

We try to improve upon our baseline by comparing the number of lemmas in common. The same steps were taken as in the tokens-in-common method with the exception that before the tokens can be compared between the scenes and chapters, they are first lemmatized using the Spacy library. Again, no cut-off value was used.

In hindsight, the fact that the average performance for lemmas in common is lower than that of tokens in common is unsurprising. Since there are fewer constraints on what can be aligned with what, less precision in word form can make more of a difference in performance.

In addition to this difficulty, the other difficulties mentioned earlier in Section 3.3.5, such as scenes from the television series not appearing in the books, and similar themes and vocabulary, still apply.

| | F-measure |
|---|---|
| Episode 1 | 0.225 |
| Episode 2 | 0.355 |
| Episode 3 | 0.481 |
| Episode 4 | 0.462 |
| Episode 5 | 0.308 |
| Average | 0.366 |

Table 3.9: Chapters & Scene Recaps Alignment Evaluation - Lemmas in Common

**TF-IDF**

For this next section, each chapter and scene is transformed into a TF-IDF vector using Scikit-Learn. Scikit-Learn is a Python library containing supervised and unsupervised machine learning algorithms. The TF-IDF vectors for each document contain all the TF-IDF scores of the individual words in that document. Once the vectors for each document are calculated, the cosine similarity of each scene is then taken with each chapter and the two with the maximum similarity is kept as the alignment for for each scene.

| | F-measure |
|---|---|
| Episode 1 | 0.479 |
| Episode 2 | 0.452 |
| Episode 3 | 0.593 |
| Episode 4 | 0.5 |
| Episode 5 | 0.308 |
| Average | 0.466 |

Table 3.10: Chapters & Scene Recaps Alignment Evaluation - TF-IDF

The results for this method, as seen in Table 3.10, show a general improvement over the scores from our baseline and the scores from the lemmas in common method. This is likely due to the fact that we are comparing only important words rather than all words in a document.

**TF-IDF with Transcripts**

As we saw in our moving window experiments in Section 3.3.4, sometimes more text used in the alignment can lead to better results. We also saw that some dialogue from the transcripts was taken verbatim from the books but that this dialogue was not frequently repeated in the scene recaps. To make use of these observations we conduct an experiment using a triple alignment between chapters & scene recaps & transcripts, instead of the two document pairs we have thus far been working with. However, this alignment was not performed using the three sets of documents at once. We first executed a scene recaps & transcripts alignment using the method which had the best alignment results with a 0.83 F-measure: tokens in common. We then consider that all dialogue from the transcripts aligned with a scene is part of the scene and included in the total text used in the scene recaps & chapter alignment. This second alignment is performed exactly as in Section 3.3.5, just with the addition of the extra text in the scene recap.

| | F-measure |
|---|---|
| Episode 1 | 0.563 |
| Episode 2 | 0.484 |
| Episode 3 | 0.519 |
| Episode 4 | 0.615 |
| Episode 5 | 0.385 |
| Average | 0.513 |

Table 3.11: Chapters & Scene Recaps Alignment Evaluation - TF-IDF with Transcripts

Table 3.11 contains the best overall results attained for the chapters & scene recaps alignment, which reflects and supports the ideas we had prior to testing. Further developing experiments on multiple alignments such as this could be an interesting and productive expansion to this study.

## 3.4 Information Extraction

In this section we extract information from some of our best-performing alignments in order to enhance the narrative chart by adding in the information, in as clear and as aesthetically pleasing a way as possible. We use a variety of methods described in Section 3.2 to extract characters, locations, and important words from the text.

### 3.4.1 Character and Location Extraction

As mentioned in an earlier section, one of the main limitations of our method is that is could not recognize characters in scenes with non-speaking roles. Those characters have therefore thus far been left out of the narrative charts. Other information discussed earlier but hitherto left off the chart is location information. We do have a list of locations in the annotations section of our corpus, but it was our intention to test a text-based recognition tool on our corpora to find the locations and show how it could be done using the methods in Section 3.2. To overcome this impediment and find these previously invisible characters and unidentified locations, we will use the information from the transcripts & scene recaps and scene recaps & chapters alignments. This is done by concatenating the aligned texts to form one document per scene to search within. We use two methods to extract the characters and locations: Named Entity Recognition and List Matching, which is similar to our tokens in common method, as it takes tokenized text from one document and compares it to the tokens of another document.

#### Named Entity Recognition

The first method of information extraction tested was Named Entity Recognition using Stanford NER[7] and Spacy tools. Stanford NER has multiple pre-trained models available but the most accurate on our corpus, using a manual comparison with a manual extraction, performed at the same level as the built-in model of Spacy so for ease of use, the results reported were ultimately extracted using Spacy. Samples

[7] http://nlp.stanford.edu/ner/

of the extractions are shown in Tables A.6 and A.5 in the Annex. The tools were tested on concatenated alignment text using the TF-IDF with transcripts method to align transcripts & scene recaps & chapters as described in Section 3.3.5. Once the alignments were established with preprocessed text, they were then reverted to the original texts using index references to recreate the aligned text from the original corpus. This step is important because capitalization and punctuation can strongly effect NER tools.

An initial test showed that not all characters were recognized as people but some inanimate entities were (e.g., "Moat Cailin", "Winterfell", etc.). This could be improved by training a new model on *Game of Thrones* data but that is outside the scope of this thesis. Another issue with the character recognition is that many characters are referred to in more than one way throughout the television series, books, and scene recaps and it is only sometimes the version that has thus far been used for a character in making the narrative charts (e.g., "the Hound" vs. "Sandor Clegane"). This could be solved by manually making lists for each character of all their names and aliases, but that is also outside the scope of this thesis.

The location extraction was even less successful. We did two tests, one taking all the tokens with a LOC (location) tag and the other taking words that have either a LOC or ORG (organization) tag as suggested in [Lingad et al., 2013]. We then said that the location of a scene was the entity that was most frequently extracted for that scene, as each scene should only be taking place in one location. There were not many locations found just using the LOC tag. More locations were found with LOC and ORG but with considerably more noise as well, as many characters were mistakenly included in this extraction. Table A.5 in the Annex shows these extractions.

