On the effects of using speech transcripts and subtitles to detect topic shifts in news broadcasts

Abstract

In this research, topic segmentation in texts (a.k.a. text segmentation) is used as a proxy for topic segmentation in videos. The main application is automatically providing a topic transition structure for videos, because it is difficult to quickly scan them and figure out where a new subject starts. Topic models are used to figure out the topic transition positions. The available data for this research is provided by the Netherlands Institute for Sound and Vision and consists of 25,600 transcripts and subtitles of the same Dutch news broadcasts.

The research questions whether it is better to use automatic speech recognition transcripts or subtitles when segmenting a video based on topics.The subtitles and speech transcripts were compared for the same news broadcasts and both qualitative and quantitative differences between them were found. However, no significant difference was found between the performance of the text segmentation algorithm using subtitles and speech transcripts. The research presents the challenges and benefits of the developed text segmentation algorithm. The research can give insight into the realizability of the application of text segmentation to help structure videos, which can become a starting point for future research.

Utrecht University

Name: Sahra Mohamed

Date: 26-06-2020

First supervisor: Dr. Dong Nguyen
Second supervisor: Dr. Meaghan Fowlie

Programme: BSc Artificial Intelligence at Utrecht University
Credits: 15 ECTS

Contents

Introduction

This thesis has chapters, sections and sub sections. They are - at least I hope so - coherent. They hint at topic shifts and new phases in the thesis. Chapter and section headings are visual cues that provide a way of naturally scanning the text and quickly finding the relevant parts. In videos, similar cues are not always present. It is difficult to catch auditory cues because it is a matter of luck that you hear a transition word. And visual cues, for example a single frame denoting a new subject start are not always present. When it comes to videos it is much more difficult to scan and get a sense of whether it contains parts worth watching. This is a problem at the Netherlands Institute for Sound and Vision (S&V). They have a large archive of millions of Dutch TV and radio programs.

In order to provide a similar structure for analysing videos it is useful to experiment with ways of automatically sensing the different phases and topics in a video. At S&V there has been a project in which automatic speech recognition was applied to parts of their archive. In a broader sense automatic speech recognition (ASR) has become prevalent over the years. This provides an opportunity to think about the use that different video texts can have in detecting topic shifts in videos. This is why I will use **topic segmentation in texts** (a.k.a. text segmentation) as a proxy for topic segmentation in videos. ASR transcripts could serve as the input for text segmentation, to provide videos with more structure. Another kind of video text that I will look at are subtitles. Both the transcripts and subtitles that will be used in this research have been provided by S&V.

The objective of text segmentation is to find out where topic shifts occur in a given text. Early research in text segmentation attempted to recognize topic shifts by using patterns of lexical co-occurrence. A famous successful example is the TextTiling algorithm, which assumes topic boundaries occur at points in a document where large shifts in vocabulary occur (Hearst, M. A. (1993, 1994)). Other text segmentation research approached the problem from a different angle. For example, Passonneau, R. J., & Litman, D. J. (1993), who used machine learning to detect cue-words to find topic boundaries.

Researchers have explored various applications of text segmentations ranging from passage retrieval (Salton, G., Allan, J., & Buckley, C. (1993)) to automatic genre detection (Karlgren, J. (1996)). And from genre detection to automated text summarization (Barzilay, R., & Elhadad, M. (1999)).

What has not extensively been studied is the kind of input data that makes text segmentation work well. And on the flip side the kind that makes it work poorly. For example, one can wonder about the suitability of ASR transcripts versus subtitles for text segmentation. That is precisely the main question in this research: *Is it better to use automatic speech recognition transcripts or subtitles when segmenting a video based on topics?*

The specific question of which input, speech transcripts or subtitles, works better for text segmentation is practically useful for archiving purposes, when the possibility for making that choice arises. But besides this, it is an important question because the reasons behind why it does or does not make a difference, can help give us a better understanding of the workings of text segmentation. Furthermore it can give insight into the realizability of the application of text segmentation to help structure videos.

The following **Section 1** is about the background information that is important for this research; mainly the techniques and evaluation metrics that will be used. In **Section 2** I look at the qualitative and quantitative characteristics of the research data. **Section 3** is about the methodology and experimental setup for this research. In **Section 4** the results of that experiment are presented and discussed. **Section 5** focuses on the benefits and limitations of the text segmentation algorithm and proposes ideas for future research. In **Section 6**, the research is concluded.

1. Background information

1.1 Topic modelling

1.1.1 Why use topic modelling?

Detecting topic shifts in text is easy enough to do for humans. We know the semantics of words and sentences, we can sense when the meaning of it is changing too much. We know that describing a soccer match might entail using words like ball, line, point, goal, team. And a description of a fire might use words like match, smoke, oxygen, gas, flames. Once enough words start appearing that we associate with another topic, we assume another one has started.

When thinking about automating that process of detecting topic shifts, there is the option of trying to replicate that process. This would mean an algorithm would have to learn an equivalent to the experiential knowledge that humans have about what topics and topic shifts look like. Teaching an algorithm what a topic is, would then entail training it to associate words with labeled topics; a form of supervised learning. This form of learning has a downside. Supervised learning with all the required annotation and labeling can be cumbersome. Furthermore, in the future other themes and topics might come into existence. If supervised learning would be used, new categories and labels would have to be made up to deal with those novel cases.

Another option for extracting topics is unsupervised learning by building a topic model. This doesn't require labels. With the potential applications of text segmentation in mind and especially considering the nature of the available data in this research this form of learning is preferable. The available data consists of news broadcasts, of which the content, not unlike this world is constantly changing. Unsupervised learning would be less cumbersome, but it is fair to mention this approach also has a downside. As opposed to supervised learning, it is harder to evaluate the answers because no labels are available beforehand. Hence the evaluation process requires more human intervention.

This view of unsupervised learning - of teaching algorithms what topics are without explicitly telling them - is why topic modelling will be used in this research. Topic modeling algorithms do not require any prior labeling of the documents (Blei, D. M. (2012)). Therefore, there is much to be said for using them.

1.1.2 What are topic models?

'Topic models' are probabilistic models for finding patterns of words in document collections. (Alghamdi, R., & Alfalqi, K. (2015)). The first half of a topic model is tasked with connecting topics to a probability distribution over all words. Here the definition of a topic is precisely that: a probability distribution over all words. To illustrate, here are two example topics:
Topic 1: [('processing' : 0.1), ('data' : 0.08), ('computer' : 0.06), …]
Topic 2: [(boat : 0.23), (sea : 0.05), (moon : 0.2) …]

A topic model gives you collections of words that make sense together (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)). Topic 1 described on the previous page, could be labeled as ‘computer science’, though only the three words with the highest probability in the distribution are shown. The second topic is already a little harder to describe using a similar label. For all we know, it could come out of an article about nature, an outdoor activity or a fiction story.

It is important to note that a topic model does not provide a label like ‘computer science’ to describe the probability distribution over the words. Those labels are entirely removed from the raw distribution over words (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)). Though such a label intuitively might feel closer to the notion of a topic we are used to, this is not the definition of a topic here.

The first half of a topic model defines the topics, while the second half links these topics to individual documents (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)). Linking the topics to documents is also done in the form of a probability distribution: a distribution over topics describing the proportion that a document exhibits each topic. A distribution over topics for two documents might like look this:
Distribution over topics for doc1: [(Topic 1: 0.2), (Topic 2: 0.8)].
Distribution over topics for doc2: [(Topic 1: 0.7), (Topic 2: 0.3)].
This means the second document is more likely concerned with a topic that could be described as ‘computer science’.

1.1.3 Latent Dirichlet Allocation (LDA)

There are different kinds of topic modelling. Here, I will only discuss the kind of topic modelling that I will use in this research, namely Latent Dirichlet Allocation (LDA) (Blei, D., Ng, A., Jordan, M. (2003). Not only is this the most used kind of topic modelling for its intuitiveness and simplicity, it is also widely extended by researchers and being used as a foundation for building newer models that tailor LDA to specific use cases. (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)).

Latent Dirichlet Allocation is a generative probabilistic model. It assumes that documents are generated in a certain way and uses the idea of this imaginary random generative process to guess the topics that might have created the documents (Blei, D. M. (2012)).

The generative process

Before talking about the way LDA infers topics I will shortly expand on the generative process it assumes. Given a number of topics, each document in the document collection shares this set of topics, but exhibits them in different proportions (Blei, D. M. (2012)). Given that LDA has a number of topics and each document has a distribution over topics, LDA assumes that documents are generated in the following way:

- First LDA randomly chooses a distribution over topics.
- Then, for each word in the document, LDA randomly chooses a topic from the chosen distribution over topics.
- Lastly LDA randomly chooses a word from the chosen topic. Since a topic is a distribution over all the words in the document, this means a word is chosen based on that distribution.

The problem that is being solved in topic modelling can be described as reversing the generative process (Blei, D. M. (2012)). Since the words in the documents are actually observable and the underlying topics are not.

Inference

The process of inferring the underlying topic structure Y that likely generated the words X in a document collection can be described as finding the posterior probability:
$P(Y \mid X) = P(X, Y) / P(X)$.
In order to determine the numerator $P(X, Y)$ the idea of the generative process can be used. The probability of a topic structure and the words being in a text depends on the assumptions made in the generative process (Blei, D. M. (2012)). The topic structure has to be able to be broken down to a distribution over topics, which in turn has to contain the distributions of words that could have caused the words in the documents to arise.

In order to determine the denominator $P(X)$, the probability of seeing the words in a corpus under any topic model has to be calculated (Blei, D. M. (2012)). Though this can be done in theory by summing over every possible instantiation of the topic structure, this would be too computationally demanding due to the large number of possible topic structures. (Blei, D. M. (2012)).

Rather than calculating the posterior probability, LDA approximates it. There are several algorithms for approximating posterior probability that can be used with LDA, such as Gibbs sampling (Steyvers, M., and Griffiths, T. (2007) and variational inference (Blei, D., Ng, A., Jordan, M. (2003)).

Assumptions of LDA

It is important to note the assumptions LDA makes along the way of inferring the topic structure of a document collection. I will discuss four assumptions.

The first assumption is related to the generative process. LDA makes the ‘bag of words’ assumption there, meaning that it ignores the order of the words in the document (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)). In other words, the generative process relates to the way the words in the document collection can be created and not to creating coherent documents in which the order of the words certainly matter. However since the goal of topic modelling is only

to uncover the semantic structure of the text, the ‘bag of words’ assumption made by LDA is reasonable (Blei, D. M. (2012)). If a text for example, concerns the topic ‘climate’, this would be derivable even if the words were jumbled and shuffled.

The second assumption made by LDA is similar to the ‘bag of words’ assumption. Namely, that the order of the documents does not matter (Blei, D. M. (2012)). This can be seen in the generative process. In the first step, the distribution over topics is chosen randomly. If the order of the documents mattered, this choice would be dependent on earlier picked distributions over topics. This ‘bag of documents’ assumption makes sense if the goal is only to find the topics in a document collection and topic distribution per document. However if the goal is to find out how topics change over time in documents, it is useful to respect the ordering of the documents (Blei, D. M. (2012)).

The third assumption that LDA makes is that the number of topics in a document collection is given in advance (Blei, D. M. (2012)). This assumption is not intuitive because different documents in a document collection might be concerned with a different number of topics. However this is not an issue because of another assumption made by LDA, that was mentioned earlier in the generative process. Namely, that every document exhibits each topic in different proportions. A document that is about a single topic will exhibit all topics, of which all but one have small probabilities. If the document would instead be about ten topics and LDA is told to only look for three, we can imagine the topics will be less precise and vague. This choice, the number of topics given to LDA, can impact the interpretability of the results. Too few topics will often result in broad topics, while too many topics cause them to be uninterpretable and peculiar in specificity (Steyvers, M., and Griffiths, T. (2007).

