Text Segmentation on Datasets from Different (Dutch) Modalities[⋆]

Giorgie I. Goiati[1] – Student No.: 2600183
Supervisors: Benno Kruit[1] and Justo van der Werf[2]

[1] Vrije Universiteit, Amsterdam, The Netherlands
[2] RTVMonitor, Amsterdam, The Netherlands

Abstract. Separating a text into topically coherent sections aids in its comprehension not only for humans, but for computers in Natural Language Processing (NLP) tasks as well. As such, a multitude of methods for automatic Text/Topic[3] Segmentation (TS) have already been proposed, featuring an array of unsupervised and supervised techniques. However, there is one common thread throughout these approaches which we seek to scrutinise, which is the reliance on texts derived from *English, formal, written* sources. Considering the differences in language between English and Dutch (and their available pre-trained models), writing and speech, and different contexts, we introduce three new Dutch datasets for TS tasks. These datasets are comprised of texts from contrasting sources: Wikipedia documents, concatenated news articles, and automated transcriptions from TV/radio broadcasts. Together with a reference English Wikipedia dataset, we implemented a selection of TS algorithms to test each of the datasets on, to illustrate the differences between datasets, to what extent these differences affect TS performance, and how robust the TS algorithms remain. Additionally, Dutch supervised models trained on a singular dataset, were tested on each of the Dutch datasets, investigating whether datasets from certain modalities make for better general training data. Our findings indicate significant differences in performance per dataset, across each TS method: the more the data diverges from the typical English Wikipedia TS dataset, the more difficult it becomes to segment. Moreover, the new types of data highlight the strengths and shortcomings of the tested algorithms and used metrics, pointing towards avenues for improvement in future models.
The datasets and the code for creating the datasets, running the models and doing the experiments, is available at:
https://github.com/DoubleGio/text_segmentation.

Keywords: Natural Language Processing · Text Segmentation · Dutch · Datasets

[⋆] Supported by RTVMonitor.
[3] Topic and Text (Segmentation) are used interchangeably.

1 Introduction

Whether we read lengthy texts, listen to the news or watch a documentary, authors tend to carefully arrange the material into (sub)sections in order to make it easier for the audience to follow and understand. This does not only supports humans, but computers as well, since many downstream Natural Language Processing (NLP) tasks, such as summarization, passage extraction, discourse analysis, and more, all benefit from having a well-structured text. As a concrete example: to summarise a text, a basic strategy often employed by humans is to build the summary out of the first sentences of each paragraph (which are generally important ones). A computer could be taught to do the same, but this requires the text to be pre-segmented of course. This is where the task of Topic/Text[4] Segmentation (TS) comes into play, which aims to automate the process of identifying topic boundaries.

Numerous solutions have already been proposed in the last two and a half decades, each following a general two-step approach of encoding the text first, and then making the boundary predictions based on that encoding. Early, unsupervised methods had to rely on heuristics and statistics to make their predictions – there was a lack of large-quantity training data and the hardware to support it. These involved techniques such as measuring lexical cohesion between (sets of) sentences [5,8], using probabilistic/distributional models [6] or graph [7] based methods. More recent developments showed that these unsupervised methods can be improved by utilising modern, pre-trained language models (i.e. BERT) [15].

Nevertheless, supervised models show the most promising results. With some architectures training dedicated text embedding models in conjunction, most aim their attention fully towards the segmenting scheme, for which a variety of methods have been proposed as well. Most include some variation of a Recurrent Neural Networks (RNNs) – Bidirectional Long Short-Term Memory (Bi-LSTM) networks are particularly popular – while the latest methods are beginning to employ attention mechanisms and Transformers, which show considerable promise.

These supervised models have been made possible by the creation of several large-quantity, labelled datasets, but all these datasets seem rather monotonous: they fail to cover every possible type of text. Most, if not all TS training datasets are composed of formal, written, English texts, from sources such as news articles, Wikipedia documents or textbooks; none featured audiovisual sources *as training data* – some used audiovisual content for evaluation purposes [6,14]. Considering the numerous differences between spoken and written language [1], we believe that this disparity might negatively impact accuracy when segmenting audiovisual content, which does not feature clear segments to begin with, making it one of the main areas in which topic segmentation can be useful.

RTVMonitor is a Dutch company that specialises exclusively in audiovisual content and its processing. As such, when attempting to extract or summarise matter from radio broadcasts, television talk-shows or podcasts, the relevant

[4] Topic and Text (Segmentation) can be used interchangeably.

segments need to be extracted from continuous Automatic Speech Recognition (ASR) transcripts. Fortunately, RTVMonitor has a dataset available, comparable in size to those used in other works, of purely ASR transcripts, with timestamps indicating the start and end of segments. We hypothesise that this dataset is considerably more difficult for models to segment, not only due to the different, less coherent content, but also because transcripts produced by ASR systems are still far from perfect. At the same time, this unique data could make the trained models better equipped to work with these more difficult texts.

A diverse assortment of effective (un)supervised models exist already; instead **this research investigates the significance of differing modalities of (Dutch) datasets in Text Segmentation tasks, across a variety of methods**. To carry this out we introduce three new, Dutch TS datasets[5], each consisting of texts from a unique source. The texts of these datasets/sources range from more to less concise & formal: Wikipedia documents, news articles and transcriptions from audiovisual media. Additionally, we evaluate on the English *Wiki-727K* [9] dataset, to compare languages and verify performance. Our experiments involve testing these datasets on a selection of unsupervised and supervised algorithms; the results showcase how the type of texts differ in terms of difficulty and structure, which implementation is most effective and robust, and in what aspects/domains the different implementations manage to (under)perform. Based on these results we propose solutions and suggestions for future implementations, especially for those looking to work with less formal, more challenging types of input.

2 Related Work

Plenty of different strategies and methods can be found throughout the numerous existing TS implementations, but each shares the same two-level structure:

1. **Text Encoding**: Transforming an unsegmented text into a representation the program can understand and work with.
2. **Boundary Identification**: Predicting where sections start/end based on the encoded text.
   - Boundaries can be placed in between paragraphs (if available), (pseudo-)sentences, words, or even characters. The choice depends on the task at hand, the methodology, and the language and lay-out of the texts to be processed [11].

The task of boundary identification is of course core to TS, but this does not make the first level trivial. With this in mind, one can make the distinction between algorithms that use supervised methods to implement (one of) these levels, and those that remain fully unsupervised.

Supervised approaches have become more accessible and feasible only recently, hence why the first TS implementations, dating back to the late 90's, were

[5] One of which is property of RTVMonitor and not publicly available.

all unsupervised. These earliest approaches, such as the *TextTiling* [8] and *C99* [5] algorithms, exploit lexical cohesion statistics: they look for changes in patterns of lexical repetition between sentences/segments. Later methods improve matters by providing more of a statistical backbone to these purely heuristic-driven approaches; *BayesSeg* [6] and *TopicTiling* [13] use Bayesian language and LDA topic models, respectively, to represent the input texts, allowing for more informed topical cohesion calculations.