Due to the difficulties explained in this section for using entities recognized by either Stanford NER or Spacy in our narrative charts, we instead focus on matching names from a list to find locations and non-speaking characters, as explained in the next section.

### List Matching

The second way we tried to extract the names was by matching the text from the transcripts & scene recaps & chapters alignment to our list of characters gathered from the *Game of Thrones* Wiki. We matched only by first name as forcing the name to match both first and last resulted in many if not most names not being extracted and searching just by last name was too ambiguous. The names we find using this method also need to be transformed before matching the format we use to create our charts so we can find the correct character id to place in the JSON file. This method, while not perfect, generates less noise than the NER method, which makes for a simpler transformation.

We did not include any characters with ambiguity, e.g., when a first name that was found matched two or more characters (e.g., "Jon Arryn" and "Jon Snow"). One way to improve this and reduce ambiguity in order to be able to use these characters found would be to create a network of character relationships like in [Vala et al., 2015]. This would allow us to determine that if an ambiguous character name often appears within a network of other characters and at one time is referred to in a non-ambiguous way, we can expand this reference to all the characters with the ambiguous name who appear within this network.

### 3.4.2 Important Word Extraction

To further enhance the narrative charts and go beyond the original xkcd charts, we put lists of the most important words per chapter, for each chapter aligned in the data used to make the chart. To find these important words we took the TF-IDF scores of each word for each chapter in the first book. This was performed on preprocessed text in exactly the same way as in Section 3.3.5 but only using the chapters, not the scene recaps. The words with the top five scores for each chapter were kept as most important and placed at the bottom of the narrative charts. The most important words for the first five chapters, including their frequency and TF-IDF scores can be found in the Annex in Table A.4.

## 3.5 Conclusion

In this section we used different alignment methods to align various parts of our *Game of Thrones* corpus. We saw how each of our experiments increased or decreased the effectiveness of our narrative chart. We described the process for aligning characters & episodes, transcripts & subtitles, characters & scenes, transcripts & scene recaps, scene recaps & chapters, as well as extraction techniques to make use of these alignments in the charts. We provided initial evaluations for the performance of each of these steps. We saw that to align transcripts & scene recaps, the best method was using tokens as common as features in the Dynamic Time Warping Algorithm. We also found that using Tf-IDF vectors as features in a majority comparison algorithm was effective in aligning chapters & scene recaps, but that we got even better results when performing a triple alignment and including aligned text from the transcripts to the scene recap text before performing the chapters & scene recaps alignment with TF-IDF. We found that information extraction using NER tools introduced extra layers of difficulty and our list matching method worked more simply.

CHAPTER 4

# RESULTS

**Contents**



## 4.1 Introduction

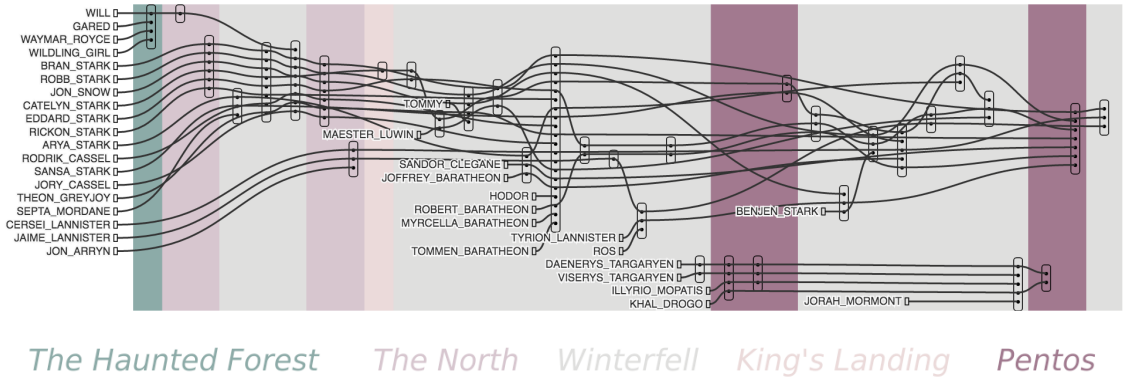
Our goal in this study has been to create narrative charts for the first season of *Game of Thrones* using information gleaned from alignments of our corpora and extractions of various information therein. We have already evaluated the performance of our alignments and extractions, now we present our final narrative charts and try to evaluate their accuracy. We can evaluate the charts' accuracy by comparing them with charts generated using manually extracted information. The locations and characters were manually extracted from watching the first episode of the television series and personal knowledge of the series and all sections of the corpora to obtain as much accuracy as possible. With the exception of the first automatically generated chart which shows the entire first season of the television series, in this section we use Episode 1 to illustrate the capacities of each alignment or extraction in making the narrative chart. The narrative charts present a chronological progression of incremental improvements. Our first chart was produced after the character & episode alignment, and our second chart after the character & scene alignment. After we performed all our transcript & scene recap and scene recap & chapter alignments, we used the methods with the best results to create text outputs from which we then extracted various information. This information, along with the character & scene alignment, appears in our third and fourth charts. The final chart is shown including all the previous improvements and also a list of important words per chapter.

## 4.2 Narrative Charts

We present two types of narrative charts in this section, manual and automatic. The first chart shown is made with manually aligned data. We refer to this chart when showing the progress of our charts made with automatically generated data, as it shows the kind of result we would like to attain automatically. The next section presents our series of progressively improving automatically generated charts.