The fourth and final assumption made by LDA to discuss here is the premise that the concentration of topics over a document does not change from the beginning of the document until the end (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)). This means that LDA does not provide information about for instance, whether two topics are merged together in the first paragraph or three topics are spread out in the middle of the document. LDA ignores the concentration of topics in the documents and essentially assumes it does not matter for figuring out the topic distribution and distribution over topics in a document collection. This means that LDA, at least by itself, will not give us insight into where the next topic starts. This assumption is strongly tied to the bag of words assumptions where the order of the words in a document are ignored. In order to tell whether topics are concentrated in the first section of a document there has to be a ‘section’ to begin with, which doesn’t exist in a jumbled bag of words.

It is these kinds of assumptions made by LDA mentioned here, that can be relaxed or added in order to adapt LDA to specific use cases. In the case of this research the four aforementioned assumptions fit the research in the following ways.

Firstly, the ‘bag of words’ assumption fits the research, because this research is interested in the topic structure of the texts and not generating new documents that are coherent.
Secondly, the ‘bag of documents’ assumption fits the research, because the research is not interested in how topics change over time in a document collection.
Thirdly, the assumption concerning the prefixed number of topics does not necessarily fit the research, since it would certainly be more ideal if LDA could learn the number of topics it needed to get coherent topics by itself. However this is a problem that is beyond the scope of this particular research.
Lastly, the assumption that the concentration of topics over a document is fixed, does not fit the research. However, this concentration of topics, for example whether two topics are closeby in the first paragraph, assumes there is a paragraph with coherent sentences to begin with. Dropping the bag of words assumptions assures a notion of a paragraph with coherent sentences. Dropping that bag of words assumption is not feasible to do because it makes computing the probabilities in LDA a lot more intensive. Therefore this research will make the final assumption as well and deal with connecting the topics to segments in the document in another way.

1.2 Evaluating text segmentations

1.2.1 Limitations of standard evaluation metrics

Precision and recall are standard evaluation measures that are often used to evaluate machine learning algorithms. However, in the context of text segmentation, these measures are problematic because they are not sensitive to near misses (Pevzner, L., & Hearst, M. A. (2002)).

In the context of text segmentation, precision is the percentage of boundaries that are identified by the algorithm to indeed be true boundaries. Recall is the percentage of true boundaries that have been identified by the algorithm.

Figure 1
Example of different text segmentations. Ref denotes the reference segment which is the right segmentation. A-0 en A-1 denote hypothesised segmenations obtained by a text segmentation algorithm. For visual clarity the first segment contains X’s, the second Y’s and the third Z’s. These variables are units of language. In this case these are words. The bitstrings are representations of the text segmentations. A word position in the bitstring has the value 1 if a segmentation follows after that word.

Ref	XXXX | YY | ZZZ
Bitstring Ref	000101000
A-0	XXX | XYYZ | ZZ
Bitstring A-0	001000100
A-1	X | XXXYYZZ | Z |
Bitstring A-1	100000011

Pevzner, L., & Hearst, M. A. (2002) show that precision and recall are not sensitive to near misses with an example using two algorithms A-0 and A-1 and the same reference segmentation as the true segmentation, to compare the segmentations from the algorithms against. Figure 1 shows a similar but slightly adapted case that will also be used in the next sections to explain other evaluation metrics. For now the bitstrings in the figure can be ignored.

Figure 1 shows the following example. Suppose Ref has two segmentations in a stream of words. A-0 segments the text and is off by one word at the otherwise right places for both segmentations. A-1 segments the text and is off by many more words and adds a third false segmentation in the text. Although A-0 is intuitively closer to the solution and should ideally receive a better score than A-1, both would receive a 0 for precision and recall (Pevzner, L., & Hearst, M. A. (2002)).

1.2.2 Pk metric

Theory

Attempting to resolve the problems with precision and recall, Beeferman, D., Berger, A., & Lafferty, J. (1997) introduced a new evaluation metric. This error metric is a probability, specifically: “the probability that two sentences drawn randomly from the corpus are correctly identified as belonging to the same document or not belonging to the same document.”
The intuition behind this is that one segmentation algorithm is better than another if it can better predict whether two sentences belong to the same document (Beeferman, D., Berger, A., & Lafferty, J. (1997)).

Two years later, Beeferman, D., Berger, A., & Lafferty, J. (1999) adopted a simpler version of the metric called the Pk metric. The k in this metric is a variable that is set to half of the average true segment size. I will calculate k in an example later on. This variable k is then used as the length of a moving window (Pevzner, L., & Hearst, M. A. (2002)). This window moves across the stream of words. The error score is increased if there is a segmentation in the window for the hypothesised segment but not in the window of the actual reference segment and vice versa. If the hypothesised break in the text and break in the reference are both present or both absent within the window, the error score does not increase. If an algorithm correctly assigns all boundaries and adds no false boundaries, the received score is a 0 (Beeferman, D., Berger, A., & Lafferty, J. (1997)).

A false negative occurs when a boundary is in the window of a reference segment but missing in the window of the hypothesised segment. A false positive is a case of assigning a boundary in the window of the hypothesised segment that does not exist in the window of the reference segment (Beeferman, D., Berger, A., & Lafferty, J. (1997)).

Example

If we go back to figure 1, we can calculate the Pk metric between Ref and A-0, denoted as Pk(Ref, A-0). The bitstrings in figure 1 are translations of the segmentations. They are used to calculate the error, because different numbers of boundaries make it difficult to compare two segmentations. For example Ref and A-1 have a different number of boundaries, meaning the moving window could not move in sync across both segmentations since A-1 is longer. Therefore bitstrings are used. A word position in the bitstring has the value one if a boundary follows after that word. That is why the bitstring for A-1 starts with a one. The bitstrings are repeated here for ease of reference.
Ref: 000101000
A-0: 001000100
A-1: 100000011

Let us first calculate Pk(Ref, A-0) and then compare it to Pk(Ref, A-1).
The first step is to calculate k. This is set to half of the average true segment size. This average can be found in Ref because that is the true segmentation.

$$k = 0.5 \cdot \frac{len(Ref)}{n_{boundaries}(Ref)}$$

In the equation above, the average true segment size is calculated in the term on the right. The size of Ref is calculated in len(Ref) and that is divided over the number of boundaries in Ref. In this case k is the following:

$$k = 0.5 \cdot (len(Ref)/n_{boundaries}(Ref)) = 0.5 \cdot (9/2) = 0.5 \cdot 4.5 = 2.25$$

The second step is to use k as a moving window to see whether the number of boundaries in that window are in agreement between Ref and A-0. If k is not a round number it will be rounded off, in this case the final k is 2.

The error measure starts at 0. We start at the first position of the bitstring and iterate past the rest of the positions. Every time that within k positions (in this case 2) of the current position the number of boundaries in Ref and A-0 do not agree, the error is increased with 1.

Ref: [00, 00, 01, 10, 01, 10, 00, 00]
A0: [00, 01, 10, 00, 00, 01, 10, 00]
(==): [T, F, T, F, F, T, F, T]

The two position combinations starting from the left for Ref and A-0 are noted in the lists above. Below these lists, these two-position strings are compared. Whenever the number of boundaries (the number of ones) in the two lists do not agree, the entry in the list gets an F for False. We can see that there are four disagreements, hence the error score is 4.

The last step is to normalize the error score.

$$P(Ref, A0) = 4 / (len(Ref) - k + 1) = 4/8 = 0.5$$

Now that we have calculated Pk(Ref, A-0), we can quickly calculate Pk(Ref, A-1) and compare the results.
k is the same as before, because the reference segmentation Ref stays the same.
Ref: [00, 00, 01, 10, 01, 10, 00, 00]
A1: [10, 00, 00, 00, 00, 00, 01, 11]
(==): [F, T, F, F, F, F, F, F]
The error score is 7.
The normalized error score:
$P(Ref, A1) = 7 / (len(Ref) - k + 1) = 7/8 = 0.875$

We can see that the error for A-0 is smaller than for A-1. This makes sense, because A-1 has an extra false boundary and all boundaries are further from the right boundaries in Ref than the boundaries in A-0 are. This means this error metric is closer to taking into account near misses than precision and recall, which would have given equal scores to both A-0 and A-1.

Issues

Although the Pk measure *does* take near misses into account, this method is not perfect either. Pevzner, L., & Hearst, M. A. (2002) show that the Pk measure allows some errors to go unpenalized because the measure does not take into account that there could be more than one boundary in the window. Another shortcoming of the measure is that it penalizes near misses too much. Even though the idea is that these types of errors are less penalized than false positives and false negatives. Lastly the measure is difficult to interpret because it doesn't reflect the function of the algorithm (Pevzner, L., & Hearst, M. A. (2002)).

1.2.3 WindowDiff

Theory

In order to solve the issues of the Pk measure, Pevzner, L., & Hearst, M. A. (2002) introduced WindowDiff. This metric makes an amendment to the Pk measure. As the name suggests the concept of a window or probe moving along the stream of words is used here as well. The biggest difference with the Pk measure is what counts as an error within the window. In WindowDiff, the number of boundaries in the reference segmentation and in the hypothesised segmentation within the window are compared, for each word position in the text. If these numbers are not the same a penalty is given that increases the error score. Whereas in the Pk measure the presence of a boundary is compared. The penalty that is given in WindowDiff is the absolute difference between the number of boundaries in the reference and hypothesised segmentation. WindowDiff, like the Pk metric, is an error metric between 0 and 1. Where 0 is the best it can get and 1 is the worst.

Example

Let us look at the example in figure 1 to calculate the WindowDiff metric for Ref and A-0, denoted as WD(Ref, A-0). The bitstrings are repeated here for ease of reference.
Ref: 000101000
A-0: 001000100
A-1: 100000011

The first step is the same as before with the Pk metric: calculate the size of the window; k. Since the same example is used here, k = 2.

The second step is to count the number of boundaries for Ref and A-0 within the window. If this number is not the same for Ref and A-0 the error score is increased with the absolute difference between the two numbers of boundaries. Below similar 2 position lists are noted, because k is 2 here, as well. In the lists where Ref and A-0 are compared, there are no longer true and false symbols, but numbers denoting the absolute difference in number of boundaries.

Ref: [00, 00, 01, 10, 01, 10, 00, 00]
A0: [00, 01, 10, 00, 00, 01, 10, 00]
(==): [0, 1, 0, 1, 1, 0, 1, 0]
The error score is 4.
The normalized error score is:
$$WD(Ref, A0) = 4/(len(Ref) - k + 1) = 4/(9 - 2 + 1) = 4/8 = 0.5$$

Lastly we can compute WD(Ref, A-1), to compare WD(Ref, A-0) against.

Ref: [00, 00, 01, 10, 01, 10, 00, 00]
A1: [10, 00, 00, 00, 00, 00, 01, 11]
(==): [1, 0, 1, 1, 1, 1, 1, 2]
The error score is 8.
The normalized error score is:
$$WD(Ref, A1) = 8/(len(Ref) - k + 1) = 8/(9 - 2 + 1) = 1$$

If we compare these results to the previous ones with the Pk metric, we can see that only WD(Ref, A-1) is different compared to Pk(Ref, A-1). WindowDiff took into account that there is an extra boundary in A1 within the window, which increased the error score from 0.875 to 1. The worst it can get.

Issues

Scaiano, M., & Inkpen, D. (2012) have summarized a few drawbacks of WindowDiff that have been noticed over the years. Firstly, the measure can't be adjusted to tolerate types of errors

over others. Secondly, false positives are more likely to occur with this measure. And lastly, errors near the beginning and end of a segmentation are counted less than other errors.

1.2.4 The selected metrics

A difficulty when evaluating text segmentation algorithms is that for different applications, different kinds of errors become important (Pevzner, L., & Hearst, M. A. (2002)). For this research it is important that near-misses don't count as false positives. That the evaluation metric is insightful and interpretable. And that the metric is being used in other text segmentation research, so it becomes possible to compare results with other research because we would somewhat share the notion of what 'a good result' means.