However, with the advent of pre-trained language models, such as Word2vec or BERT, unsupervised algorithms have mostly moved away from statistical modelling. These new language models have made the text encoding process simpler, while providing more semantic information at the same time. *GraphSeg* [7] is one of these newer methods: it uses GloVe word vectors as main encoding tool, and uniquely builds a fully connected *semantic relatedness graph* (with sentences as nodes and the relatedness to another sentence as vertex), from which it gets to form sections using graph calculations. Even though most research in recent years has been focused on seemingly superior supervised approaches, new unsupervised algorithms do still show up, incorporating the latest pre-trained language models for competitive results. E.g. Solbiati et al. [15] upgraded the TextTiling algorithm by basing its similarity score calculations on BERT embeddings, demonstrating significant performance gain relative to the original.

The first supervised methods arose paired with new, large enough datasets to train with; commonly used *training* datasets have been made out of concatenated news articles [14], extracted Wikipedia documents [2,9], or textbook chapters [3]. Early solutions focused on LSTMs as trainable networks; Sehikh et al. [14] decided to measure the topic cohesion, similar to TextTiling, around *words* (encoded through word embeddings) using the hidden layer activations for each direction of a Bi-directional LSTM (Bi-LSTM) network. *SECTOR* [2] instead looks to segment on a sentence level (in addition to taking on topic classification) and tests a variety of unsupervised sentence encoding methods, before adopting the segmentation approach of Sehikh et al. [14]. *TextSeg* [9] runs fully supervised and introduces another Bi-LSTM to create sentence embeddings; it also forgoes calculating topical cohesion directly and lets its segmenting network make the predictions directly. Attention/Transformer based methods have also started to show up; Badjatiya et al. [3] and Xing et al. [18] used attention layers to build upon the preceding RNN-based approaches; others [10,16] based their algorithm fully on transformers for both encoding and segmenting layers.

These state-of-the-art supervised methods have shown better performances than unsupervised ones, though these results mostly pertain to the domain in which their training data resides; supervised models trained on written data, can be approximated and even outperformed by unsupervised ones in tests on transcripts [14,15]. Aside from being more challenging and unique in structure, our Dutch audiovisual dataset can function as a training dataset derived from a completely different domain, allowing us to closely examine how training on different types of texts affects text segmentation performance.

3 Methodology

In order to run experiments to get to some answers, we first had to construct the three new, Dutch datasets (to accompany to the existing English one) and implement the algorithms we wanted to test with, ensuring that everything was compatible with each other. We ended up implementing three unsupervised and three supervised algorithms, chosen for being commonly referenced or state-of-the-art, relatively straightforward to implement, well documented, and comparable (e.g. by being built upon one another). Ensuring compatibility was not the only challenge we faced however; consider the numerous amount of possible configurations: $4\ datasets \times 6\ algorithms = 24\ tests$, and that is excluding the out-domain experiments; it was imperative to optimise each the algorithms (especially older ones) in terms of speed as well.

3.1 Creating new text segmentation datasets

Text Segmentation can be applied to any text of reasonable length, but for a corpus to be suitable for evaluation and training purposes, it must be pre-segmented and sufficiently large. Some datasets, like WikiSection [2] or Wiki-727K [9], include additional information such as section titles, labelled granularities for nested sections, or domain specific matters (e.g. timestamps for transcriptions), but for our use-case, the datasets only featured the essentials: the text and the boundaries.

Table 1 gives a concise overview of each of the datasets we used for our experiments, describing its contents, what the ground truth segmentation is based on, how many documents the dataset contains, and the presumed difficulty to segment. With the Wiki-727K dataset [9] as base (renamed to *EN-Wiki* for this paper), each of the other datasets was created to resemble it. *NL-Wiki* followed a creation method similar to its English counterpart, but simplified (no need for section granularities). The *NL-News* dataset was based on newspaper articles, which generally do not come with sections or multiple headers. To create pre-segmented texts, we concatenated together 2-5 articles of appropriate lengths.

Table 1: Dataset descriptions. N refers to the amount of documents in the dataset.

	Language - Name	**Contents**	**Segmented on**	N
Easy	*EN-Wiki (Wiki-727K)*	Wikipedia documents	Headers	727 746
	NL-Wiki	Wikipedia documents	Headers	390 709
	NL-News	Newspaper articles	Concatenations	199 148
	NL-AuVi_C (concat)	Radio/TV transcriptions	Concatenations	28 217
Hard	*NL-AuVi_N (normal)*	Radio/TV transcriptions	Start/end relevant	98 530
		segment		

The *NL-AuVi* (Audio-Visual) dataset is a unique case, in that the ground truth segmentations are incomplete in a sense. This is due to the way in which RTVMonitor processes their transcriptions:

1. RTVMonitor continuously looks for mentions of certain keywords, which their customers are interested in.
2. When the keyword appears, 5 minutes of recording before and after its appearance gets transcribed and saved.
3. Finally, the single segment relevant to the keyword is marked by hand.

As a result, each document contained merely a single "true" segment, surrounded by a significant amount of filler (perceived as two, incoherent sections by the algorithms); this raised the question of how this unconventional lay-out might affect performance. To investigate, and in an attempt to create a better dataset, we introduced another version of the dataset (denoted *NL-AuVi_C*) consisting of solely true segments, concatenated together in the same fashion as we did with *NL-News*. To figure out which version was the most useful, we set them up against eachother in experiments, further described in Section 4.1.

Figure 1 showcases the characteristics of each of the datasets and how they differ from one another (see Appendix A.2, Table 6 for exact averages). Particularly clear becomes the decline in text "quality" when moving from the (presumed) easier to the more difficult datasets – from *EN-Wiki* to *NL-AuVi_N*: sections contain increasingly more sentences, of shorter lengths. This pattern correlates with the decline in formality of the sources used, in the sense that Wikipedia documents are more dense and informative than news articles, which again are more formal than the spoken language in the radio/TV transcriptions. The decrease in text quality when moving from the English to Dutch Wikipedia corresponds with the difference in "Wikipedia Depth"; a rough indicator for quality, as described by Wikipedia themselves [17]. We hypothesise that a lower text quality makes for more difficult to segment texts.

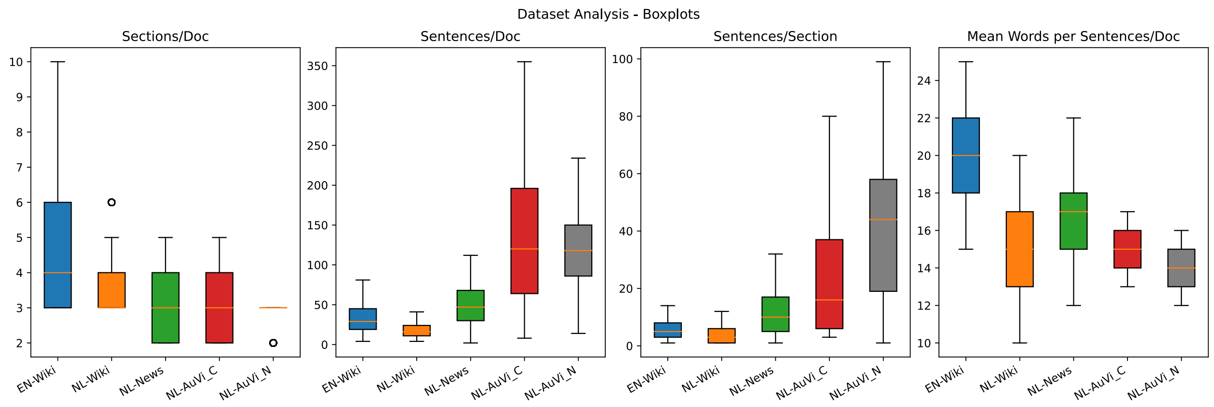


Fig. 1: Boxplots showcasing several characteristics for each of the datasets.