### 4.2.1 Manual Chart

Figure 4.1 shows our narrative chart crafted using the same library as the automatically generated charts but using entirely manually aligned data. This chart serves as a reference to our automatically generated charts. Scene segmentations were judged using the scene recaps from the *Game of Thrones* Wiki. The episode was watched with these segmentations used as a guide to list all named characters in each scene. The character names were then converted to their ids using the same reference as the automatically aligned characters. Location information was more difficult to judge from watching the episode, so the location information from the annotations section of the corpus was heavily relied upon. An important note is that there can be quite a bit of ambiguity in the location information, as some locations such as "the North" are relative (e.g., when a character is it Winterfell, they can either refer to the North as the land where the currently are, or it could be further away and geographically North of where they are standing) and other locations can be described from different levels of precision, for example "Winterfell" is located in "the North". The chapters listed are those from fan information and the important words were calculated with TF-IDF as they are in the automatically generated charts, as manually finding all the important words in a chapter would be extraordinarily time consuming.




**Chapter 1** gared, waymar, royce, sable, lordling
**Chapter 2** bran, pup, greyjoy, pups, jon
**Chapter 3** ned, catelyn, godswood, weirwood, faith
**Chapter 4** illyrio, dany, magiste, viserys, khal
**Chapter 5** ned, lantern, crypt, robert, tombs
**Chapter 6** jon, ghost, benjen, dwarfs, dwarf
**Chapter 7** ned, catelyn, luwin, lens, maester
**Chapter 9** bran, gargoyle, ledge, pony, roof
**Chapter 12** dany, illy, drogo, khal, dothraki

Figure 4.1: Narrative Chart for Episode 1 Using Manually Extracted Characters and Locations

### 4.2.2 Automatically Generated Charts

Our first automatically generated chart is shown in Figure 4.2. It is the resulting narrative chart for the alignment of characters & episodes using the Tapestry library. We were initially intending to use this alignment as the base information to create our narrative chart but the results were unsatisfactory and the chart is very unclear. Unlike the book, where each chapter focuses on one character, the series splits up

the points of view to show at least a small part focusing on each of the main characters in most episodes. This resulted in an illegible chart with most lines passing through each episode point. The 'start' and 'duration' times here were chosen for graph readability and do not reflect the actual episode run-times.

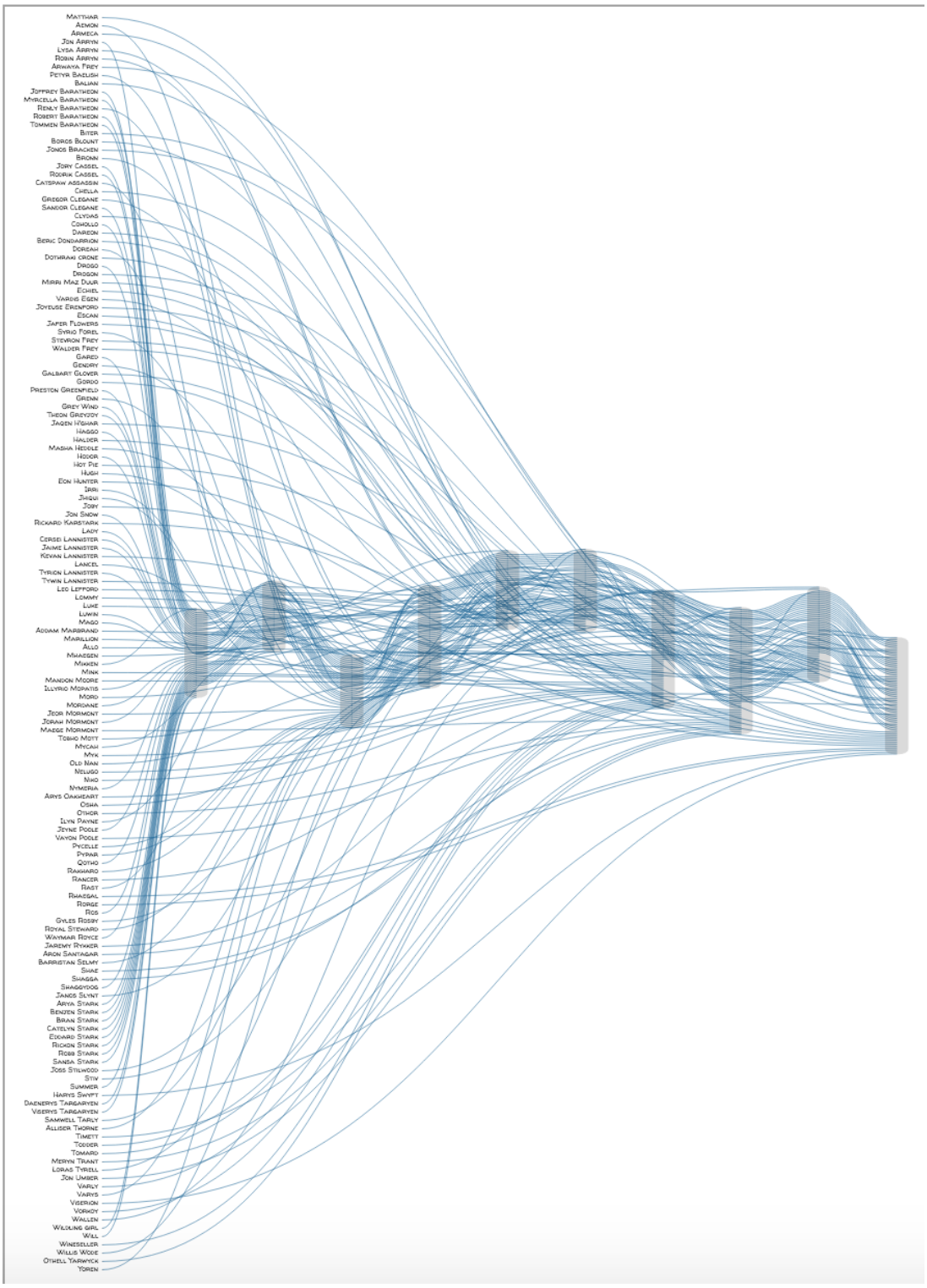



Figure 4.2: Characters & Episodes Alignment

Figure 4.3 shows the result of the character & scene alignment. From this point on in the experiments, the second charting library was used to create the narrative charts. It provides additional information by showing approximately at what point each character enters into the episode instead of presenting them all at once at the beginning as if all the characters are actually present in the first scene. Also, it appears more legible and the aesthetic is preferable as it is closer to that of the xkcd charts.