It is important that the evaluation metric is interpretable because the research will look at the performance of text segmentation on subtitles versus speech transcripts. A better interpretability would be more useful when thinking about the way the performance relates to the hypothesis about the different kinds of textual input. It would also help with thinking about and motivating choices relating to the tuning of parameters.

Other measures have been presented in the years after 2002 that attempt to resolve the issues of WindowDiff that were quickly mentioned before (Scaiano, M., & Inkpen, D. (2012)) (Fournier, C., & Inkpen, D. (2012)) (Fournier, C. (2013)). However, these metrics are more complex. For this research the earlier discussed metrics have a better balance between complexity, capability and interpretability. Furthermore, the Pk and WindowDiff measures are both often used as a metric in (recent) text segmentation studies (He, X., Wang, J., Zhang, Q., & Ju, X. (2020)). That is why I will report these two metrics.

To get a sense of whether the text segmentation algorithm puts too many or too few boundaries in the text, I will report this difference in number of boundaries and call it "Delta boundaries". In total, the Pk measure, WindowDiff, and Delta boundaries will be reported as evaluation metrics.

1.3 Data used in earlier research

Previous research contains different elements that are brought together in this research. Speech transcripts have been used before in the context of video retrieval (Hauptmann, A., & Smith, M. (1995)). And in the context of video segmentation (Repp, S., & Meinel, C. (2008)). Subtitles have been used before in the context of text segmentation (Scaiano, M., et al. (2010)) to segment movie subtitles.

LDA has been applied to search queries, historical documents, newspapers, scholarly literature, scientific publications, fiction, social media data (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017), educational materials (Steyvers, M., and Griffiths, T. (2007)), images, biological data, survey data (Alghamdi, R., & Alfalqi, K. (2015)), geographical data, patient record notes, drug reviews, music, dissertation abstracts, U.S. senator statements, blog posts, biological terminology sets,

Yelp data and Flicker data (Jelodar, H., Wang, Y., Yuan, C., et al (2019)).
It is not traditional to use ASR transcripts and subtitles with LDA but it has been done in the context of video retrieval (Cao, J., Li, J., Zhang, Y., & Tang, S. (2007)). LDA has also been used to segment texts (Misra, H. et al. (2011)).

To my knowledge there has not been research that studied the effects of different data types used for text segmentation.

In order to form a hypothesis about the performance of text segmentation using topics derived with LDA on subtitles and transcripts, I will look at the research data characteristics and statistics.

Throughout this research ‘888’ will be used as the abbreviation for subtitles. A reference to the division within the dutch public broadcasting system that creates the subtitles. That in turn refers to the number on teletext that needs to be pressed to get the subtitles on TV. For the speech transcripts ‘ASR’ will be used as the abbreviation.

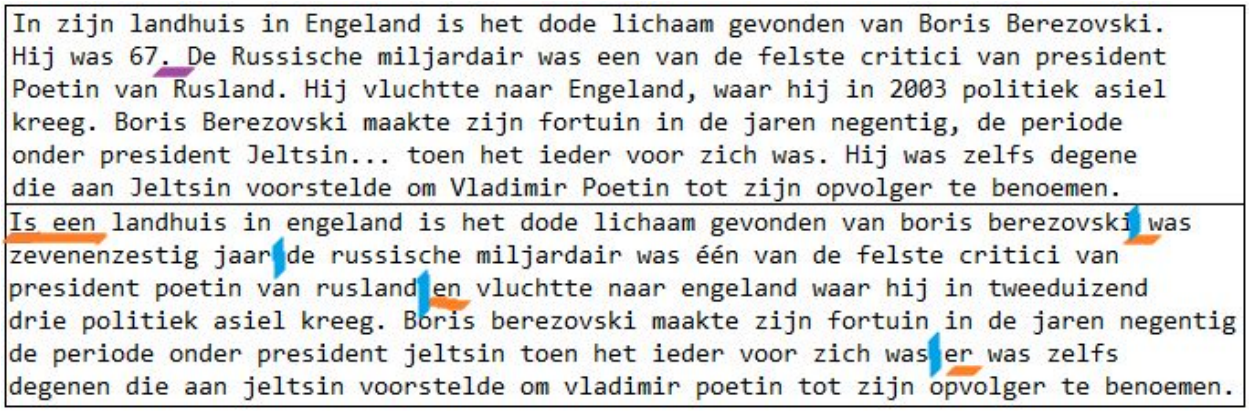
2. Data

This research will use Dutch news broadcasts (NOS) provided by The Netherlands Institute for Sound and Vision. There are two separate datasets: the subtitle dataset and the ASR dataset. In total 25,600 NOS broadcasts with subtitles and ASR transcripts are available. Ranging from 2011 to 2017 (incl). In both datasets the files are each labeled with the identifier of the broadcast video. For any of the broadcast videos, the accompanying files can be retrieved, meaning the subtitles and transcripts can be compared against each other and if need be the original video can be looked up.

2.1 Characteristics about the data

The subtitles and speech transcripts of the same broadcast are never the same. The same kind of differences can be noticed after repeatedly comparing them, as shown in figure 2.
Firstly, the sentences in the subtitles tend to be more concise. It leaves words out. Like the word 'jaar', meaning 'year' in figure 2 underlined in purple.
Secondly, the average length of sentences in the speech transcripts is longer. The full stops at the end of a sentence are not always present in the speech transcript. In figure 2 these would-be sentence boundaries are marked in blue.
Thirdly, the subtitles contain numeric characters, whereas numbers in the transcripts are written out.
Fourthly, the transcripts contain filler words like 'ehm' and 'mhmm', as opposed to the subtitles.
Fifthly, there are mistakes in the speech transcripts that are marked in orange. Words that might sound similar when they are uttered after each other like 'in zijn' and 'is een' are mistaken for each other. Pronouns might disappear in the speech transcript. Or be changed into another word altogether. For example, 'hij' that got changed into 'er' in the fourth line of figure 2.
Lastly, sometimes the mistakes are more semantically troublesome; the speech transcript might contain a string of words that don't grammatically make sense.

Figure 2
The text in the upper half are the subtitles. The text in the lower half is the speech transcript. The blue markings indicate sentence boundaries that are missing in the transcript but are in the subtitles. In orange the mistakes in the transcripts are noted. In purple a missing word in the subtitles is noted.



The ASR transcripts at the Sound and Vision Institute were generated using the Dutch KALDI speech recognition toolkit (Ordelman, R., & van der Werff, L. (z.d.)).

The subtitles in the dataset used in this research, were partly transcribed before the broadcasts were aired and partly as the broadcast was being aired. In either cases the final call is made by a human editorial team overseeing the quality of the subtitles. The subtitle dataset contains triple stars "***" denoting topic shifts. These labels were added by human annotators when the subtitles were being created. The speech transcripts do not contain labels denoting topic shifts.

2.2 Statistics of the data

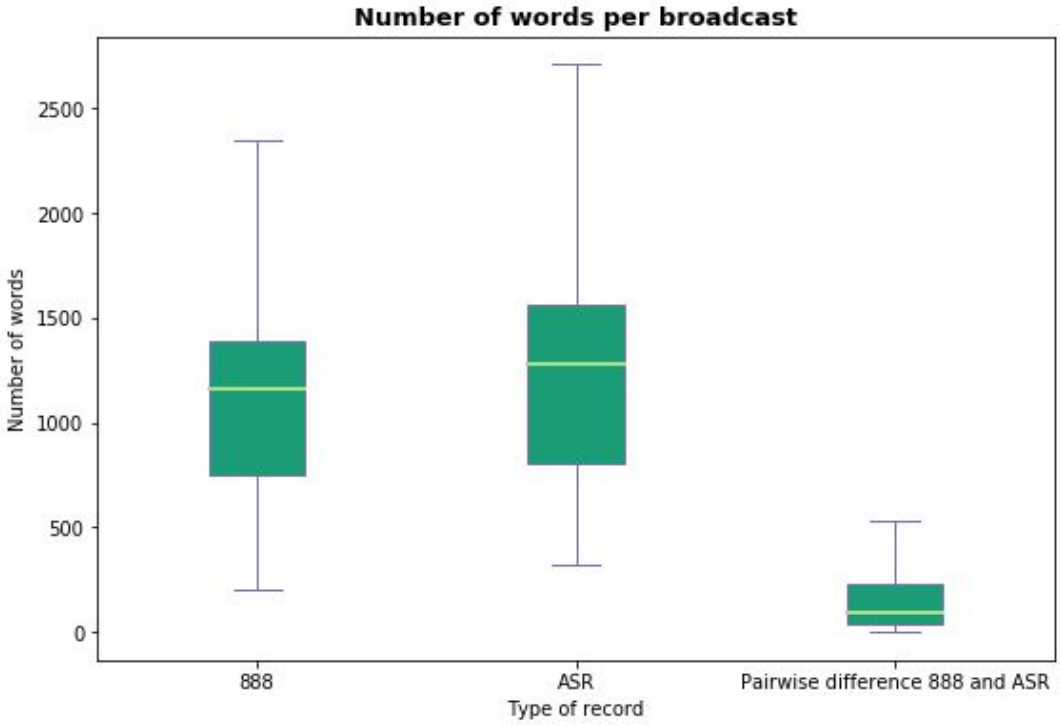
In the previous section, it was noted that the speech transcripts have longer sentences because sentences are merged without a full stop dividing them. The speech transcripts are also longer in terms of sheer number of words. In figure 3, we can see that. The number of words were found by splitting the text on spaces. The average number of words per broadcast for the subtitles is: 1,241 words. For the speech transcript this average is a little higher: 1,429 words. This was to be expected because subtitles are not a word for word record of what is being said, as opposed to ASR transcripts.

To get an idea of the pairwise difference in the number of words between the same transcript/subtitle broadcast, this difference is also plotted in figure 3, in the third boxplot on the right. The average pairwise difference in figure 3 is 198 words. The difference in the number of words between subtitles and transcripts can be twice this number, if we look at the figure in Appendix Section A, where the outliers are shown.

The maximum difference in the number of words between the same broadcasts is 12,625 words. These outliers are often live news broadcasts of special occasions like King's Day and breaking news broadcasts containing a lot of interviews that are not all transcribed in length in the subtitles.

Figure 3

Number of words per broadcast in the different types of record being used in this research. Namely, ASR and subtitles. Outliers are not shown.



As noted in the previous section, the speech transcripts do not contain topic labels, whilst the subtitles do contain them. In the subtitle dataset, the number of topics per broadcast can be calculated. The average number of topics per broadcast is 6.6 topics.

2.3 Training, development and test sets

To properly evaluate the text segmentation algorithm later on, I will split the dataset consisting of the 25,600 transcripts and subtitles into a training-, development- and test set.

A perfect divide of the dataset into these subsets does not exist, because there are benefits and downsides to all partitions. If a model is trained on more data it is likely to be better, which is a reason for wanting a large training set. If the development set is large, the parameters in the model can be tuned better. If the test set is large, more can be said about the generalizability of the model. Making any of these sets larger unfortunately means that another one has to shrink.

Following a few rules of thumbs we will divide the dataset here into these parts: 65% training set, 15% development set and 20% test set (Elkan, C. (2012)).

Since there won't be a large number of parameters to tune, the development set is chosen to be slightly smaller than the testset. The parameters that are interesting to pay attention to will be noted in the Section 4.2.1.

The broadcasts that will go in the different sets will be randomly selected, in order to make the testset representative of the data in the other sets (Elkan, C. (2012)). Though the broadcasts are shuffled, the same randomly selected broadcasts will be selected to go in the training-, development- and test set for both transcripts and subtitles. This will be done in order to be able to measure and compare the performance on the same broadcast for the different kinds of broadcast records (subtitles and speech transcripts).