3.2 Unsupervised approaches

Recall the two-level structure we mentioned earlier in section 2: **Text Encoding** and **Boundary Identification**. We describe each of the selected algorithms with this framework in mind. Refer to Appendix A.1 for hyperparameter settings.

TextTiling (1997) [8]. One of the earliest TS methods, that still manages to remain reasonably effective; often used as performance baseline.

- **Text Encoding**: Tokenizes the text into equal-sized units of size w (approximates average sentence length), called *token-sequences*, to allow for better comparisons; also filters out stopwords (which do still contribute to token-sequence size).
- **Boundary Identification**:
  1. Compares adjacent blocks of token-sequences of size k (approximates average section length) for overall lexical similarity, resulting in the similarity score for the *token-sequence gap*:

     $$score(i) = \frac{\sum_t w_{t,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}} \tag{1}$$

     Where t ranges over all the registered tokens (excl. stopwords), and $w_{t,b}$ is the weight (frequency within a block) assigned to term t in block b.
  2. These similarity scores are then smoothed and given a depth score, which corresponds to the depth of the valley (if one occurs); a bigger depth score, a "deeper" valley, means a stronger cue the subtopic changed at the token-sequence gap:

     $$ds(i) = score(peak_{left}) + score(peak_{right}) - score(i) \times 2 \tag{2}$$

  3. Boundaries are drawn if the depth score exceeds the measure $LC : \overline{s} - \sigma$ or (the more conservative) $HC : \overline{s} - \sigma/2$, with $\overline{s}$ as average and σ as standard deviation of the depth scores.

One problem with the original TextTiling implementation is that it assumes the input text features paragraphs, so it may assign the boundaries to the closest paragraph start – a necessity since the boundaries correspond to token-sequence gaps, as opposed to gaps between sentences/paragraphs. However this is a rather naive assumption to make, since not every text is well-structured like this (e.g. our NL-AuVi texts). As a workaround, our implementation[6] assigns the gap boundary to the closest sentence start instead.

[6] Adapted from the NLTK TextTiling module.

GraphSeg (2016) [7]. This method builds a *semantic relatedness graph* in which nodes denote sentences and edges are created for pairs of semantically related sentences, allowing for segments to be identified using graph theory.

- **Text Encoding**: Uses pre-trained GloVe word embeddings to encode all words.
- **Boundary Identification**:
  1. Starts by measuring the semantic relatedness between all pairs of sentences. The method for calculating this is elaborate, but to summarise, for a given sentence pair (S_1, S_2), with the set of non-stopword pairs in the optimal (most similar) alignment as $A = \{(w_1, w_2) \mid w_1 \in S_1 \land w_2 \in S_2\}$:

     $$sr(S_1, S_2) = \sum_{(w_1,w_2)\in A} \cos(emb(w_1), emb(w_2)) \cdot min(ic(w_1), ic(w_2)) \quad (3)$$

     This calculates the sum of the cosine similarity of all word pairs, weighted by the *information content* $ic(w)$, which is based on the relative frequency of w in some large corpus. To deal with discrepancies in sentence sizes, the scores are also normalised.
  2. Creates the semantic relatedness graph by having all sentences become nodes in graph G, and adding an edge for each sentence-pair where the relatedness score exceeds some threshold value.
  3. Employs the following graph computations to create the set of segments:
     (a) Compute the set Q of all maximal cliques of G using the *Bron-Kerbosch* algorithm.
     (b) Create initial segmentation SG by merging adjacent sentences found in at least one maximal clique $q \in Q$.
     (c) Merge adjacent segments for which there is at least one clique $q \in Q$ containing a sentence from both segments:
     $SG = \{(sg_i, sg_{i+1}) \mid \exists q \in Q((S_{sg_i} \cap S_q \neq \emptyset) \land (S_{sg_{i+1}} \cap S_q \neq \emptyset))\}$
     (d) Finally, given the minimal segment size, merge too small segments with the most semantically related adjacent segment.

BertTiling* **(2021) [15].** This method is based on the TextTiling algorithm, following its outline while using more up-to-date techniques.

- **Text Encoding**: Computes sentence embeddings, by averaging over the second to last hidden layer from a pre-trained BERT model.
- **Boundary Identification**:
  1. Compares adjacent blocks of k (approximates average section length) sentences, by calculating the cosine similarity of the block embeddings (obtained through a max-pooling operation). This functions as the similarity/gap score of TextTiling (Equation 1).

* Named by us.

2. Unchanged from the original TextTiling, these similarity scores are smoothed and given a depth score (Equation 2); a bigger depth score means a stronger cue the subtopic changed at the block gap.
3. Boundaries are drawn if a local depth score maxima $lm \in LM$ exceeds some set threshold τ multiplied with the highest depth score.

$$LM = \{ds(i) \mid ds(i-1) < ds(i) > ds(i+1)\}$$
$$boundaries = \{lm \in LM \mid lm > (\tau * \max(ds))\}$$

The method outlined here follows the implementation taken from the code that the authors provided, as opposed to following the specifics defined by the paper – the final boundary identification step as described by the paper performed worse than what was actually happening inside the code.

3.3 Supervised approaches

TextSeg (2018) [9]**.** The first (to our knowledge) fully supervised TS algorithm; they also introduced the *Wiki-727K* (EN-Wiki) dataset to make training neural networks feasible to begin with. All code, for creating the datasets and the models, is fully open source, which allowed us to relatively easily create a Dutch Wikipedia dataset and copy the structure for the creation of the other datasets.

- **Text Encoding**:
  1. Uses pre-trained Word2vec embeddings to encode all words.
  2. For each sentence, creates an embedding by feeding the word embeddings (belonging to the sentence) to the lower-level Bi-LSTM network (the sentence encoder) and max-pooling over its output.
- **Boundary Identification**:
  1. Feed sentence embeddings to the upper-level Bi-LSTM network (the boundary predictor).
  2. Connect to a fully connected layer to obtain an output vector h of size $n \times 2$: for n sentences and the two output classes [non-boundary, boundary].
  3. The predictions are obtained by applying a *softmax* function to the boundary class outputs (the second column of h).

The model trains by minimising the *cross-entropy loss*; during inference a minimal P_k + *WindowDiff* (TS specific metrics, see section 5.1) is looked for instead. A greedy strategy is used to create segments, i.e. a segment ends when p_i is greater than threshold τ, which is optimised during each validation round.

TextSeg+[⋆] (2020) [18]. Directly iterates on the TextSeg algorithm described above, with the focus on improving context modelling and robustness (unchanged parts are coloured grey).