When we compare the character alignment in this chart to the character alignment in the manually aligned chart we can see that is looks a bit sparse. There are fewer characters contained in the chart and less vertical complexity. This is due to the fact that the scenes in this chart only contain characters with speaking roles and ignore other characters who are present but silent. The chart contains some other errors, such as Robert Baratheon being present with the Targaryens in scene 16, due to quick scene cuts causing the scene segmentation data to be slightly off.

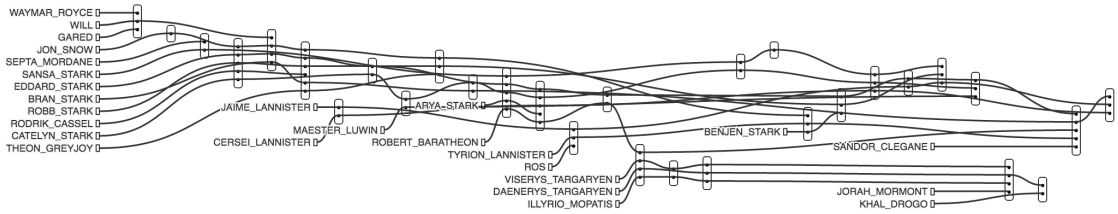



Figure 4.3: Narrative Chart for Episode 1 Using Character and Scene Alignments

To help resolve the missing characters issue, we extracted characters from scene recaps by list matching and added their character ids to the lists of character ids we already had for each scene. As we can see in the result in Figure 4.4, we found many of the missing characters for each scene but also added some characters to scenes where they were just being talked about. Nevertheless, this chart is much closer in appearance to the manually aligned chart in section 4.2.1.

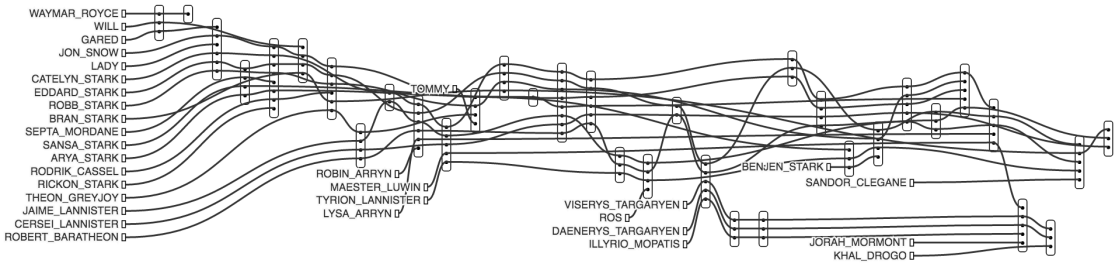



Figure 4.4: Narrative Chart for Episode 1 Including Characters Found in Scene Recaps

| | Precision | Recall | F-measure |
|---|---|---|---|
| before extraction | 0.921 | 0.681 | 0.711 |
| after extraction | 0.722 | 0.838 | 0.735 |

Table 4.1: Character Extraction Evaluation - Episode 1

We can see in Table 4.1 a comparison evaluation of the characters found per scene using speaking characters (before extraction) and the characters once we added in the

extracted information (after extraction), using a manual alignment as a reference. After adding in extracted information our precision decreased as was to be expected since we since our method also adds characters who are spoken about but not present as well as those who are present. Our recall increased, showing that we found many of the characters with no lines in each scene. We compare the F-measures to find that they are similar, yet show a slight overall improvement in the accuracy of our narrative chart when using our extraction methods to add characters to scenes.

In our fourth chart, seen in Figure 4.5, we took our previous chart with the character extractions added, and added another level of information: location. This information was extracted from the scene recap & transcript & chapter alignments described in section 3.3.5. The location information is shown in the bars of color added on top of each scene. A key containing the location names can be found under the chart. For scenes in which no location was found, no color was added.

We can see in a comparison of the manually aligned chart that the location information is much more varied than it should be. As briefly discussed in Section 4.2.1, there are some location ambiguities causing errors such as "the North", as it can refer either to a large area of land north of the Twins (a location in the middle of one of the *Game of Throne*'s world's continents), ruled by the Warden of the North or a relatively northern location to where the speaking character is. As such, "the North" could simply be "the North" or it could be confused with either Winterfell, the Wall, the Haunted Forrest, etc.

We performed an evaluation of the location extraction for episode 1 using the locations automatically extracted and the locations found manually. As we can see in Table 4.2, the recall is higher than the precision.

| | Precision | Recall | F-measure |
|---|---|---|---|
| Episode 1 | 0.444 | 0.667 | 0.533 |