The broadcasts will be clustered around the same period before being shuffled, because of the similarity between different broadcasts that were aired around the same time. For example, the evening news and the morning news might overlap a lot. If the evening news is in the testset and the morning news in the training set, this could be as unideal as putting the same broadcast in both sets. Clustering the broadcasts around the same period of time will be done manually. Since all the files have the date and time of airing in the name of the file, these files can be ordered in Windows by sorting the maps on filename. Afterwards the same clusters of files will be moved to the training, development and test sets.

3. Methodology

Let's now move on to the method for researching whether ASR or subtitles work better to segment a text based on topics. First the process that will be used to find the topics will be explained (Section 3.1). Secondly the process of using these topics to segment the text (Section 3.2). And lastly the planned process of evaluating these text segmentations (Section 3.3).

3.1 Topic detection

3.1.1 Preprocessing

A few things will be done to the texts in order to be able to apply LDA to it: tokenizing, stopping and bigram collocation.

Tokenizing converts the text into words. Stopping removes the most commonly used Dutch (stop)words gathered by *Gemoderniseerde stopwoorden lijst*. (z.d.). This list contains 501 words.

After stopping, the bigram collocation finds combinations of two words that commonly co-occur. For example "prime minister" and "social media". Logically the sentences have to be split into words before it is possible to remove commonly occurring ones. The reason that stopping is done *before* bigram collocation is to prevent bigrams like 'over the'. If stopping is done before bigram collocation, these commonly occurring words will be gone, which means those types of bigrams will not be found.

Another thing that is done in the preprocessing stage is the removal of accents and other special symbols. Accents are removed to merge words that alternately have those accents. For example there are cases in which a text includes the word "Israel" and "Israël", these are semantically the same. To map them to the same word types, the accents are removed. For this same reason all words are converted into lowercase tokens.

The words that are left are used to train a topic model. That topic model can then in a later stage be used to detect topics on similarly preprocessed texts. After preprocessing the speech transcript training data there were 7,343,667 words left. For the subtitles this was 6,462,130 words.

3.1.2 Training

In order to train a topic model using LDA, two software libraries will be used: Mallet and Gensim. Mallet uses Gibbs sampling to infer the topics (Boyd-Graber, J., Hu, Y., & Mimno, D. (2017)). Gensim contains a wrapper for Mallet that allows the use of Mallet in Python. Mallet was originally written in Java.

To train a topic model, the number of topics needs to be specified beforehand. Determining the right number of topics is a difficult task as noted in Section 1.1.3, even if heuristics are used. For example, a heuristic like the average number of topics per news broadcast that was calculated from the annotated subtitle dataset (Section 2.2). It would be difficult to use that as the number of topics to give to LDA, because we do not have any information about the overlap between those broadcasts in the dataset. I mentioned briefly that the morning news and evening news might overlap a lot. This means that the number of different topics in the broadcasts that are aired around the same time is less than the number of topics denoted by the topic shifts. One can imagine that words describing a topic that occurs a lot like a fire or a soccer match occur frequently throughout the entire dataset regardless of whether the broadcasts were aired around the same time. This is why it is difficult to come up with a heuristic for the number of topics to look for and also the reason I will experiment with this parameter in Section 4.2.

3.2 Text segmentation

3.2.1 Steps

Once the topic model has been trained, it can be used to predict the topics of previously unseen texts. This will be used by the text segmentation algorithm to segment texts in the following five-step process.

1. The to-be segmented text will first be randomly cut in segments. The length of these segments will be called *the first cut size.* If a number of sentences remain at the end of the text that are smaller than the first cut size, they are merged with the last segment.
2. The topic model will predict the topics for each of those segments.
3. A similarity approach will determine whether two consecutive segments share similar topics.
4. If two consecutive segments share similar topics they will be merged together.
5. The merged segments that remain after the last iteration of step 4, are the segments that the text segmentation algorithm will return.

There are two methods for determining the similarity in step three, that will be implemented and compared, after which the best method will be chosen: Maximum probability (Section 3.2.3) and Jensen-Shannon distance (Section 3.2.4).

3.2.2 The first cut size

In the methodology of text segmentation (Section 3.2.1) the first step - cutting the broadcast in segments - is important, because it determines whether the correct text segmentation can be found. If the first cut size is too large, the topic boundaries in the text might be missed. If the first cut size is smaller there will be more opportunity to merge segments.

Although it might intuitively seem that choosing a very small first cut size is the best option, there is a catch. If the number of sentences in a segment is too short, the topic model does not have a lot of information to go on. This causes the topics that are predicted by the topics model to have low probabilities. This is not ideal, because it is unclear what a text is about if it could be about a lot of different topics that have low probabilities.

In Table 1, this intuition is shown; a smaller first cut size causes the probabilities to be lower. The probabilities are not the same for the subtitles and transcripts, although the probabilities do increase with similar increments with a larger first cut size. This difference between the subtitles and transcripts can be explained by the data statistics (Section 2.2). Since transcripts contain longer sentences, a single transcript sentence is more informative than a subtitle sentence. This causes the probability of the highest predicted topic for ASR to be higher than for subtitles, given the same first cut size. In Table 1 for example, given a first cut size of 3 the probability for ASR is 0.06 and for subtitles it is 0.03.

Table 1

The average probability of the highest predicted topics calculated from the development set are presented. Given a list of texts that each have the same number of sentences, and a topic model that has n topics, the topics for each text can be predicted. All n topics will have a certain probability for each text. The average highest probability over all texts in the list is noted for different sizes of that text. This size is denoted in the number of sentences.

Number of sentences	3	5	7	10	15	20	30	40
Average probability of highest predicted topics on ASR broadcasts.	0.06	0.08	0.10	0.12	0.14	0.15	0.16	0.16
Average probability of highest predicted topics on 888 broadcasts.	0.03	0.04	0.05	0.05	0.06	0.07	0.08	0.09

The chosen number of sentences cannot be too large because it could cause topic boundaries to be missed. But it should not be too small either, because that causes topic probabilities to be too low to decide which topics are important. In Section 4.2.2 the best values are presented after parameter tuning.

3.2.3 Maximum probability

Suppose we have two segments, seg_1 and seg_2, and a topic model T that was trained to look for ten topics. Suppose that T finds these ten topics: $t_1, t_2, t_3, \ldots, t_{10}$ where t_i denotes a topic. Both seg_1 and seg_2 will have a distribution over those topics denoting the likelihood that each segment is about one of those topics:

$$seg_1 : (t_1 \times prob_1) + (t_2 \times prob_2) + \ldots + (t_{10} \times prob_{10})$$
$$seg_2 : (t_1 \times prob_a) + (t_2 \times prob_b) + \ldots + (t_{10} \times prob_j)$$

The first approach for determining the similarity between seg1 and seg2, is to only look at the topics that have the *highest* probability. Suppose that in this example, $prob_1$ has the highest probability for seg_1 and similar for $prob_a$ in seg_2. Since these are both about the likelihood of topic t_1, it means that both segments are most likely about that topic. Using this approach these segments would be merged together.

This approach is called maximum probability because it disregards all other topics that do not have the highest probability in order to decide whether two segments should be merged.

There are theoretical upsides and downsides to this method. The upside is that it's quick to only consider the topic that has the highest probability. The downside is that the information about the other topics in the topic distribution gets ignored. This could firstly cause a problem if the probabilities of topics are close to each other. Picking only one of two topics that have similarly high probabilities might prove not to be informative enough to make a good decision. A second problem is related. The method could be sensitive to background topics that have high probabilities for every document. Should those topics get the highest probability it would not be informative to what a piece of text is about.

3.2.4 Jensen-Shannon distance

The second approach for determining whether to merge seg1 and seg2 is by looking at the entire topic distributions of both segments. To determine whether topic distributions are similar enough to merge both segments, the Jensen-Shannon distance will be used.

The Jensen-Shannon distance is a metric for determining the similarity between two probability distributions. If the two distributions are exactly the same the returned distance is 0. Otherwise, the Jensen-Shannon distance is larger than 0 and at a maximum 1.

If the returned distance is smaller than a certain boundary, the segments that were assigned those topic distributions are merged together. The boundary here is a parameter that will be called the JS-parameter. A small JS-parameter makes the threshold more difficult to pass, since a small Jensen-Shannon distance would be needed to merge those segments. That means that the probability distributions of the segments must look more alike. The JS-parameter will be tuned in Section 4.2.

The upside to this method is that it takes into account that multiple topics might be alike in the topic model. For example, a text segment might be assigned topic 3 as the topic with highest probability and the consecutive segment might be assigned topic 7. If these topics are similar to each other, the first text segment must have topic 7 assigned as well, with a semi-high probability and vice versa. Since the Jensen-Shannon method takes the entire distribution into account, it would sense that these segments are about similar topics.

3.3 Evaluation

3.3.1 Ground truth segmentations

The different evaluation metrics (Pk, WindowDiff, Delta boundaries) that were discussed earlier (Section 1.2.4) all assume there is a reference text segmentation to compare the hypothesised one against.

The correct text segmentation for a news broadcast transcript is easy to see for humans. For a quick test, two people segmented 5 speech transcripts to compare their segmentations and they agreed on all but one topic boundary. Though this is a small sample size, it indicates that for news stories it is quite clear where new topics start.

Luckily I will not have to label all the data myself, because the subtitles contain topic shift labels (Section 2.1).

This leaves the question of how to evaluate the text segmentations on the ASR transcripts. Since the topic boundaries cannot be moved automatically from the subtitles to the speech transcripts in a straightforward way, because of the differences between them, I will have to label a part of the speech transcripts myself. These labels can then be used as a golden standard. Details on how I will manually label the speech transcripts can be found in Appendix Section B. For the development and test set I will each label 31 broadcasts, 62 in total.

After manual labeling, evaluation will be possible for the text segmentation algorithm on both subtitles and speech transcripts. The question that remains then is how this process will be set up in order to compare the performances of both kinds of input data.

3.3.2 Experimental setup

To investigate whether transcripts or subtitles are more suitable for text segmentation, the experiment could be set up in two ways.

The first approach, suppose we have a text segmentation algorithm that uses a topic model trained on 50/50 percent ASR and subtitle data. We could use that algorithm to segment on the one hand ASR data and on the other subtitle data. Afterwards the performance on both sets could be compared to draw a conclusion.

The second approach, suppose we have a text segmentation algorithm that uses a topic model trained on ASR data, namely T1. And another text segmentation algorithm that uses a topic model trained on subtitle data, namely T2. In this approach the effect of a match between the training data used for the topic model and the data used to segment the text can be studied. This is why this second approach has my preference.

The experiment will be set up like this: T1 will use a topic model that was trained on ASR data. T2 will use a topic model that was trained on subtitle data. The following four conditions will be compared to draw conclusions about the performance of the text segmentation algorithm on ASR transcripts and subtitles:

(1) Topic model trained on ASR data. Text segmentation on ASR data ($T_{ASR-ASR}$)
(2) Topic model trained on ASR data. Text segmentation on subtitle data ($T_{ASR-888}$)
(3) Topic model trained on subtitle data. Text segmentation on ASR data ($T_{888-ASR}$)
(4) Topic model trained on subtitle data. Text segmentation on subtitle data ($T_{888-888}$)

In machine learning tasks the performance tends to be better if there is a match between the training and testing data. Let's assume this is the case and call that the "match assumption". Then, the conditions that need to be compared to conclude whether a different kind of data affects the performance of the text segmentation algorithm are conditions (1) and (4).

The match assumption will be studied by comparing condition (1) to condition (2) and condition (3) to condition (4) as a check.

3.3.3 Hypothesis

To answer the research question with hypothesis testing (Section 3.3.4), I need a hypothesis about the results. Since the steps of the text segmentation algorithm are the same for all data and I do not yet know the parameter influences, I will focus on the topic modelling aspect of the algorithm to formulate a hypothesis. Considering the theory and methodology, what are the expected results of topic modelling applied to ASR and subtitles? What difference in results (if any) can be expected here?