- **Text Encoding**:
  1. Uses pre-trained Word2vec embeddings to encode all words.
  2. For each sentence, creates an embedding by feeding the word embeddings (belonging to the sentence) to the lower-level Bi-LSTM network (the sentence encoder) and applying an attention mechanism [19] on its output.
  3. Additionally, to enhance generality, BERT sentence embeddings are concatenated to the embeddings from the sentence encoder.
     The BERT sentence embeddings are computed by averaging over the second to last hidden layer from a pre-trained BERT model.
- **Boundary Identification**:
  1. Feed sentence embeddings to the first upper-level Bi-LSTM network. Its outputs are not used to predict boundaries directly, but for use in the next two steps.
  2. An auxiliary task (an additional loss to be minimised) predicts the consecutive sentence-pair coherence by using the output from step (1):
     - The ground truth is obtained by labelling pairs as 1 if the sentences are from the same segment and 0 otherwise.
     - Coherence is calculated by taking the sigmoid of the cosine similarity between embeddings e_i and e_{i+1} (where $\overrightarrow{h}$ and $\overleftarrow{h}$ denote the Bi-LSTM forward and backward output states):

$$\begin{aligned} e_i &= \tanh(W_e(\overrightarrow{h_i} - \overrightarrow{h_{i-1}}) + b_e) \\ e_{i+1} &= \tanh(W_e(\overleftarrow{h_{i+1}} - \overleftarrow{h_{h+2}}) + b_e) \end{aligned} \tag{4}$$

     - Cross-entropy loss formulates the objective of the auxiliary task.
  3. *Sentence-Level Restricted Self-Attention*, encourages the model to absorb more information from nearby sentences:
     - Use the obtained output from step (1) to compute the similarities between the current sentence i and its neighbours N within a set range S:

$$\text{sim}_{i,j} = W_a[h_i; h_j; (h_i \cdot h_j)] + b_a \mid j \in N \tag{5}$$

     - With a as the attention weights, the *local context embedding* c (the weighted sum of all sentence outputs) is calculated for each sentence:

$$c_i = \sum_{j=1}^{S} a_{i,j} h_j + \sum_{j=1}^{S} a_{i,-j} h_j \tag{6}$$

  4. The original output from step (1) and the corresponding local context embedding are concatenated and input to a final Bi-LSTM (the boundary predictor).

⋆ Named by us.

5. Connect to a fully connected layer to obtain a $n \times 2$ vector, for n sentences and the two output classes [non-boundary, boundary].
6. The predictions are obtained by applying a *softmax* function to the boundary class (second column) outputs.

The model trains by minimising the *cross-entropy loss*; during inference a minimal P_k + *WindowDiff* (TS specific metrics, see section 5.1) is looked for instead. A greedy strategy is used to create segments, i.e. a segment ends when p_i is greater than threshold τ, which is optimised during each validation round.

Transformers^2 (2021) [10]. Fully Transformer-based; the paper featured multiple versions of the model – using different pre-trained data and experimenting with topic classification as well – we implemented the most effective version, best suited for our experiments: using BERT and excluding the single sentence embeddings and topic classification.

– **Text Encoding**: Generate sentence embeddings by taking the `[CLS]` token from a pre-trained BERT model with pairwise-sentence inputs.
– **Boundary Identification**:
  1. Inject the obtained sentence embeddings with their positional encoding and input them into a number of Transformer encoder layers.
     • These Transformer encoders layers can take a mask as input, which tends to improve performance. The paper briefly mentioned masking out 70% of the non-boundary sentences for training, but due to a lack of time and no clear description, we only implemented a standard *square subsequent mask*[7].
  2. Connect to a fully connected layer to obtain a $n \times 2$ vector, for n sentences and the two output classes [non-boundary, boundary].
  3. Predictions are obtained by applying *argmax* column-wise (whichever class is deemed most likely).

The model trains by minimising the *cross-entropy loss*; during inference a minimal P_k + *WindowDiff* (TS specific metrics, see section 5.1) is looked for instead.

4 Experiment Setup

Our investigation concerns the significance of different kinds of corpora as input for a Text Segmentation task. Section 3.1 already demonstrated that the datasets differ from one another; the following experiments seek to give us a more precise idea of *how* these differences manifest themselves in a TS task specifically, and to find new, unforeseen differences between the datasets and algorithms.

[7] See PyTorch implementation.

4.1 Should we concatenate NLAuVi dataset or not?

Before carrying out all the main experiments – to reduce the total number of them to be carried out – we wanted to figure out which layout made for a better dataset in the case of the NL-AuVi dataset: unaltered or concatenated? As discussed in Section 3.1, the unusual structure of the RTVMonitor data, with each document only featuring a single "true" segment, could be detrimental to its viability as training data. We introduced a presumably better version of the dataset, NL-AuVi_C, whose documents consisted of concatenations of relevant segments only. To compare this dataset to the original, NL-AuVi_N, we performed these preliminary tests:

– Both datasets were tested on the unsupervised methods.
– Two Transformers[2] (since it is the most state-of-the-art) models were trained, one on the unaltered data and the other on the concatenations. These trained models were then tested for performance on the Dutch datasets, as we are interested in its performance on real-world data.

The rest of the experiments were then to be performed including the better of the two NL-AuVi datasets.

4.2 How are the unsupervised algorithms affected by differing datasets?

The experiments with the unsupervised algorithms were straightforward: we tasked each algorithm with segmenting each of the test sets. These results serve as baselines to compare performance of the supervised algorithms against, and also should give a good idea of which method was most effective and why. The easier to interpret nature of unsupervised algorithms could also give further insight into how the datasets are syntactically or semantically structured.

4.3 How are the supervised algorithms affected by differing datasets?

Besides looking at the performance of the supervised methods, we also wanted to see which dataset is best suited as training data. We first trained each algorithm on every individual dataset, ending up with 3×4 trained models. Each model was then tested on the dataset it was trained on, but also on compatible (so for Dutch models only) outer-domain test sets.
Additionally, we wanted to explore the potential value of using a mixed training dataset, containing documents from the NL-Wiki, NL-News and NL-AuVi datasets. To test its viability, we trained new models on this mixed dataset and tested its performance on each of the Dutch test sets.

5 Results

5.1 Text Segmentation Metrics

Before getting into the results, we ought to clear up the metrics used, because TS tasks are generally speaking not evaluated with standard measures for classification tasks, like precision, recall, or their relatives. This is because these standards disregard *near misses*, which in a TS task should count for something. Take the segmentations in Figure 2 for example, prediction A is clearly much better than prediction B, but the precision (the fraction of boundaries identified which are correct) and the recall (the fraction of true boundaries identified) are for both 0%.

To tackle this issue, the P_k score was introduced by Beeferman et al. [4]. A probabilistic error metric, representing the chance that a randomly chosen pair of words/sentences of a set distance apart is incorrectly classified. It is calculated by sliding a window of size k (usually set to half the average section length) across the predictions, and counting the amount of times there is a mismatch with the ground truth within the window; this is then divided by the number of measurements taken to end up with a number between 0 and 1.

Aside from the P_k score, we also use the *WindowDiff* metric (shortened to *WD*), which was introduced by Pevzner and Hearst [12], who identified a number of issues with the P_k score and tried to address them. Without going into too much detail, these issues were:

- Penalising false negatives more than false positives.
- Ignoring the number of boundaries within a window.
- Sensitivity to variations in segment size.
- Penalising near-misses too much.