Table 4.2: Location Extraction Evaluation - Episode 1

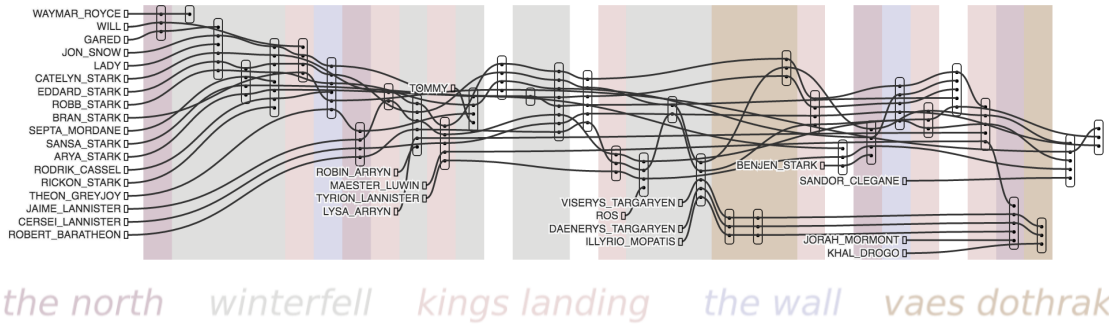


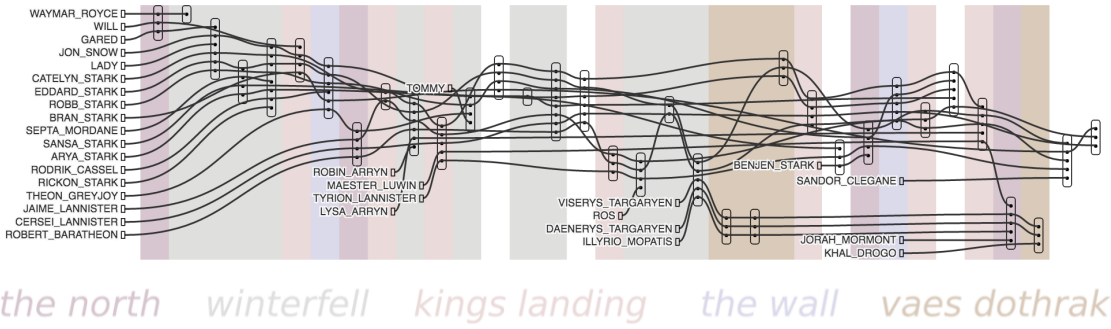

Figure 4.5: Narrative Chart for Episode 1 Including Characters Found in Scene Recaps and Locations Found in Scene Recap & Transcript & Chapter Alignments

Figure 4.6 shows our final narrative chart. It includes both the character and location extracted information described above and also important words per chapter. The important words were extracted using TF-IDF as described in section 3.4.2. The chapters listed as those that were found to be aligned with the scenes in section 3.3.5.

While the important words within the chapters are reliably accurate, the chapters chosen were based on an alignment with a 0.563 F-measure. Thanks to a fan's episode

& chapter alignment[1], we can see that the chapters should have been 1-7, 9, and 12.

Despite the slight lack in precision, we can see overall that the effect of the automatically generated chart and the manually aligned chart is very similar.




**Chapter 1** gared, waymar, royce, sable, lordling
**Chapter 2** bran, pup, greyjoy, pups, jon
**Chapter 4** illyrio, dany, magiste, viserys, khal
**Chapter 5** ned, lantern, crypt, robert, tombs
**Chapter 6** jon, ghost, benjen, dwarfs, dwarf
**Chapter 7** ned, catelyn, luwin, lens, maester
**Chapter 8** arya, septa, mordane, stitches, needlework
**Chapter 10** tyrion, jaime, tommen, chayle, differ
**Chapter 12** dany, illy, drogo, khal, dothraki
**Chapter 23** arya, needle, septa, mordane, sansa
**Chapter 38** bran, robb, stiv, theon, osha
**Chapter 52** sansa, boros, joffrey, jeynes, jeyne

Figure 4.6: Narrative Chart for Episode 1 Including Characters Found in Scene Recaps, Locations Found in Scene Recap & Transcript & Chapter Alignments, and Important Words per Chapter

## 4.3 Conclusion

We can see that the charts we created after each alignment or extraction task improved incrementally with each new chart. We compared the charts to a manually aligned chart and saw the final chart generated shares many similarities with the manual chart. However, the final chart generated is far from being completely accurate. While the character extraction helped the overall accuracy of the graph, the location extraction has more room for improvement. We will discuss some of the possible sources of error in the next section.

[1] http://joeltronics.github.io/got-book-show/bookshow.html

CHAPTER 5

# DISCUSSION

**Contents**



## 5.1 Introduction

In this chapter we take a global view of this study and we discuss some of the successes we experienced as well as some of the limitations and potential sources of error in our experiments. We also discuss what issues could arise in future studies on alignment with the purpose of creating a narrative chart and propose a potential solution. Overall, we succeeded in our mission to create a narrative chart based off *Game of Thrones*. However, instead of creating one chart based off the entire first book and season of the television series, we reduced the complexity and increased legibility by creating one chart for each episode. While the narrative charts were successfully made, the alignments and extractions used to create them contain some room for improvement.

## 5.2 Summary of the Results

In terms of creating a visual representation of our *Game of Thrones* corpus, this study was a success. As planned, we created a narrative chart resembling the charts from xkcd. The character interactions are shown over a period of time, helping to clarify who is where and when for each of the people in the series.

We successfully identified many characters with non-speaking roles in a scene using only text clues. We also identified locations from the text and added them in a color-coded fashion to improve our narrative chart. Lastly, we went one step further than the xkcd charts and added lists of important words per chapter to help convey the most important characters and concepts in an episode at a glance.

This was accomplished by aligning sections of the corpus. This alignment became more difficult the more different the texts we were trying to align were. We saw that sometimes the simplest method can give us the best results. This was seen especially in the transcripts & scene recaps alignment (Section 3.3.4), where the measure of

number of tokens in common outperformed other more complex methods like using word embeddings or part-of-speech filters.

We also saw in the chapters & scene recaps alignment experiments (Section 3.3.5), that combining alignment methods and corpora can yield interesting and more accurate results. We did this by taking the best alignment methods thus far for both transcripts & scene recaps (number of tokens in common) and chapters & scene recaps (TF-IDF scores) and creating a two step process to align all three transcripts & scene recaps & chapters. The result of this triple alignment helped boost the accuracy of the chapters & scene recaps alignment. Further research could attempt to improve the accuracy of the alignments in this study by exploring alignment method combinations and multiple alignment for sections of the corpora.