One guess could be that subtitles contain more semantic structure because human annotators are behind them. They could have added an extra layer of meaning that can help create the topic model. A layer that doesn't exist in the speech transcripts. This guess would be incorrect. If we look at the methodology and how the topic model is trained, we can see that this cannot be the case. The texts are stripped down to words and special symbols are removed. There is no way that information about where topics are could accidentally make it into the topic model. Mainly because the parts in the text that caused the topic model to find a particular topic is not information that a topic model returns as a result. This means that the performance of the text segmentation algorithm on the subtitles can't be higher than on the speech transcripts. At least not for this reason.

If we look at the data characteristics mentioned in Section 2.1, there is a reason that the speech transcripts might perform better and another reason that suggests it will perform worse. The speech transcripts are less concise than the subtitles and contain more words. The words are essentially used to figure out the topics and this could be helped along, if more important

keywords are in the text. Furthermore, the speech transcripts sometimes contain redundancies by mistake, which could end up helping create a better topic model.
A reason that the speech transcripts could perform worse is simply because it contains mistakes. Some words are wrong, which would mean fewer keywords for determining the right topics.

It could really go either way, the subtitles could perform better or the speech transcripts could, which makes it all the more interesting to figure out. It is of course also possible that the performance of both input data types is the same and no significant difference between the performance can be found. Seeing as one type of data is computer generated and the other one is not, that could still be quite an interesting result.

Since there is no real reason to assume that subtitles will outperform the speech transcripts or vice versa, a two-tailed test will be performed. What will be tested then is whether a difference in performance for using speech transcripts and subtitles exists.

3.3.4 Hypothesis testing

Once the results have been gathered for the different conditions mentioned in Section 3.3.2, they will be analyzed to see whether the different kinds of input data have a significant effect on the performance.

The null hypothesis states that the average performances are the same for both input data.
$H_0 : \mu_{T\ ASR-ASR} = \mu_{T\ 888-888}$

The alternative hypothesis states the opposite; the performance on both kinds of data is different.
$H_a : \mu_{T\ ASR-ASR} \neq \mu_{T\ 888-888}$

To determine whether the null hypotheses can be rejected, a statistical test will be used. Here, a permutation test has been chosen because it does not require that the results are distributed in a specific way, such as a normal distribution.

A permutation test works in the following way. Given the two groups ($T_{ASR-ASR}$ and $T_{888-888}$), a number of simulations will be run, in which the labels of the results for each group will be randomly swapped. In this case 10,000 simulations will be run. The name ‘permutation test’ refers to this label swapping, where the results of the groups are permuted. If the two groups do not significantly differ, swapping the labels of the results should not make the average result for that group very different from the average result before swapping labels. This is the basis for deciding whether to reject the null hypothesis. The (two-sided) p-value is the proportion of permutations where the absolute difference between the groups *was* greater than or equal to the absolute difference before swapping the labels (Dror, R.,et al (2018)).

4. Results

4.1 Early qualitative results and adaptations

A few adaptations were made after testing the text segmentation algorithm on small examples. The topics that were found by the topic model were often meaningful; they contained words that seem to make sense together. However some of the topics seemed to capture an incoherent set of words. Most of those words were Dutch verbs like (translated) go, went and think. Other words in incoherent topics were words like year, child, human and words denoting numbers.

Many segments often got those incoherent topics ascribed to them, because that segment contained those words. This caused the text segmentation algorithm to merge a lot of different segments together, because they were assigned the same generic topics. This issue was more present when segmenting ASR transcripts data than subtitles data.

The words that are frequently in meaningless topics, are not traditionally Dutch stopwords - which is why they weren't removed in the preprocessing stage. Although they do occur a lot in this particular news broadcast dataset. That is why a part of these frequent generic words were added to the stopword list (which increased in size from 501 to 627 words) and were both removed in the LDA and text segmentations stage. The words that were chosen to be removed were often verbs and otherwise were judged not to be meaningful enough to make it into a topic. The list of stopwords can be found in Appendix Section C. Although the effect of adding these words on results for the text segmentation was not tested, the problem of many segments being ascribed the same generic topics slightly bettered.

4.2 Parameter tuning

First I will list all parameters that have been mentioned and expand on how these will be tuned (Section 4.2.1). Afterwards, the best parameter values will be presented (Section 4.2.2).

4.2.1 Tuning process

Different parameters

The different parameters that will be tuned in the next section are noted in Table 2.

Table 2

The to-be tuned parameters are presented. The first row contains the parameters for the topic model. The second row contains the parameters for the text segmentation stage. The names of the parameters are in bold. Following that is a cursive explanation of the parameter. Lastly, the options that will be tried for each parameter are noted.

Topic model	**Number of topics** *The number of topics that LDA is told to look for when creating a topic model.* **Options**: 100, 200	**Document or segment** *The way a document is defined in LDA. A document can be a segment of a broadcast or the entire broadcast.* **Options**: “document”, “segment”	**Initial segment size** *If a document in LDA is a segment. The size of this segment is the initial segment size, defined in number of sentences.* **Options**: 10, 20
Text segmentation	**The first cut size** *The first step of the text segmentation algorithm is cutting the broadcast in segments (Section 3.2.1).The size of these segments is the first cut size, defined in the number of sentences.* **Options**: 2, 3, 5, 7	**Text segmentation method** *There are two methods for determining the similarity between two segments and deciding whether to merge them together.* **Options**: “Maximum probability”, “Jensen-Shannon”	**JS-parameter** *The threshold parameter in the Jensen-Shannon method for determining whether two probability distributions are similar enough (Section 3.2.4).* **Options**: 0.3, 0.6

Tuning process

The parameters will be tuned by trying out the different combinations and evaluating the text segmentation algorithm on the development set. The parameter value combination that results in the best Pk metric will be selected. It is also possible to select parameters based on the best WindowDiff measure or Delta boundaries (Section 1.2.4), but the Pk measure is the metric that is used most in previous studies and is still being used in recent studies (He, X., Wang, J., Zhang, Q., & Ju, X. (2020)).

The parameter values will be tuned for two text segmentation algorithm variants, each using different data. One uses a topic model that was trained on ASR data to segment ASR data ($T_{ASR-ASR}$). The other uses a topic model trained on subtitle data to segment subtitle data ($T_{888-888}$).

4.2.2 Best parameters values

In Table 3 and Table 4 (p.31/32) the parameter results for $T_{ASR-ASR}$ and $T_{888-888}$ respectively are presented. The best five parameter combinations (based on their Pk value) are listed for the Maximum probability and Jensen-Shannon method. For the best parameter combination (the top one), the other evaluation metrics are also noted.

The differences between some of the evaluation metric results are very small. However, the best five parameter combinations were consistent through running the process several times even though probability distributions are involved in the process.

I will now discuss the results for the parameter tuning, starting with the ones for which the optimal values was the same for both $T_{ASR-ASR}$ and $T_{888-888}$: *the text segmentation method, the JS-parameter and the number of topics.*

The text segmentation method

The options are: "Maximum probability" and "Jensen-Shannon".
The best method for $T_{ASR-ASR}$ and $T_{888-888}$ is the Jensen-Shannon method.
For $T_{ASR-ASR}$ let us look at Table 3. In the evaluation results for the top one, on the far right of the table, we can see that the Pk results between both methods are not so different (both 0.4 rounded off). However if we look at the Delta boundaries metric, the results for the Jensen-Shannon method are better than for the Maximum probability method. In the latter method the average is 2.00 and standard deviation is 6.75. In the former method the standard deviation is lower, 1.53, albeit the average of -4.18 is slightly farther from 0 than for the Maximum probability method.

For $T_{888-888}$ in Table 4, the difference between the possible methods is greater than for $T_{ASR-ASR}$ in Table 3. The average Pk score for the Jensen-Shannon method is lower (0.466) compared to the other method (0.520). The same goes for the WindowDiff score which is more visibly lower for the Jensen-Shannon method (0.479) compared to the other method (0.699). The Delta boundaries metric is also significantly closer to 0 for the Jensen-Shannon method (-5.46) than for the Maximum probability method (14.03).

The JS-parameter

The options are: 0.3 and 0.6.
As can be seen in Table 3 and 4, for both $T_{ASR-ASR}$ and $T_{888-888}$, the best parameter here is 0.3. There is no parameter combination that made the top five that contained another value for the JS parameter. The JS-parameter is the largest Jensen-Shannon distance that is allowed for merging two segments (Section 3.2.4). This means a more strict parameter works better. In other words a smaller upper bound of 0.3 works better than a more lenient 0.6.

The number of topics

The options are 100 and 200.

For both $T_{ASR-ASR}$ and $T_{888-888}$, the best number of topics given that the Jensen-Shannon method will be used is 200. If we look at Table 3 and Table 4 the top three parameter combinations contain 200 as the number of topics for the Jensen-Shannon method. This is contrary to the Maximum probability method which has 100 as the number of topics for the top three parameter combinations in Table 3. This result can be understood if we think about both methods. Since the Jensen-Shannon method takes the entire probability distribution into account, more topics (200 as opposed to 100) actually provide more information. However, for the Maximum probability method, a huge portion of the probability distribution is ignored. It could make sense that effectively ignoring less of the distribution works better than ignoring more of the distribution. Hence a 100 topics works better for that method than 200.

I now will expand on the three remaining parameters for which the best values are different for $T_{ASR-ASR}$ and $T_{888-888}$.

The first cut size

The options are: 2, 3, 5 and 7.

For the speech transcripts, in Table 3, the best size is 3 sentences. For the subtitles, in Table 4, the best size is about 5 sentences. This result is not strange, given the statistics about the data (Section 2.2). Since the transcripts have longer sentences than the subtitles, the best size for the transcripts is smaller than for the subtitles.

Document or segment and Initial segment size

The options for Document or segment are: "document" and "segment".

The options for the Initial segment size are: 10 and 20.

For the document or segment parameter there is no clear answer as to which one is better. Given that the answer would be ‘segment’ and we look at the initial segment sizes, these vary a lot too, especially for the Jensen-Shannon method in Table 4. However if we look at Table 3, the top three parameter combinations contain ‘seg’ for the Jensen-Shannon methods with the same accompanying initial segment size of 20. This hints at a longer segment being better than a short segment. The longest segment would be the entire document. Since there is no consistent answer in the results here, I will take that hint and choose ‘doc’ as the best parameter.

Table 3

Parameter tuning results for the text segmentation algorithm that uses a topic model trained on ASR to segment ASR texts. The top 5 parameter combinations are shown (based on Pk value), given a certain method (Maximum probability or Jensen-Shannon). A good Pk or WindowDiff value is close to 0. The best being 0 and the worst being 1. Delta boundaries can be positive or negative and respectively indicates too many or too few boundaries detected by the algorithm. The best value there is 0.

$T_{ASR-ASR}$	1)	2)	3)	4)	5)	Dispersion Pk in top 5	Evaluation of best top1 result. (mean, std)
Maximum probability						0.436 - 0.452	
Number of topics	100	100	100	200	200		**Pk:** (0.436, 0.090)
Doc or seg	doc	seg	doc	doc	seg		**WindowDiff:** (0.488, 0.081)
Init seg	-	20	-	-	20		**Delta boundaries:** (2.00, 6.75)
First cut size	2	2	2	3	2		
Jensen-Shannon						0.442 - 0.488	
Number of topics	200	200	200	100	200		**Pk:** (0.442, 0.11)
Doc or seg	seg	seg	seg	doc	doc		**WindowDiff:** (0.465, 0.096)
Init seg	20	20	20	-	-		**Delta boundaries:** (-4.18, 1.53)
First cut size	3	3	2	2	3		
JS-parameter	0.3	0.3	0.3	0.3	0.3		

Table 4

Parameter tuning results for the text segmentation algorithm that uses a topic model trained on subtitle data to segment subtitles. The top 5 parameter combinations are shown (based on Pk value), given a certain method (Maximum probability or Jensen-Shannon). A good Pk or WindowDiff value is close to 0. The best being 0 and the worst being 1. Delta boundaries can be positive or negative and respectively indicates too many or too few boundaries detected by the algorithm. The best value there is 0.