The proposed solution was simple: instead of counting the amount of times an error occurs within a window, the amount of times the difference between the number of reference and hypothesis boundaries within a window is greater than 0 is counted instead. More formally, with b_i as the number of boundaries within window i:

$$\mathrm{WD} = \frac{\sum_{i=0}^{N-k}[|b_i(\mathrm{ref}) - b_i(\mathrm{hyp})| > 0]}{N-k} \tag{7}$$

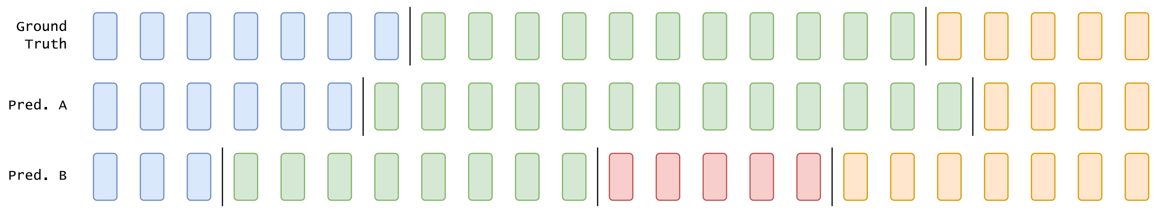


Fig. 2: Example segmentations. Each rectangle represents a sentence, different colours indicate separate sections, and the black lines indicate boundaries.

Besides P_k and WD, we also provide accuracy ($Acc. = \frac{\text{correct classifications}}{\text{all classifications}}$) as a more intuitive metric, though it functions mostly as reference – the focus should remain on the TS specific metrics.

5.2 Should we concatenate NLAuVi dataset or not?

Table 2a shows the results from the unsupervised tests. The scores between the normal and concatenated test sets are almost equal across each of the algorithms – the concatenated test set was only slightly easier to segment for BertTiling. It seems that concatenation does not affect the difficulty much, though this might also be due to the NL-AuVi corpus being quite challenging to segment in the first place, since the P_k and WD scores are high across the board.

The supervised test results, as seen in Table 2b, paint a clearer picture: training on the concatenated dataset does not improve performance for any of the real-world, unaltered documents. This model only performs better on the *NL-News* test set, pointing towards it picking up on the structure of concatenated texts, more than gaining a good understanding of the semantic content. Consequently, we stuck to the unaltered NL-AuVi dataset for the rest of the tests.

Table 2: The tests comparing the normal vs. concatenated NL-AuVi datasets.

(a) Results from testing the unsupervised methods on the NL-AuVi datasets.

Test sets	**NL-AuVi_N**			**NL-AuVi_C**		
Method	P_k	WD	$Acc.$	P_k	WD	$Acc.$
TextTiling	**0.51**	0.97	0.81	0.59	**0.92**	0.81
GraphSeg	**0.64**	0.97	0.80	0.72	0.97	0.79
BertTiling	0.50	0.60	0.96	**0.48**	**0.55**	0.94

(b) Results from testing the Transformers^2 models on the Dutch datasets.

Test sets	**NL-Wiki**			**NL-News**			**NL-AuVi_N**		
Trained on	P_k	WD	$Acc.$	P_k	WD	$Acc.$	P_k	WD	$Acc.$
NL-AuVi_N	0.36	0.36	0.88	0.33	0.33	0.95	**0.28**	**0.30**	0.98
NL-AuVi_C	0.35	0.36	0.88	**0.24**	**0.26**	0.95	0.33	0.40	0.96

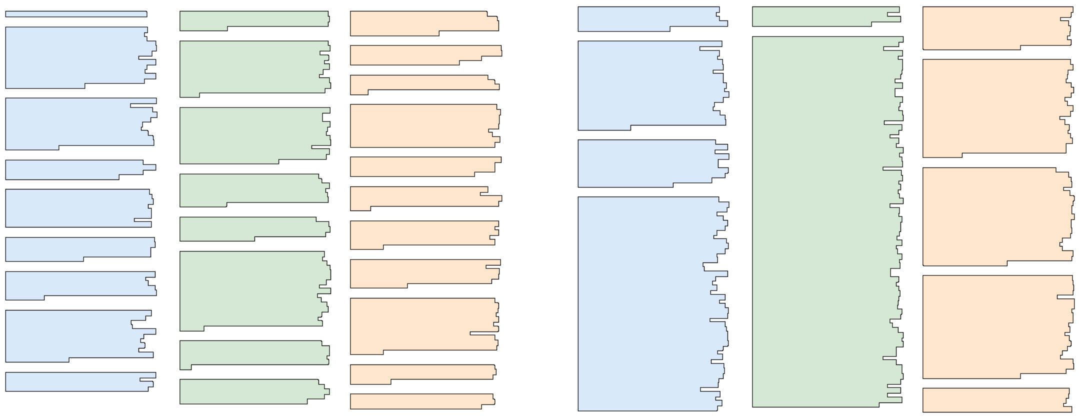
5.3 Unsupervised test results

Table 3 shows the unsupervised tests results, where a clear gradient in terms of dataset difficulty and algorithm effectiveness can be identified. First off, the results confirm our hypothesis that *EN-Wiki* < *NL-Wiki* < *NL-News* < *NL-AuVi* in terms of segmenting difficulty.

Table 3: Results from testing the unsupervised methods on all datasets. **Bold results** refer to the best performance for a test set. Underlined results refer to the best performance for a method.

Test sets	**EN-Wiki**			**NL-Wiki**			**NL-News**			**NL-AuVi**		
Method	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*
TextTiling	0.49	0.57	0.72	0.50	0.62	0.71	**0.46**	0.73	0.81	0.51	0.97	0.81
GraphSeg	0.50	0.56	0.77	0.53	0.63	0.78	0.61	0.81	0.79	0.64	0.97	0.80
BertTiling	**0.44**	**0.45**	0.84	**0.47**	**0.49**	0.83	0.47	**0.50**	0.90	**0.50**	**0.60**	0.96

Secondly, BertTiling significantly outperforms TextTiling and GraphSeg, especially in the *WD* metric. This is to be expected, since BertTiling is more state-of-the-art; by more closely examining some of the segmented documents, the underlying cause becomes clear. Consider Figure 3 as example, showcasing what segmented texts by TextTiling (and likewise GraphSeg) and BertTiling look like. For TextTiling, each of the segmented documents has roughly an equal number of sections, of similar sizes, as if it follows a pattern; on the other hand, each of the segmentations by BertTiling is distinct and seemingly more adjusted to the contents of the text. The usage of pre-trained BERT embeddings gives BertTiling this edge in robustness and effectiveness, whereas the rhythmic structure in the predictions of TextTiling and GraphSeg reveals how sensitive they are to parameter changes.



(a) TextTiling (b) BertTiling

Fig. 3: Three example documents from NL-Wiki (blue), NL-News (green) and NL-AuVi (orange), as segmented by TextTiling and BertTiling. Each block represents a single segment, containing a number of sentences.

These older algorithms do not capture as much semantic information as BertTiling, resulting in patterns, a rhythmic structure in their segmentations. Changing parameters (like k in TextTiling, approximating the average section length) directly changes these patterns and the resulting predictions; hence the importance of tuning the parameters correctly. We did our tuning based on the dataset characteristics we described at the end of Section 3.1, but these only got us so far – the algorithms struggle if the characteristic a parameter is supposed to describe varies greatly, like the average section length in NL-AuVi (sections between the first and third quantile range from 20 to 60 sentences). Especially GraphSeg still has room to improve with this in mind, since it failed to outperform the more basic TextTiling in our tests.