## 5.3 Limitations

There were many potential sources for error in our alignments and extractions. Some of these errors were within but some were outside of our control. We determined the main reason for a less than perfect alignment between the transcripts & the subtitles was an inclusion of additional data from one or the other documents. This inclusion usually takes the form of interjections such as “uh” or “hmm” or sound effects included from the subtitles such as “(Horses galloping)” or “(Dothraki speaking)”. As mentioned in section 3.3.2, the sound effects were kept for possible improvement in a subtitle & chapter or subtitle & scene recap alignment. The scene recaps & transcripts alignment presented more difficulty than the transcripts & the subtitles alignment due to the greater difference in the two documents’ texts. The main difficulties were as follows:

- Characters talking about other characters who appear in adjacent scenes
- Scenes that have little to no dialogue in them
- Similar themes in multiple scenes
- Longer sequences of very short transcripts
- Character overlap in adjacent scenes
- Transitions between scenes (moving window method (see Section 3.3.4))

These issues all blurred the difference between the scenes or transcripts. When for example a sequence of scene recaps share the majority of their defining words, the transcripts that would normally be aligned with those scenes could more easily be aligned with an incorrect scene, because the defining words are used in both scenes instead of just the correct scene.

In our final alignment, that of scene recaps & chapters, our two sources of text were even more disparate. The two source documents not only no longer followed the same chronological order, but each of them also contained a significant amount of text not contained in the other. We tried to solve this issue by creating a null alignment, where a scene that does not exist in the document with which we are attempting an alignment is aligned instead with a placeholder to show that it does not exist, rather than simply creating an incorrect alignment. This technique of dealing with these departures from the storyline is seen in [Tapaswi et al., 2015] but

this was not as simple in our case using only text cues. When the two documents no longer follow the same storyline, they still contain similar vocabulary such as character names, places and common nouns like “sword” and “horse” that appear in many scenes and chapters. For example, this can easily be seen in the final narrative chart when looking at the locations. Many of the “Vaes Dothrak” scenes should be labeled “Pentos”. The scenes set in both the Pentos and Vaes Dothrak locations come from chapters about Daenerys Targaryen. All of the principle characters and themes in these scenes are the same which caused Daenerys-central scenes to be aligned with the incorrect Daenerys-central chapters.

## 5.4 Future Studies

One issue that could arise in the continuation of this study is an issue of data availability or formatting. Due to its popularity, *Game of Thrones* has a wide fan base who collect data on and write about the series’ content. As we saw in the Related Work section, other researchers have also made use of this wealth of comparable data and produced their own additions such as the scene segmentations from [Tapaswi et al., 2015] or the location annotations provided by LIMSI. We were able to perform so many experiments because the amount of information available is so vast. Repeating the methods listed in this study to generalize to other series or events could prove difficult if the series generalized to does not have as many resources.

However, much of this difficulty could be resolved using other automatic methods. If manual transcripts are not available, they can be constructed using speech-to-text methods for the dialogue lines and using speaker identification methods, such as the one described in [Roy et al., 2015], to add speaker names to each of the lines of dialogues. Face detection, as used in [Tapaswi et al., 2015], is an automatic and accurate way to place characters in scenes and episodes. Finally, automatic text summarization methods, like the ones elaborated on in [Mani and Maybury, 1999], could be used to create summaries out of longer texts, to add to and enhance the narrative chart.

## 5.5 Conclusion

In this chapter we discussed the performance and limitations of our methods and study. We also discussed an issue that anyone continuing this work in another domain might face, and provided a possible solution to the problem.

# Conclusion

In this study we discussed and experimented with the problem of text-based alignment methods. Our objective was to clarify and illustrate the *Game of Thrones* character interactions by making a visual representation, similar to the narrative charts from the webcomic xkcd, of the first season of the series. To accomplish this task we gathered comparable corpora centered on our subject, and applied a number of alignment techniques to group various aspects of it. We aligned characters & episodes, characters & scenes, subtitles & transcripts, transcripts & scenes, and scenes & chapters. We evaluated the accuracy of our alignments and compared our automatically generated results with manually generated ones. We extracted information from our alignments to create enhanced narrative charts.

Our final narrative chart displays characters by scene and location. This successfully helps to clarify character interactions within the series by showing what characters are with what other characters at a given point in time, and also where they are. Our chart goes one step further and includes lists of important words per chapter for each chapter that was aligned with the episode. This continues to help clarify the *Game of Thrones* storyline in that we can see at a glance the most important characters and concepts in an episode.

This type of narrative chart is useful in more ways than simply showing enhanced *Game of Thrones* timelines. With slight modifications, it could be used for other television series or even in real-world events using news media of differing types or other factual information. A fine example of one such graph, generated given manually assembled data but using the same chart library as is used in this study, is a visual representation of the ICAC scandal.[1] The data to create this graph could have, instead of being manually put together, been gathered on news websites, aligned from transcripts to articles or articles to other articles treating the same topic, then parsed to form the graph using the methods described in this paper.

Another hypothetical use for this type of narrative chart is to apply alignment methods to information from the current US elections such as transcripts from debates, summaries of debates with opinions, etc., to create a timeline graphic. This would potentially help American citizens make a more informed choice when voting by better understanding the timeline of events, opinions and candidate's policies during the lead up to the elections.

These ideas are just some of the myriad uses text alignment and text extraction could be applied to. Our main issue in this study was attaining satisfactory accuracy. Our final chart is close to our manually aligned ground truth, but it still contains room for significant improvement. With the inclusion of other forms of media such as audio or video, alignment accuracy could increase, making narrative chart generation more precise with even more elaborate graphs produced.

[1] http://www.abc.net.au/news/2014-08-21/untangling-the-web-how-the-icac-scandal-unfolded/5686346

# BIBLIOGRAPHY


[Barzilay and Elhadad, 2003] Barzilay, R. and Elhadad, N. (2003). Sentence Alignment for Monolingual Comparable Corpora. *EMNLP '03 Proceedings of the 2003 conference on Empirical methods in natural language processing*. – Cited page 16.

[Gibbon, 2002] Gibbon, D. (2002). Generating Hypermedia Documents from Transcriptions of Television Programs Using Parallel Text Alignment. *IEEE Eighth International Workshop on Research Issues in Data Engineering (RIDE)*. – Cited pages 15 and 16.