$T_{888-888}$	1)	2)	3)	4)	5)	Dispersion Pk in top 5	Evaluation of best top1 result. (mean, std)
Maximum probability						0.519 - 0.525	
Number of topics	200	100	100	200	100		**Pk:** (0.520, 0.123)
Doc or seg	seg	seg	doc	seg	seg		**WindowDiff:** (0.699, 0.185)
Init seg	10	10	-	10	10		**Delta boundaries:** (14.03, 11.81)
First cut size	5	5	5	3	3		
Jensen-Shannon						0.466 - 0.468	
Number of topics	200	200	200	100	100		**Pk:** (0.466, 0.135)
Doc or seg	seg	seg	doc	seg	doc		**WindowDiff:** (0.479, 0.130)
Init seg	10	20	-	20	-		**Delta boundaries:** (-5.46, 3.34)
First cut size	7	5	3	2	2		
JS-parameter	0.3	0.3	0.3	0.3	0.3		

4.3 The best segmenter

4.3.1 Results for best segmenters on test set

Given the chosen parameter values (Section 4.2.2), these will be used to test the performance of the text segmentation algorithms on the test set.

In the experimental setup (Section 3.3.2) two things are outlined that will be tested. Whether a match in training and test data influences results (comparing (1) to (2) and (3) to (4)). And the research question: if there is a difference in results between using speech transcripts and subtitles (comparing (1) and (4)).

(1) Topic model trained on ASR data. Text segmentation on ASR data ($T_{ASR-ASR}$)
(2) Topic model trained on ASR data. Text segmentation on subtitle data ($T_{ASR-888}$)
(3) Topic model trained on subtitle data. Text segmentation on ASR data ($T_{888-ASR}$)
(4) Topic model trained on subtitle data. Text segmentation on subtitle data ($T_{888-888}$)

Table 5
Results for different conditions in experimental setup. The mean Pk, WindowDiff and Delta boundaries scores are noted. For all metrics the best score is 0. For Pk and WindowDiff the worst score is 1. For Delta boundaries, a positive or negative number respectively denotes too many and too few boundaries detected by the algorithm.

(mean, std)	(1) $T_{ASR-ASR}$	(2) $T_{ASR-888}$	(3) $T_{888-ASR}$	(4) $T_{888-888}$
Pk	(0.47, 0.05)	(0.49, 0.04)	(0.47, 0.07)	(0.49, 0.04)
WindowDiff	(0.47, 0.04)	(0.49, 0.04)	(0.47, 0.07)	(0.49, 0.04)
Delta boundaries	(-7.12, 2.72)	(-7.74, 2.90)	(-7.10, 2.70)	(-7.71, 2.91)

In Table 5 the results for the different conditions are presented. On a first inspection, the results between the different conditions do not differ a lot. Although the results of condition 1 are similar to 3 and condition 2 is similar to 4.

These results will first be compared to random baselines (Section 4.3.2). Afterwards the statistical test will be used to determine the significance of the different conditions on the results (Section 4.3.3).

4.3.2 Comparing with random baselines

In order to evaluate the quality of the text segmentation algorithm that uses the best parameters I will compare the results (Table 5) with two random segmenters.

Random segmenter **R1** segments a news broadcast by generating a random bitstring that has the length of that news broadcast, where a 1 and 0 have equal probability. This segmenter does not use any information relating to topics to decide where topic boundaries should be. Random segmenter **R2** starts with a string containing all zeros that has the length of the news broadcast and flips 7 bits randomly from a 0 to a 1. This means that it will look for 7 topic boundaries. This is a more informed random segmenter. The 7 topics are derived from the average number of topics in the annotated subtitle dataset (Section 2.2).

A random baseline is usually a good indication of how good a method is. It should be noted that in the context of text segmentation this is less true. Since topic boundaries can occur anywhere in a video or text it is very specific to the particular instance. This means that the possible places where topic boundaries can be 'randomly' placed is huge and it is not all that telling to have a method that is better than randomly putting a boundary at every possible place. This would be baseline R1. On the flipside a more informed random baseline - this would be R2 - is not that indicative either, because it is not random anymore. It *does* use information to make the decision. That being said, comparing the results to these baselines can provide some insight because both ends of this random baseline spectrum - R1 and R2 - will be reported.

Table 6

The results of the random baselines segmenting different types of texts (noted in underscore) are presented. The results for the text segmentation algorithms first presented in Section 4.3.1 are repeatedly presented for comparison. Those results are the same as in Table 5, for clarity the standard deviations are omitted. For all metrics the best score is 0. For Pk and WindowDiff the worst score is 1. For Delta boundaries, a positive or negative number respectively denotes too many and too few boundaries detected by the algorithm.

(mean)	(1) $T_{ASR-ASR}$	(2) $T_{ASR-888}$	(3) $T_{888-ASR}$	(4) $T_{888-888}$	$R1_{ASR}$	$R1_{888}$	$R2_{ASR}$	$R2_{888}$
Pk	0.47	0.49	0.47	0.49	0.51	0.53	0.57	0.56
WindowDiff	0.47	0.49	0.47	0.49	1.0	1.0	0.60	0.58
Delta boundaries	-7.12	-7.74	-7.10	-7.71	5280	$1.18e^{6}$	-2	0.25

As can be seen in Table 6, the random baselines, R1 and R2, both perform worse than the text segmentation algorithms. Regardless of the kind of input data the text segmentation algorithm uses. And regardless of whether we are comparing the algorithm with $R1_{ASR}$, $R1_{888}$, $R2_{ASR}$ or $R2_{888}$. The Pk and WindowDiff scores for $R1_{ASR}$, $R1_{888}$, $R2_{ASR}$ and $R2_{888}$ are all higher - which is worse - than those scores are for the text segmentation algorithm. The WindowDiff scores - which is stricter than the Pk metric (Section 1.2.3) - are especially better for the text segmentation algorithm. If we look at the Delta boundaries results, we can see a few things. The results for R1 are far worse than for R2. The results for R1 are far worse than for the text segmentation algorithm. And the only results that are better than the text segmentation algorithm are achieved by R2 on the Delta boundaries. The results for R2 on Delta boundaries are closer to zero than any of the Delta boundaries results for the four text segmentation results.

This makes sense, because R2 is informed on that specific point. It uses the average number of boundaries derived from the annotated subtitle dataset. Although R2 outperforms the text segmentation algorithm on Delta boundaries, it does not on the Pk and WindowDiff metrics.

4.3.3 Statistical test

Now that the results for the different conditions have been gathered (Table 5), the statistical test can be carried out. That will point out whether transcripts or subtitles are more suitable for text segmentation.

I will carry out the statistical test for the different evaluation metrics (Pk, WindowDiff en Delta boundaries) between the different conditions ($T_{ASR-ASR}$, $T_{ASR-888}$, $T_{888-ASR}$, $T_{888-888}$). As noted in Section 3.3.1, I labeled 31 ASR broadcasts to be used here for $T_{ASR-ASR}$ and $T_{888-ASR}$. The subtitles for the same 31 broadcasts will be used for $T_{ASR-888}$ and, $T_{888-888}$.

Table 7

The p-values resulting from permutation tests are noted. In the second and third column, the match assumption (does a match in test and training data result in different performance) is tested by comparing condition $T_{ASR-ASR}$ to $T_{ASR-888}$ and condition $T_{888-ASR}$ to $T_{888-888}$ (Section 3.3.2). In the first column of this table, the conditions that need to be compared for the research question are tested.

(mean)	Research question $T_{ASR-ASR}$ $T_{888-888}$	Match assumption $T_{ASR-ASR}$ $T_{ASR-888}$	Match assumption $T_{888-ASR}$ $T_{888-888}$
Pk	Lower = 0.16 Upper = 0.18 Confidence = 0.99	Lower = 0.16 Upper = 0.18 Confidence = 0.99	Lower = 0.17 Upper = 0.19 Confidence = 0.99
WindowDiff	Lower = 0.21 Upper = 0.23 Confidence = 0.99	Lower = 0.19 Upper = 0.21 Confidence = 0.99	Lower = 0.24 Upper = 0.26 Confidence = 0.99
Delta boundaries	Lower = 0.42 Upper = 0.45 Confidence = 0.99	Lower = 0.40 Upper = 0.43 Confidence = 0.99	Lower = 0.41 Upper = 0.44 Confidence = 0.99

As can be seen in Table 7, with a significance level of alpha of 0.05, none of the hypotheses can be rejected. This can be caused by two things. Either the dataset of labeled ASR broadcasts is too small. That would mean that differences between both types of data could not be detected, because these broadcasts happened not to cause different results. Or there really is not a significant difference between using either kind of data, which means that increasing the set of labeled ASR broadcasts would not change the results. It is difficult to determine which explanation is correct, without actually increasing the number of labeled ASR broadcasts.

4.4 Error analysis

In order to get a better sense of the quality of the text segmentation algorithm and the challenges that could be taken up in future research, I will analyze the process and problems in a qualitative fashion.

4.4.1 Topics

Starting with the topics. In Table 8, a few of the topics of the topic models using the best parameters for the transcripts and subtitles are presented.

Table 8

For two topic models of which one is trained on transcripts and the other on subtitles, four example topics are noted. These models use the best parameters (Section 4.2.2).

	Example topic 1	Example topic 2	Example topic 3	Example topic 4
Transcripts	('0.006*"tijd" + 0.006*"keer" + 0.006*"amsterdam" + 0.004*"schepen" + 0.004*"ijmuiden" + 0.004*"sail" + 0.004*"minister" + 0.004*"kinderen" + 0.004*"landen" + 0.004*"zee"')	('0.008*"vrachtwagens" + 0.006*"grens" + 0.006*"hulpgoederen" + 0.006*"konvooi" + 0.006*"oekraine" + 0.005*"china" + 0.005*"buien" + 0.004*"twintig" + 0.004*"rusland" + 0.004*"rode_kruis"')	('0.021*"schiphol" + 0.017*"klm" + 0.006*"passagiers" + 0.006*"luchthaven" + 0.006*"reizigers" + 0.005*"gogh" + 0.005*"vliegtuigen" + 0.004*"vluchten" + 0.004*"week" + 0.004*"rij"')	('0.014*"prinsjesdag" + 0.012*"haag" + 0.009*"troonrede" + 0.008*"kabinet" + 0.007*"gouden_koets" + 0.005*"binnenhof" + 0.005*"koning" + 0.004*"procent" + 0.004*"vanmiddag" + 0.004*"kinderen"')
Subtitles	('0.013*"zon" + 0.009*"nieuwe" + 0.006*"slachtoffers" + 0.006*"geld" + 0.005*"bewolking" + 0.005*"werken" + 0.005*"samen" + 0.005*"elkaar" + 0.005*"tijd" + 0.004*"justitie"')	('0.087*"rusland" + 0.027*"poetin" + 0.026*"oekraine" + 0.022*"russen" + 0.021*"russische" + 0.015*"navo" + 0.015*"moskou" + 0.011*"sancties" + 0.010*"president_poetin" + 0.008*"russische_president"')	('0.005*"procent" + 0.004*"willem" + 0.004*"informatie" + 0.004*"antibiotica" + 0.003*"onderzoek" + 0.003*"aanslagen" + 0.003*"halen" + 0.003*"gevonden" + 0.003*"wedstrijd" + 0.003*"vis"')	('0.005*"regering" + 0.003*"westen" + 0.003*"regen" + 0.003*"enkelband" + 0.003*"nisman" + 0.003*"president" + 0.003*"tweede_kamer" + 0.003*"bersani" + 0.003*"gratis" + 0.003*"jaren"')

In Table 8, for each example topic, the ten most probable words for that topic are noted in descending probability. If we look at the first example topic for the transcripts, this means that the word that is the most indicative and most probable for that topic is ‘tijd’. If we look at that topic qualitatively, it could be about the sea around Amsterdam and the sail/travel on it. The first two words (‘tijd’ meaning time and ‘keer’ meaning times) have little to do with that. The topic is also quite specific because it mentions the words minister and children (in Dutch: minister, kinderen).