5.4 Supervised test results

The most self-evident takeaways from the supervised test results, as presented in Table 4, are that the newer models perform better on average – Transformers^2 > TextSeg+ > TextSeg in terms of performance – and that the best results are achieved when the model is trained on a dataset that is similar to/the same as the one it is tasked to segment. However, there are still some meaningful outliers worth pointing out.

The biggest outliers are the near flawless segmentations of NL-News documents, when done by models trained on the same corpus. Considering that these remarkable results are not repeated by other models or on any of the other test sets, we can be fairly confident that they overfitted on recognising the concatenated structure of these texts; something we also noticed (to a lesser extent) with the normal vs. concatenated NL-AuVi experiments.

Another exception is that the Transformers^2 models, in spite of their best average performance, fall behind the TextSeg based ones when trained on and tasked to segment NL-AuVi documents. We believe there are two possible explanations for this: for one, the TextSeg methods train their sentence encoder, while Transformers^2 sticks to pre-trained BERT embeddings; this should make the prior more resistant to the inherent noise present in the NL-AuVi documents. The other reason concerns the true quality of the segmentations made; when examining those made by the TextSeg algorithms by eye, it becomes apparent that the texts are not actually well segmented. We discuss this further in Section 6, but it seems that the TextSeg based models overfitted on maximising the P_k and WD scores.

5.5 MIXED dataset test results

Finally, the results for the models trained on the MIXED dataset (containing documents from each of the Dutch corpora) are also included in Table 4. The TextSeg model ended up getting better averages, but performed worse than models whose test and training set were from the same dataset. On the other hand, the Transformers^2 model performed significantly worse with the MIXED dataset and had the best averages when trained on the NL-Wiki dataset instead.

Table 4: Results from testing the supervised methods on all datasets. **Bold results** refer to the best performance for a test set. Underlined results refer to the best performance for a model. Results for TextSeg+ trained on the MIXED dataset are missing since we ran out of time. See Appendix A.2, Table 7 for averages.

Test sets		**EN-Wiki**			**NL-Wiki**			**NL-News**			**NL-AuVi**		
Method	*Trained on*	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*
TextSeg	*EN-Wiki*	0.28	0.31	0.90		-			-			-	
	NL-Wiki		-		0.34	0.38	0.89	0.38	0.43	0.94	0.49	0.54	0.97
	NL-News		-		0.49	0.49	0.88	**0.03**	0.05	0.99	0.43	0.53	0.96
	NL-AuVi		-		0.49	0.49	0.87	0.46	0.48	0.93	0.14	0.27	0.98
	MIXED		-		0.39	0.45	0.86	0.10	0.14	0.98	0.24	0.36	0.98
TextSeg+	*EN-Wiki*	0.24	0.27	0.92		-			-			-	
	NL-Wiki		-		0.30	0.34	0.90	0.31	0.39	0.94	0.45	0.52	0.97
	NL-News		-		0.48	0.48	0.88	**0.03**	0.05	0.99	0.40	0.50	0.96
	NL-AuVi		-		0.48	0.49	0.88	0.44	0.46	0.93	**0.13**	**0.23**	0.98
	MIXED		-			-			-			-	
Transformers[2]	*EN-Wiki*	**0.23**	**0.25**	0.91		-			-			-	
	NL-Wiki		-		**0.29**	**0.30**	0.90	0.26	0.28	0.95	0.34	0.36	0.98
	NL-News		-		0.36	0.37	0.89	**0.03**	**0.04**	0.99	0.33	0.39	0.96
	NL-AuVi		-		0.37	0.37	0.89	0.33	0.33	0.95	0.28	0.29	0.98
	MIXED		-		0.46	0.63	0.62	0.38	0.45	0.82	0.58	0.85	0.25

6 Discussion

6.1 The disconnect between numbers and results.

Going into this research, we and RTVMonitor wanted to end up with an algorithm that could consistently pick out the relevant section of a lengthy radio/TV transcription – the idea was that a Text Segmentation system could achieve this. But, despite the supervised algorithms showing promising *numbers*, none of them produced segmentations that we considered as passable. While there is still plenty of room to improve on – whether it be by designing a TS algorithm specialised for the data or task, or by adjusting the objective to more of a passage/information extraction task[8] – this disconnect between numbers and results has made us question the usefulness of the *Pk* and *WindowDiff* metrics, and of training to maximise them.

Consider the segmentations in Figure 4, neither TextSeg+ or Transformers^2 did a particularly good job at segmenting this text, but one could argue that TextSeg+ did a better job – it recognised more near-correct boundaries, around sentences 40 and 60 in particular. This assessment is not reflected in any of the metrics however, which claims that Transformers^2 is superior. To make matters even worse, if we were to score a segmentation with only a single boundary at the first sentence, it would outscore both algorithms with scores of (0.46, 0.46, 0.91) for its (P_k, *WD*, *Acc.*).

This is not the only text where unintuitive scores like these occur – Pevzner and Hearst [12], who introduced the *WD* metric, acknowledged the unintuitive nature of TS specific metrics as well. The bigger the imbalance between boundary and non-boundary sentences, which coincides with the increasing difficulties of

[8] Given a query, return the relevant sentences/information.

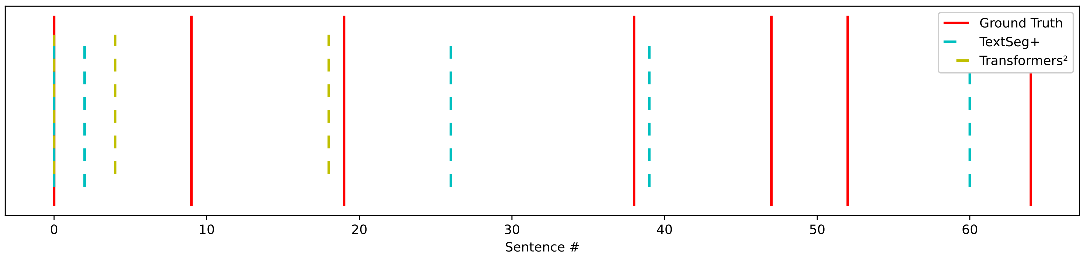


Fig. 4: Example segmentations by TextSeg+ and Transformers^2, with each vertical line denoting the starting sentence of a new section. Full text in Appendix A.3. Results of the segmentations:

Results	P_k	*WD*	*Acc.*
TextSeg+	0.57	0.58	0.86
Transformers^2	0.48	0.49	0.88

our datasets, the less useful these TS metrics appear to be. Nevertheless, without these metrics we would not be able to measure performance at all, so when looking to segment texts with a specific objective in mind, we would certainly suggest a custom(ised) metric adapted to that objective. This is easier said than done, but exploring what constitutes as a good segmentation for the task should be a good place to start.