[Jones, 1972] Jones, K. S. (1972). "A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21. – Cited page 29.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *CoRR*. – Cited pages 28 and 39.

[Leidner et al., 2003] Leidner, J. L., Sinclair, G., and Webber, B. (2003). Grounding spatial named entities for information extraction and question answering. *HLT-NAACL-GEOREF '03 Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references - Volume 1*, pages 31–38. – Cited page 17.

[Levy and Goldberg, 2014] Levy, O. and Goldberg, Y. (2014). Linguistic Regularities in Sparse and Explicit Word Representations. *CoNLL 2014*. – Cited page 28.

[Lingad et al., 2013] Lingad, J., Karimi, S., and Yin, J. (2013). Location Extraction From Disaster-Related Microblogs. *Proceedings of the 22nd international conference on World Wide Web companion International World Wide Web Conferences Steering Committee 2013*, pages 1017–1020. – Cited pages 17 and 45.

[Mair and Hundt, 2000] Mair, C. and Hundt, M. (2000). *Corpus Linguistics and Linguistic Theory*. Rodopi. – Cited page 27.

[Mani and Maybury, 1999] Mani, I. and Maybury, M. T. (1999). *Advances in Automatic Text Summarization*. The MIT Press. – Cited page 55.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546. – Cited page 28.

[Ratinov and Roth, 2009] Ratinov, L. and Roth, D. (2009). Design Challenges and Misconceptions in Named Entity Recognition. *CoNLL '09 Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. – Cited page 30.

[Roy et al., 2015] Roy, A., Bredin, H., Hartmann, W., Le, V. B., Barras, C., and Gauvain, J.-L. (2015). Lexical Speaker Identification in TV Shows. *Multimedia Tools and Applications*. – Cited pages 17 and 55.

[Roy et al., 2014] Roy, A., Guinaudeau, C., Bredin, H., and Barras, C. (2014). TVD: a reproducible and multiply aligned TV series dataset. *9th Language Resources and Evaluation Conference LREC*. – Cited pages 15, 16 and 33.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). "Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1). – Cited page 25.

[Sanchez-Perez et al., 2014] Sanchez-Perez, M. A., Sidorov, G., and Gelbukh, A. F. (2014). A Winning Approach to Text Alignment for Text Reuse Detection at PAN 2014. *CLEF*. – Cited page 16.

[Santorini, 1990] Santorini, B. (1990). "Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision). *Technical Reports (CIS)*. – Cited page 28.

[Scheider and Purves, 2013] Scheider, S. and Purves, R. (2013). Semantic place localization from narratives. *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place 2013*, page 16. – Cited page 17.

[Tapaswi et al., 2014] Tapaswi, M., Bauml, M., and Stiefelhagen, R. (2014). StoryGraphs: Visualizing Character Interactions as a Timeline. *Conference on Computer Vision and Pattern Recognition*. – Cited page 32.

[Tapaswi et al., 2015] Tapaswi, M., Bauml, M., and Stiefelhagen, R. (2015). Book2Movie: Aligning Video scenes with Book chapters. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1827–1835. – Cited pages 15, 16, 21, 32, 54 and 55.

[Vala et al., 2015] Vala, H., Jurgens, D., Piper, A., and Ruths, D. (2015). Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 769–774. – Cited pages 17 and 45.


# ANNEX

| | Method #1 Tokens | Method #2 Tokens | Method #3 Tokens | Method #2' Lemmas | Method #3' Lemmas |
|---|---|---|---|---|---|
| Episode 1 | 0.006 | 0.854 | 0.864 | 0.867 | 0.851 |
| Episode 2 | 0.003 | 0.679 | 0.754 | 0.716 | 0.689 |
| Episode 3 | 0.014 | 0.893 | 0.825 | 0.839 | 0.862 |
| Episode 4 | 0.009 | 0.920 | 0.868 | 0.905 | 0.863 |
| Episode 5 | 0.009 | 0.806 | 0.780 | 0.806 | 0.764 |
| Average | 0.008 | 0.830 | 0.818 | 0.826 | 0.805 |

Table A.1: Transcripts & Scene Recaps Alignment Evaluation - Comparison of all Methods - Part I

| | Method #4 Word2vec | Method #5 Word2vec | Method #6 Doc2vec | Method #7 Part-of-speech | Method #8 Moving Window |
|---|---|---|---|---|---|
| Episode 1 | 0.589 | 0.851 | 0.658 | 0.477 | 0.852 |
| Episode 2 | 0.490 | 0.699 | 0.765 | 0.048 | 0.672 |
| Episode 3 | 0.593 | 0.808 | 0.819 | 0.048 | 0.805 |
| Episode 4 | 0.778 | 0.823 | 0.800 | 0.021 | 0.853 |
| Episode 5 | 0.198 | 0.757 | 0.801 | 0.012 | 0.774 |
| Average | 0.529 | 0.787 | 0.769 | 0.121 | 0.791 |

Table A.2: Transcripts & Scene Recaps Alignment Evaluation - Comparison of all Methods - Part II

| | Method A Tokens | Method B Lemmas | Method C TF-IDF | Method D TF-IDF + T |
|---|---|---|---|---|
| Episode 1 | 0.338 | 0.225 | 0.479 | 0.563 |
| Episode 2 | 0.355 | 0.355 | 0.452 | 0.484 |
| Episode 3 | 0.333 | 0.481 | 0.593 | 0.519 |
| Episode 4 | 0.615 | 0.462 | 0.5 | 0.615 |
| Episode 5 | 0.308 | 0.308 | 0.308 | 0.385 |
| Average | 0.389 | 0.366 | 0.466 | 0.513 |