The third topic for the transcripts already looks better. Only one of the words seems odd or out of place there, namely ‘gogh’. All the other words have to do with air travel. We can see the probabilities for those words are higher than the previously discussed topic too. The topics that have high probabilities for the words generally look better.

For the subtitles, the topics don’t seem especially different. Some of the topics are good, others are bad. The second example topic is a good one; clearly about Russia and Ukraine. Not a single word in that topic is out of place. The first word there has the highest probability from all topics in that table. A worse topic is the third example topic of the subtitles, where the word ‘aanslag’ (meaning attack) is mixed with ‘vis’ (fish) and ‘antibiotica’ (antibiotics). That topic has low probabilities for each word in it.

The first topic-related problem is that some words are barely informative albeit they still make sense for that topic. For example here *('0.006*"water" + 0.005*"peking" + 0.005*"problemen" + 0.004*"straks" + 0.004*"bekend" + 0.004*"weken" + 0.004*"gevaarlijk" + 0.003*"minister" + 0.003*"daardoor" + 0.003*"huizen"').* I could imagine words like ‘weken’ (meaning weeks), ‘straks’ (later) and ‘daardoor’ (therefore) being used in a news segment about what seems to be a dangerous flood in ‘Peking’ (Beijing). However they are not so telling in this topic. These are words that *could be* important in a topic but aren’t in the one in question.

The second problem is that some words in topics are completely misplaced. It seems as though several subjects have been mixed. Like in this topic *('0.010*"nepal" + 0.008*"dordrecht" + 0.006*"tijd" + 0.005*"groningen" + 0.005*"gisteravond" + 0.004*"griekenland" + 0.004*"kinderen" + 0.004*"utrecht" + 0.004*"aardbeving" + 0.004*"plaats"').*This topic is not so telling. If I had to speculate, the words ‘groningen’ en ‘nepal’ are related because those are places where earthquakes occurred that made the news. The word ‘aardbeving’ means earthquake in Dutch. For the other words in the topic, I cannot think of why they are there.

The third problem is that some topics are very specific albeit they point to a coherent event. A small example is in this topic *('0.025*"club" + 0.016*"knvb" + 0.014*"spelers" + 0.010*"clubs" + 0.010*"wedstrijd" + 0.008*"jongens" + 0.008*"voetbal" + 0.008*"almere" + 0.007*"veld" + 0.007*"wedstrijden"')*, where ‘almere’ is a specific city. I can imagine that general topics about sport and soccer would be more useful than very specific ones.

4.4.2 Text segmentation

The main problem for the text segmentation algorithm is that it finds too few topic boundaries. If we look at Table 5, we can see that the average number of boundaries that are found is too small, for both the transcripts and subtitles. There are no cases where the number of detected topics is larger than or equal to the true number of topics in that broadcast.

This problem has everything to do with the issue mentioned in Section 4.1; many segments get similar topic distributions ascribed to them, which makes the text segmentation algorithm merge those segments together. Even though the Jensen-Shannon method takes the entire probability distribution into account, the distributions don’t seem to have large enough probabilities to make a difference. I will analyze the role of the JS-parameter in this process.

When tuning the JS-parameter, two options were tested, namely 0.3 and 0.6. If the Jensen-Shannon distance is smaller than JS-parameter, the segments are merged together. This means that a small JS-parameter makes the threshold more difficult to pass, because the segments need to be more alike to produce a small Jensen Shannon distance. The best parameter was 0.3 (Section 4.2.2), meaning that a stricter parameter value worked better.

Given the problem of the algorithm detecting too few boundaries, it is quite interesting that the best parameter is 0.3. On first thought it makes sense that a stricter upper bound works better, forcing the segments to be more alike. On second thought it does not make sense given the problem of detecting too few boundaries, because a stricter threshold does not move the solution towards detecting more boundaries. In the end the JS-parameter presents a trade-off between putting boundaries after every segment which results in too many boundaries or merging every segment which results in too few boundaries. In that trade-off the parameter tuning favored the stricter parameter value, but the results still show that there are too few boundaries. This could be seen as a causal relationship, but during the parameter tuning all combinations were tested. So if a more lenient parameter value of 0.6 would have improved the results that would have been the best parameter value that was chosen there. A parameter value of 0.4 or 0.5 could theoretically present a better balance in the trade-off and improve the results. However, when testing these values on the test set without changing the other best parameter values that were chosen, no changes in results were found.

For both the transcripts and subtitles, the algorithm performed better on broadcasts that have fewer topic boundaries to detect. Most likely because the number of topic boundaries that are missed is less in those cases than broadcasts that have a lot of topic boundaries. Other patterns were not found.

5. Discussion

5.1 Challenges

To be able to answer the research question with more certainty, more news broadcasts should be labeled in future work. This larger sample size would make it possible to infer more conclusions and see more patterns, should they exist. A lesson I learned during this research is that sketching the experimental setup should be done as soon as possible. In this case that would have made labeling more data feasible.

The main challenge of this text segmentation algorithm is figuring out how to make the topic distributions more informative. In other words, how to make sure that a document is assigned a set of topics - each with a probability - that is informative enough to indicate what that document is about. A better topic distribution would help this text segmentation algorithm but could also give those distributions more purpose. Perhaps they could be used to label the different topic segments in a text or video as opposed to only marking the boundaries of those segments.

The topic distribution can firstly be made more informative by making the topics better, because better topics cause a better allocation of the topics to documents. This in itself is challenging because some words that are indicative for one topic, might get mixed up with other topics because it occurs a lot throughout the entire dataset. Those words bring the problems that stopwords bring to the story, but can't similarly be removed at once because they are not articles like 'the' or 'a' that do not matter for topics. Although these words are not stopwords, they usually are treated the same and are added to a "curated stopwords" list. Whether this is the right choice is being debated. For example, Schofield, A., Magnusson, M., & Mimno, D. (2017) argue that the removal of curated stopwords is superficial and that topic inference benefits little from it.

Another way of improving the text segmentation algorithm is by preprocessing the data more. The first cut size (Section 3.2.2) is now based on the best number of sentences as a result of parameter tuning. This first cut size could be more informed. For example, it could be based on changes in vocabulary between sentences, where we are only paying attention to the non-stopwords. With the present system there is a tradeoff between missing boundaries and informing the topic distribution (Section 3.2.2). A large first cut size and accompanying large segment is informative, but it could cause topic boundaries to be missed. A shorter segment provides more opportunity for topic boundaries to be placed, but it is less informative for making the topic distribution. With another way of deciding the first cut size, a better balance could be found in this trade-off.

In future works, these challenges could be a starting point for not only improving this text segmentation algorithm but also understanding the limitations and benefits of using topic models for solving similar problems.

This research started with the problem of a lack of apparent structure in videos. For archival institutes like the Netherlands Institute for Sound and Vision (S&V), this is a problem. Solving this problem can make all kinds of use cases possible. The most clear one being, jump-in points for topic transitions in videos. With this use case in mind, future work can focus on evaluation metrics for this particular case. In this research several evaluation metrics for text segmentation have been discussed, but it is possible that videos require a different form of evaluation. Jump-in points that are too late may be more problematic than jumping points that are too early. Developing different evaluation metrics can not only be helpful for figuring out how well a system works, it can also help materialize the priorities of that system, with a use case in mind.

5.2 Benefits

The developed text segmentation algorithm has various benefits.

The first benefit has been mentioned already (Section 1.1.1) and relates to topic modelling. Since topic modelling does not require labeling to learn the topics and topic distributions, human annotators are not needed to label large datasets. With ever-changing language, this unsupervised learning method can also withstand new labels and categories being invented. On a related note, some of the topics in the topic model qualitatively look really good. A few examples are in Table 8. Keeping in mind that topic models do not have any semantic information, this is quite a positive result.

The second benefit of this text segmentation algorithm is that the general idea of clustering broadcasts by cutting it and then merging it using a certain method is quite simple. The algorithm consists of clear different steps, which makes it possible to improve each step separately.

The third benefit is that the effect of parameter changes within the algorithm can be tested clearly, because it is not a black-box algorithm.

The fourth benefit is that the purpose of the algorithm could be extended. For example the topics could be used to label segments instead of only detecting the boundaries of the segments. Or the topics could be used to cluster segments across large datasets to connect different items to each other.

The final benefit is that the used packages and techniques - for example Mallet, Gensim, LDA and WindowDiff - are well known. This means that the “ingredients” of the algorithm - so to speak - are well documented and are being improved in active research communities.

6. Conclusion

For this research, topic segmentation in texts was used as a proxy for topic segmentation in videos. The main use case in mind was automatically providing a topic transition structure for videos, because it is difficult to quickly scan them and figure out where a new subject starts. The available data for this research was provided by the Netherlands Institute for Sound and Vision and consisted of 25,600 subtitles and automatic speech recognition (ASR) transcripts of the same Dutch news broadcasts. Topic models were used to figure out the topic transition positions in the speech transcripts and subtitles.

This thesis researched whether it is better to use ASR transcripts or subtitles when segmenting a video based on topics. The text segmentation algorithm that was used contains various parameters. The effect of these parameters and different techniques within the algorithm were tested on several types of evaluation metrics. Although no significant difference was found between the subtitles and transcripts on the performance, the research presents the challenges and benefits of using topic models for this use case. The main challenge of the text segmentation algorithm is figuring out how to make the topic distributions - the set of topics that each have a probability and are assigned to a document - more informative. The main benefit of the algorithm is the modularity of the components. The steps in the algorithm can each separately be improved (Section 5.2) and the influence of changes can be tested clearly as it is not a black-box algorithm.

Future research using a larger annotated dataset, can help shed light on the influence of the kind of data on the performance of text segmentation. This research, especially the sections that look at the topic distributions and influence on the text segmentation process, can help understand the topic modelling process and how it fits with this application.

In a broader sense, this research notes the challenges of automating a process that is quite easy for humans: recognizing when topics are changing. Although replicating the way humans recognize topic shifts sounds like a good starting point, it becomes completed quickly. Mainly because humans learn a lot by experience and translating that here means teaching a computer to associate words with labeled topics; a form of supervised learning that can be cumbersome. In artificial intelligence it can sometimes prove more useful to think about what computers *can* do that humans *cannot*. Instead of putting a camera on a driverless car to mimic eyes, we can give it all kinds of sensors that humans don’t have to assist driving. Similar thinking can be useful for this problem of recognizing topic shifts.