6.2 What makes for a good segmentation?

The reason why we considered the segmentations of NL-AuVi texts to be unacceptable, is that they do not enable us to *consistently* pick out a single cohesive, relevant section from the resulting texts. What mattered to us was to capture as many of the relevant sentences as possible, while minimising the amount of irrelevant ones. So whether the placed boundaries were off by a few sentences was unimportant in the grand scheme of things, as long as we had at least three sections, with the middle one being placed near its true location – we could even enlarge/shorten the section in post-processing to improve matters. With this objective, the focus lies on preventing false negatives, to always have at least three segments; false positives would not need to be penalised as hard.

This is specific to the RTVMonitor objective, where these cohesive, relevant sections are needed for creating good summaries, among other things, but another task will have a different focus, a different definition of what makes for a well segmented text.

6.3 Research limitations

With only three months to research, implement, and test everything, we were bound to reach limits of what was feasible, and not every error could be addressed. The NL-News dataset we created, on one hand showed some of the shortcomings of concatenated datasets, but also why extra care ought to be taken when creating this kind of augmented data. The newspaper articles we used had to be carefully processed, as they often contained obvious openings, like:

"<LOCATION> - Today...",

or endings, such as:

"(<PUBLISHER>) Photo <NAME>".

We did attempt to filter all of these out, but some still managed to get through and made the dataset too easy as a result.

Not every method implementation was perfect either. We did not entirely grasp the code behind TextSeg+, making it difficult to fully optimise; because of this we did not have to time to train a model on the MIXED dataset. The Transformers^2 model lacked the right masking method which was poorly described in its accompanying paper – this could have been a rather impactful element, especially for the large class imbalance present in the NL-AuVi documents.

The training method should also be brought into question; training to maximise P_k and WD could have been a mistake, considering the unintuitive nature of these metrics and the fact that we saw the validation loss rise as the P_k and WD was still decreasing in the final few training epochs.

Lastly, we did not get a complete answer to whether the NL-AuVi dataset made for a good TS training dataset or not. While we can be quite certain that concatenation does not improve matters, we still have our doubts about its unusual structure, which essentially is only a partially labelled text.

7 Conclusion

Our paper highlights the effects of differing Dutch datasets for a Text Segmentation task. We show that as texts become less formal, less concise, and less informative, that they become more difficult to segment, and less suitable as *general* training data. Additionally, we point out the shortcomings of datasets consisting of concatenated documents; these should be avoided if possible, as they tend to be too easy to segment for supervised TS algorithms, and unrepresentative of real-world data. Our results also confirm that the best results are achieved when training on the same data that the algorithm is going to segment; mixing data from different sources only produces better results on average.

Furthermore, from the algorithms we examined, we conclude that attention mechanisms/Transformers provide noticeable performance gain in TS tasks, and that a custom and trainable text encoder, even a simplistic one, is preferable over a pre-trained encoder, when working with challenging texts. Future work should also look into masking mechanisms as a likely area for improvement, especially when dealing with difficult texts where a large class imbalance between boundary and non-boundary texts exists.

Finally, we believe that the common TS specific metrics, P_k and WD, while still useful, should not be relied upon too much, if looking for specific kinds of results. The unintuitive nature of these metrics can make them misleading; they describe performance, not necessarily the quality of the results, which ultimately is subjective and task-dependant.

References

1. Akinnaso, F.N.: On the differences between spoken and written language. Language and speech **25**(2), 97–125 (1982)
2. Arnold, S., Schneider, R., Cudré-Mauroux, P., Gers, F.A., Löser, A.: Sector: A neural model for coherent topic segmentation and classification. Transactions of the Association for Computational Linguistics **7**, 169–184 (2019)
3. Badjatiya, P., Kurisinkel, L.J., Gupta, M., Varma, V.: Attention-based neural text segmentation. In: European Conference on Information Retrieval. pp. 180–193. Springer (2018)
4. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Machine learning **34**(1), 177–210 (1999)

5. Choi, F.Y.: Advances in domain independent linear text segmentation. arXiv preprint cs/0003083 (2000)
6. Eisenstein, J., Barzilay, R.: Bayesian unsupervised topic segmentation. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. pp. 334–343 (2008)
7. Glavaš, G., Nanni, F., Ponzetto, S.P.: Unsupervised text segmentation using semantic relatedness graphs. In: Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics. pp. 125–130. Association for Computational Linguistics (2016)
8. Hearst, M.A.: Text tiling: Segmenting text into multi-paragraph subtopic passages. Computational linguistics **23**(1), 33–64 (1997)
9. Koshorek, O., Cohen, A., Mor, N., Rotman, M., Berant, J.: Text segmentation as a supervised learning task. arXiv preprint arXiv:1803.09337 (2018)
10. Lo, K., Jin, Y., Tan, W., Liu, M., Du, L., Buntine, W.: Transformer over pre-trained transformer for neural text segmentation with enhanced topic coherence. arXiv preprint arXiv:2110.07160 (2021)
11. Pak, I., Teh, P.L.: Text segmentation techniques: a critical review. Innovative Computing, Optimization and Its Applications pp. 167–181 (2018)
12. Pevzner, L., Hearst, M.A.: A critique and improvement of an evaluation metric for text segmentation. Computational Linguistics **28**(1), 19–36 (2002)
13. Riedl, M., Biemann, C.: Topictiling: a text segmentation algorithm based on lda. In: Proceedings of ACL 2012 Student Research Workshop. pp. 37–42 (2012)
14. Sehikh, I., Fohr, D., Illina, I.: Topic segmentation in asr transcripts using bidirectional rnns for change detection. In: 2017 IEEE automatic speech recognition and understanding workshop (ASRU). pp. 512–518. IEEE (2017)
15. Solbiati, A., Heffernan, K., Damaskinos, G., Poddar, S., Modi, S., Cali, J.: Unsupervised topic segmentation of meetings with bert embeddings. arXiv preprint arXiv:2106.12978 (2021)
16. Somasundaran, S., et al.: Two-level transformer and auxiliary coherence modeling for improved text segmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, no. 05, pp. 7797–7804 (2020)
17. Wikipedia contributors: List of wikipedias – edition details. https://en.wikipedia.org/wiki/List_of_Wikipedias#Edition_details (2022), [Online; accessed 16-April-2022]
18. Xing, L., Hackinen, B., Carenini, G., Trebbi, F.: Improving context modeling in neural topic segmentation. arXiv preprint arXiv:2010.03138 (2020)
19. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. pp. 1480–1489 (2016)

A Appendix

A.1 Hyperparameter Settings

Table 5: Hyperparameter settings for the methods used. Parameters not mentioned are left on default and can be found in the code as well.

Method	**Hyperparameters**
TextTiling	w = Avg. sentence length k = Avg. section length Cutoff policy: HC
GraphSeg	Relatedness Threshold = 0.25 Minimal segment size = 2
BertTiling	k = Avg. section length Cutoff policy: HC
All supervised algorithms	Max dataset size = 100000 Train/Validate/Test split = $[80\%, 10\%, 10\%]$ 15 epochs max – Early stop after 3 static epochs Max sentence length = 30 words Max section length = 70 sentences Max text length
TextSeg & TextSeg+	Learning rate = 0.001 LSTM - Hidden size = 256 LSTM - Num layers = 2 (Bidirectional)
Transformers[2]	Learning rate = 0.0001 Transformer - nhead = 24 Transformer - dim feedforward = 1024 Transformer - num encoder layers = 5

A.2 Averages Tables

Table 6: Dataset analysis averages. (Doc. = Document; Sect. = Section; Sent. = Sentence)

	Sect./Doc.	**Sent./Doc.**	**Sent./Sect.**	**Mean Sent. Length (in Words)/Doc.**
EN-Wiki	4.85	33.45	5.84	19.94
NL-Wiki	3.77	17.96	3.94	14.7
NL-News	3.5	49.86	11.88	16.66
NL-AuVi_C	3.49	136.55	23.84	14.97
NL-AuVi_N	2.99	120.84	39.72	13.96

Table 7: Average results from testing the supervised methods on all datasets. **Bold results** refer to the best performance for the method. Inclusive/Exclusive concerns the test results on the test set from the same dataset as the training set.