Table A.3: Chapters & Scene Recaps Alignment Evaluation - Comparison of all Methods

| Chapter | Word | Frequency | TF-IDF score |
|---|---|---|---|
| Chapter 1 | gared | 30 | 0.02504 |
| | waymar | 22 | 0.01671 |
| | royce | 22 | 0.01089 |
| | sable | 6 | 0.00564 |
| | lordling | 7 | 0.00364 |
| Chapter 2 | bran | 40 | 0.00793 |
| | pup | 10 | 0.00729 |
| | greyjoy | 12 | 0.00554 |
| | pups | 8 | 0.00552 |
| | jon | 30 | 0.00523 |
| Chapter 3 | ned | 17 | 0.00655 |
| | catelyn | 15 | 0.00624 |
| | godswood | 7 | 0.00495 |
| | weirwood | 4 | 0.00386 |
| | faith | 3 | 0.00322 |
| Chapter 4 | illyrio | 30 | 0.01599 |
| | dany | 28 | 0.01278 |
| | magister | 16 | 0.00905 |
| | viserys | 19 | 0.00757 |
| | khal | 18 | 0.00717 |
| Chapter 5 | ned | 56 | 0.01177 |
| | lantern | 5 | 0.00423 |
| | crypt | 5 | 0.00423 |
| | robert | 32 | 0.00338 |
| | tombs | 4 | 0.00308 |

Table A.4: Top Words per Chapter - TF-IDF - First 5 Chapters

| scene | LOC & ORG | LOC | List Matching | Manual |
|---|---|---|---|---|
| 1 | Ser Waymar Royce | Haunted Forest | the north | Haunted Forest |
| 2 | N/A | N/A | winterfell | The North |
| 3 | Winterfell | N/A | winterfell | The North |
| 4 | N/A | N/A | winterfell | Winterfell |
| 5 | Septa Mordane | Donnis | winterfell | Winterfell |
| 6 | Winterfell | Eddard | kings landing | Winterfell |
| 7 | Eddard | N/A | the wall | The North |
| 8 | North | North | the north | The North |
| 9 | Kings Landing | Casterly Rock | kings landing | King's Landing |
| 10 | Winterfell | N/A | winterfell | Winterfell |
| 11 | Kings Landing | N/A | kings landing | Winterfell |
| 12 | Winterfell | N/A | winterfell | Winterfell |
| 13 | N/A | N/A | N/A | Winterfell |
| 14 | N/A | N/A | winterfell | Winterfell |
| 15 | Winterfell | N/A | winterfell | Winterfell |
| 16 | Queen Cersei | N/A | N/A | Winterfell |
| 17 | House | East | kings landing | Winterfell |
| 18 | N/A | N/A | winterfell | Winterfell |
| 19 | Tyrion | N/A | winterfell | Winterfell |
| 20 | Winterfell | N/A | winterfell | Winterfell |
| 21 | Usurper | the Jade Sea | vaes dothrak | Pentos |
| 22 | Dothraki | N/A | vaes dothrak | Pentos |
| 23 | Viserys | the Jade Sea | vaes dothrak | Pentos |
| 24 | Grand Maester Pycelle | N/A | kings landing | Winterfell |
| 25 | N/A | N/A | N/A | Winterfell |
| 26 | Night | North | the north | Winterfell |
| 27 | N/A | N/A | the wall | Winterfell |
| 28 | Grand Maester Pycelle | N/A | kings landing | Winterfell |
| 29 | Eddard | N/A | N/A | Winterfell |
| 30 | Sansa | N/A | kings landing | Winterfell |
| 31 | House | Vale | the north | Winterfell |
| 32 | Dothraki | N/A | vaes dothrak | Pentos |
| 33 | Drogo | N/A | N/A | Pentos |
| 34 | N/A | N/A | N/A | Winterfell |
| 35 | Winterfell | N/A | winterfell | Winterfell |

Table A.5: Location Extraction by NER, List Matching, and Manual Extraction - Episode 1

| scene | characters |
|---|---|
| 1 | Ser Waymar Royce, Mallisters, Wills, Ser Waymar, Gared, Will, Ser Waymar, Mormont, Castle Black, Robert, Maester Aemon, Royce |
| 2 | House Stark, Stark |
| 3 | N/A |
| 4 | House Tully, Lord Eddard Stark, Robb Stark, Catelyn, Lady Catelyn, Bran Stark, Jon Snow |
| 5 | Jeyne, Eddard Starks, Theon, Poor Jon, Stark, Beth Cassel, Rodrik, Tullys, Baratheon, Prince Tommen, Robb, Theon Greyjoy, Jeyne Poole, Myrcella, Joffrey, Rhoyne, Beth, Ser Rodrik, Joff, Jon, Lew, Prince Joff, Mother, Winterfell, Arya, Bran, Ser Rodrik Cassel, Nymeria, Ser Rodrik, Sansa, Septa Mordane, Tommen, Lannister, Lady Catelyn, Rickon, Wed Tully |
| 6 | Mance Rayder, Maester Luwin, Moat Cailin, Theon, Greyjoy, Father, Hullen, Quent, Rodrik, Starks, Wayn, Lannisters, Stiv, Lord Eddard, Robb, Joseth, Wyl, Osha, Theon Greyjoy, Hodor, Hal Mollen, Benjen Stark, Lannister, Alyn, Hali, Rickon, Jon, Maester, Wallen, Hallis Mollen, Brandon Stark, Grey Wind, Stark, Mother, Wolves, Arya, Bran, Jory, Ser Rodrik Cassel, Benjen, Summer, Heward, Eddard, Dancer, Robb Stark, Luwin, Hack, Jory Cassel, Maester Pycelle, Jon Snow |
| 7 | Eddard, Starks, Bran, Jon |
| 8 | Mance Rayder, Theon, Robert, Father, Snow, Stark, Rodrik, Starks, Robb, Theon Greyjoy, House Stark, Warden, Ser Rodrik, Greyjoy, Jon, Nan, Lord Eddard Stark, Hullen, Winterfell, Bran, Jory, Wall, Rickon, Eddard, Lord Stark, Jory Cassel, Jon Snow, Harwin |
| 9 | Arryn, Robert, Cersei, Hand, Jaime, Robert Baratheon |
| 10 | Catelyn |

Table A.6: Character Extraction by NER - Episode 1 - First 10 Scenes