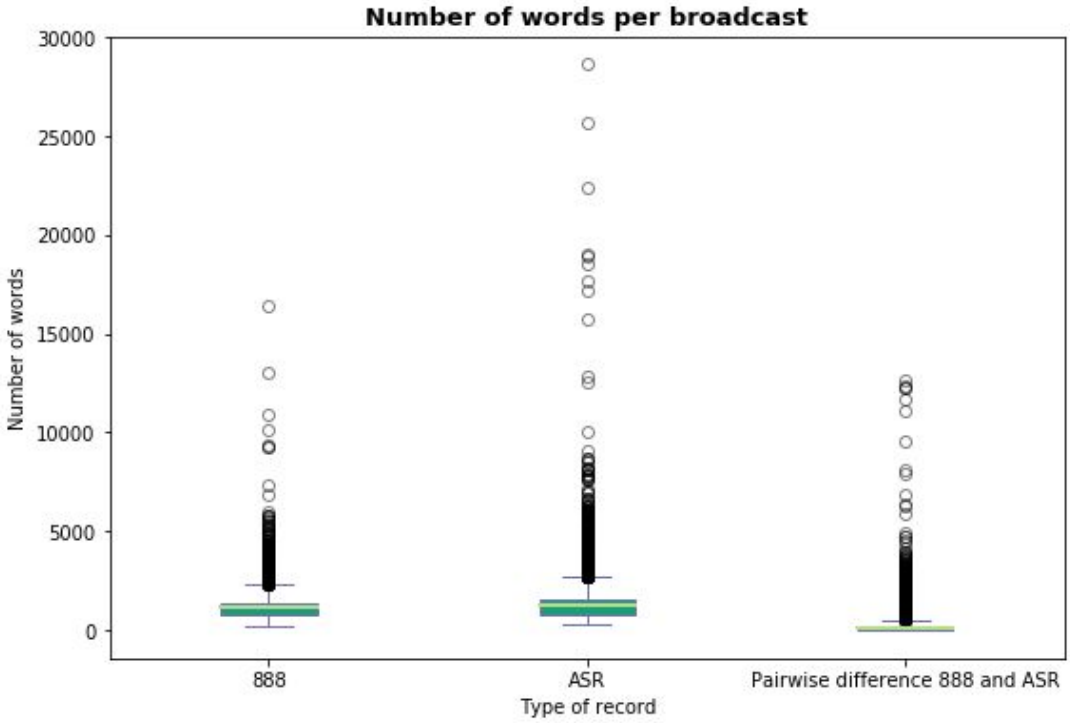
References

- Alghamdi, R., & Alfalqi, K. (2015). A Survey of Topic Modeling in Text Mining.
- Barzilay, R., & Elhadad, M. (1999). Using lexical chains for text summarization. *Advances in automatic text summarization*, 111-121.
- Beeferman, D., Berger, A., & Lafferty, J. (1997). Text segmentation using exponential models. *arXiv preprint cmp-lg/9706016*.
- Beeferman, D., Berger, A., & Lafferty, J. (1999). Statistical models for text segmentation. *Machine learning*, *34*(1-3), 177-210.
- Bestgen, Y. (2009). Quel indice pour mesurer l'efficacité en segmentation de textes. *Actes de TALN*, *9*.
- Blei, D., Ng, A., Jordan, M. (2003). Latent Dirichlet allocation. *J. Mach. Learn. Res. 3 , 993–1022.*
- Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77. https://doi.org/10.1145/2133806.2133826
- Bookstein, A., Kulyukin, V. A., & Raita, T. (2002). Generalized hamming distance. *Information Retrieval*, *5*(4), 353-375.
- Boyd-Graber, J., Hu, Y., & Mimno, D. (2017). Applications of Topic Models. *Foundations and Trends® in Information Retrieval, 11(2–3), 143–296.* https://doi.org/10.1561/1500000030
- Cao, J., Li, J., Zhang, Y., & Tang, S. (2007). LDA-based retrieval framework for semantic news video retrieval. In *International Conference on Semantic Computing (ICSC 2007)* (pp. 155-160). IEEE.
- Dror, R., Baumer, G., Shlomov, S., & Reichart, R. (2018). The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1383-1392).
- Elkan, C. (2012). Evaluating classifiers. San Diego: University of California.
- Fournier, C., & Inkpen, D. (2012). Segmentation similarity and agreement. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 152-161). Association for Computational Linguistics.
- Fournier, C. (2013). Evaluating text segmentation using boundary edit distance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1702-1712).
- *Gemoderniseerde stopwoorden lijst*. (z.d.). Geraadpleegd op 20 februari 2020, van https://eikhart.com/nl/blog/moderne-stopwoorden-lijst
- Hauptmann, A., & Smith, M. (1995). Text, speech, and vision for video segmentation: The informedia tm project. In *Proceeding of AAAI Fall Symposium Computational Models for Integrating Language and Vision, Boston*.
- He, X., Wang, J., Zhang, Q., & Ju, X. (2020). Improvement of Text Segmentation TextTiling Algorithm. In *Journal of Physics: Conference Series* (Vol. 1453, p. 012008).

- Hearst, M. A. (1993). *TextTiling: A quantitative approach to discourse segmentation*. Technical report, University of California, Berkeley, Sequoia.
- Hearst, M. A. (1994). *Multi-paragraph segmentation of expository text. Proceedings of* http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf https://doi.org/10.14569/ijacsa.2015.060121 https://doi.org/10.3115/981732.981734 *International Journal of Advanced Computer Science and Applications , 6 (1).*
- Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, *78*(11), 15169-15211.
- Karlgren, J. (1996). Stylistic variation in an information retrieval experiment. *arXiv preprint cmp-lg/9608003*.
- Misra, H., Yvon, F., Cappé, O., & Jose, J. (2011). Text segmentation: A topic modeling perspective*. Information Processing & Management , 47 (4), 528–544.* https://doi.org/10.1016/j.ipm.2010.11.008
- Ordelman, R., & van der Werff, L. (z.d.). *KALDI speech recognition toolkit*. Geraadpleegd op 6 mei 2020, van https://opensource-spraakherkenning.nl/
- Passonneau, R. J., & Litman, D. J. (1993). Intention-based segmentation: Human reliability and correlation with linguistic cues. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics* (pp. 148-155). Association for Computational Linguistics.
- Pevzner, L., & Hearst, M. A. (2002). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, *28*(1), 19-36.
- Repp, S., & Meinel, C. (2008). Segmentation of lecture videos based on spontaneous speech recognition. In *2008 Tenth IEEE International Symposium on Multimedia* (pp. 692-697). IEEE.
- Salton, G., Allan, J., & Buckley, C. (1993). Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 49-58).
- Scaiano, M., & Inkpen, D. (2012). Getting more from segmentation evaluation. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 362-366).
- Scaiano, M., et al. (2010). Automatic text segmentation for movie subtitles. In *Canadian Conference on Artificial Intelligence* (pp. 295-298). Springer, Berlin, Heidelberg.
- Schofield, A., Magnusson, M., & Mimno, D. (2017). Pulling out the stops: Rethinking stopword removal for topic models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 432-436).
  Semantic Analysis: A Road to Meaning. Laurence Erlbaum
- Steyvers, M., and Griffiths, T. (2007). Probabilistic topic models. *In T. Landauer, D McNamara, S. Dennis, and W. Kintsch (eds), Latent the 32nd annual meeting on Association for Computational Linguistics.*

Appendix

Section A: number of words per broadcast for different kinds of data with outliers



Section B: additional information about manual labeling

The subtitles and ASR datasets contain transcripts of the same broadcasts. This means the topic boundaries in the subtitles are the same as the boundaries in the accompanying speech transcript. When labeling the ASR data, I will look at the subtitle data, to make sure that the boundaries are put in the same places for the same broadcasts.

It is necessary to label the data myself, because the ASR transcripts and subtitles have too many differences. For example, a topic boundary in the subtitle might be between sentence 1 and sentence 2. In the speech transcripts these sentences are often merged into a single sentence, in which some words might be missing and others might exist that can't be found in the subtitles. If a topic boundary is stuck between two wrongly merged sentences in the speech transcripts, the question is where I would put the boundary. If I would put it in around the merged sentence as opposed to within the sentence the downside is that the boundary is in fact slightly off. It is supposed to be within the merged sentence. If I would put the boundary within the merged sentence there is another downside. Since the text segmentation algorithm uses the sentences in the text to determine where a boundary should be the algorithm could never find a boundary within a sentence. Compared to a fraction of the boundaries being off by a maximum of half a sentence, this is a bigger downside. That is why I will put the boundary around the merged sentence, should those cases appear.

Section C: list of stopwords

The general stopwords:

a aan aangaande aangezien achter achterna aen af afd afgelopen agter al aldaar aldus alhoewel alias alle allebei alleen alleenlyk allen alles als alsnog altijd altoos altyd ander andere anderen anders anderszins anm b behalve behoudens beide beiden ben beneden bent bepaald beter betere betreffende bij bijna bijvoorbeeld bijv binnen binnenin bizonder bizondere bl blz boven bovenal bovendien bovengenoemd bovenstaand bovenvermeld buiten by daar daarheen daarin daarna daarnet daarom daarop daarvanlangs daer dan dat de deeze den denk der ders derzelver des deszelfs deszelvs deze dezelfde dezelve dezelven dezen dezer dezulke die dien dikwijls dikwyls dit dl doch doen doet dog door doorgaand doorgaans dr dra ds dus echter ed een eene eenen eener eenig eenige eens eer eerdat eerder eerlang eerst eerste eersten effe egter eigen eigene eigenlijk elk elkanderen elkanderens elke en enig enige enigerlei enigszins enkel enkele enz er erdoor et etc even eveneens evenwel ff gauw gaat gaan ging ge gebragt gedurende geen geene geenen gegeven gehad geheel geheele gekund geleden gelijk gelyk gemoeten gemogen geven geweest gewoon gewoonweg geworden gezegt gij gt gy haar had hadden hadt haer haere haeren haerer hans hare heb hebben hebt heel heeft hele hem hen het hier hierbeneden hierboven hierin hij hoe hoewel hun hunne hunner hy ibid idd ieder iemand iet iets ii iig ik ikke ikzelf in indien inmiddels inz inzake is ja jaar je jezelf jij jijzelf jou jouw jouwe juist jullie kan klaar kon komen komt konden kijken krachtens kunnen kunt konden laetste lang land later liet liever like m maar maken maeken maer mag martin me mede meer meesten men mensen menigwerf met mezelf mij mijn mijnent mijner mijzelf min minder mmm misschien mocht mochten moest moesten moet moeten mogelijk mogelyk mogen my myn myne mynen myner myzelf na naar nabij nadat naer net niet niets nimmer nit no nou noch nog nogal nooit nr nu o of ofschoon om omdat omhoog omlaag omstreeks omtrent omver onder ondertussen ongeveer ons onszelf onze onzen onzer ooit ook oorspr op opdat opnieuw opzij opzy over overeind overigens p pas pp precies pres prof publ reeds rond rondom rug s sedert sinds sindsdien sl slechts sommige staat spoedig st steeds sy t tamelijk tamelyk te tegen tegens ten tenzij ter terwijl terwyl thans tijdens toch toe toen toenmaals toenmalig tot totdat tusschen tussen tydens u uur uit uitg uitgezonderd uw uwe uwen uwer vaak vaakwat vakgr van vanaf vandaan vanuit vandaag vanwege veel veeleer veelen verder verre vert vervolgens vgl vol volgens voor vooraf vooral vooralsnog voorbij voorby voordat voordezen voordien voorheen voorop voort voortgez voorts voortz vooruit vrij vroeg vry waar waarom wanneer want waren was wat we weg wege weet wegens weinig weinige wel weldra welk welke welken welker werd werden werdt wezen weten wie wiens wier wierd wierden wij wijzelf wil wilde willen worden wordt wy wyze wyzelf zal ze zeer zei zeker zeggen zegt zekere zelf zelfde zelfs zelve zelven zelvs zich zichzelf zichzelve zichzelven zie zig zij zijn zijnde zijne zijner zo zo'n zoals zodra zien zommige zommigen zonder zoo zou zoude zouden zoveel zowat zulk zulke zulks zullen zult zy zyn zynde zyne zynen zit zat zaten zitten zyner zyns

The stopwords for this particular dataset:

vinden vind vond staan stonden stond ga sta echt denk weer vind vindt blijkt goed dag krijgt krijgen vorig vorige volgend volgende nederlandse een twee drie vier vijf zes zeven acht negen tien nederland man vrouw iedereen vanavond gisteren morgen goed natuurlijk praat praten inderdaad meestal hoor beetje allemaal helemaal laten graden gezegd duidelijk allemaal nee goed laten dank ligt blijft krijgt praat programma live ondertiteling dag achterlopen ondertitel ondertiteld ondertitelt 888 tt888 www nl praten tt tt888reacties(a)omroep mee dit werd voor meer info: info npo nos journaal staan vinden terug kwam ziet lig zag lag