Test sets		Avg. Inclusive			Avg. Exclusive		
Method	*Trained on*	P_k	*WD*	*Acc.*	P_k	*WD*	*Acc.*
TextSeg	*NL-Wiki*	0.41	0.45	0.93	**0.44**	**0.48**	0.95
	NL-News	0.32	0.36	0.95	0.46	0.51	0.92
	NL-AuVi	0.36	0.41	0.93	0.47	0.49	0.90
	MIXED	**0.24**	**0.31**	0.94		-	
TextSeg+	*NL-Wiki*	0.36	0.42	0.94	**0.38**	**0.46**	0.95
	NL-News	**0.30**	**0.35**	0.95	0.44	0.50	0.92
	NL-AuVi	0.35	0.39	0.93	0.46	0.47	0.91
	MIXED		-			-	
Transformers^2	*NL-Wiki*	0.30	0.31	0.95	**0.30**	**0.32**	0.97
	NL-News	**0.24**	**0.27**	0.95	0.35	0.38	0.93
	NL-AuVi	0.32	0.33	0.94	0.35	0.35	0.92
	MIXED	0.47	0.64	0.57		-	

A.3 Example Text

`wiki_6.txt` from the NL-Wiki dataset. ▮ for true boundaries, 🟦 for TextSeg+ boundaries, and 🟨 for Transformers^2 boundaries.

▮ 🟦 🟨 Groen is een Vlaamse, progressieve en groene politieke partij, die 10.000 leden telt (2018). Pacifisme, sociale rechtvaardigheid en ecologie (duurzame ontwikkeling) zijn drie pijlers van het Groen-programma. 🟦 De partij noemt 'de kwaliteit van leven' de 'groene draad door haar programma'. Groen is een zusterpartij van het Frans- en Duitstalige Ecolo. 🟨 Van 1979 tot 15 november 2003 heette de partij Agalev. Daarna werd de naam Groen!. Begin 2012 werd het uitroepteken weggelaten. De huidige voorzitter is Meyrem Almaci. In 2017 had de partij 9.484 leden en is daarmee de kleinste in het parlement vertegenwoordigde Vlaamse partij wat het aantal partijleden betreft.

▮ Na een korte chaotische periode in de zomer van 2003 neemt Vera Dua het roer over van Dirk Holemans en brengt de rust terug in het groene kamp. De zware klap voor de groenen heeft nog meer gevolgen: honderden nieuwe leden melden zich aan. Intussen wordt er getimmerd aan een nieuw inhoudelijk project. De sleutelwoorden van deze vernieuwde missie zijn "solidariteit", "lange termijn" en "grenzen". Om de vernieuwing volop duidelijk te maken voor de buitenwereld kiest men ook voor een nieuwe naam die meer gelijkenis vertoont met die van de andere Europese groene partijen, kortweg Groen!. Op het congres van november 2003 wordt het partij-jargon nog meer gemoderniseerd: Dua mag zich als eerste groene "voorzitter" laten noemen in plaats van "politiek secretaris". Voor het eerst mogen de groene verkiezingskandidaten ook persoonlijke affiches gebruiken. Ook enkele andere oude principes sneuvelen om efficiënter te kunnen werken. Desalniettemin blijft de uiteindelijke beslissingsmacht bij het ledencongres liggen. 🟨 Een aantal bekende gezichten (zoals het Antwerps boegbeeld Fauzaya Talhaoui en de toenmalige minister Ludo Sannen) stapten vóór de verkiezingen van 2004 over naar het kartel sp.a - Spirit.

▮ Groen! kiest voor de verkiezingen van 13 juni voor een gerichte campagne. Met als inzet "Vera zoekt... 280.000 mensen" worden alle creatieve en weinig kostende middelen ingezet voor een dynamische campagne. Groen! spreekt haar kiezers rechtstreeks aan met de verkiezingsslogan "De bal ligt in uw kamp". Het is een spannende campagne, maar op de avond van de verkiezingen blijkt dat Groen! het gehaald heeft: de partij haalt opnieuw scores die ruim boven de kiesdrempel uitsteken: 7,6% voor het Vlaamse, 9,8% voor het Brusselse en 8% voor het Europees Parlement. 🟦 In Limburg slaagde de partij er echter niet in om de kiesdrempel te halen. Na de verkiezingen beslist de partij om in de oppositie te gaan. In de periode na de verkiezingen van 2004 keert de rust een beetje weer in de partij. Ondanks het feit dat de financiële en personele gevolgen van de nederlaag van 2003 nog zwaar doorwegen, wordt er gewerkt aan de versterking van de partij met het oog op de volgende jaren. Er wordt sterk geïnvesteerd in inhoud met congressen in 2005 en 2006. Bij de federale verkiezing in 2007 slaagt Groen! er in terug in Kamer en Senaat te komen. In 2007 had Groen! vier zetels in de Kamer van volksvertegenwoordigers, twee in de Senaat, zes in het Vlaams Parlement, één in het Brusselse en één in het Europees Parlement. In het Europees Parlement zetelt ex-Volksunie Europarlementariër Bart Staes voor de Europese Federatie van Groene Partijen/Europese Vrije Alliantie; hij vormt daar met het Nederlandse GroenLinks een transnationale fractie. Op 10 november 2007 werd tijdens een partijcongres een opvolger gekozen voor partijvoorzitster Vera Dua. Tussen zes kandidaten werd Mieke Vogels in twee stemronden verkozen.

| Na de Vlaamse verkiezingen op 7 juni 2009 vergrootte Groen! | zijn aantal zetels in het Vlaams Parlement van 6 naar 7. Na de Europese verkiezingen die ook op 7 juni gehouden werden, bleef Bart Staes de enige vertegenwoordiger in het Europees Parlement voor de Vlaamse groene partij. Eind 2009 beslisten zowel Groen! als de Sociaal-Liberale Partij dat SLP voortaan deel zou uitmaken van Groen!. Niet iedereen was daar echter mee akkoord: ex-minister Jef Tavernier en Rita Brauwers, fractieleidster in de Brugse gemeenteraad, verlieten uit onvrede de partij. In het voorjaar van 2008 werd Wouter Van Besien, die bij de voorzittersverkiezingen tweede was, aangeduid als ondervoorzitter. Op 25 oktober 2009 volgde hij Vogels - die ontslag nam - op als voorzitter, terwijl Björn Rzoska ondervoorzitter werd. In 2013 werd die laatste opgevolgd door Elke Van den Brandt.

| Voor het eerst slaagde de partij erin op provinciaal niveau deel uit te maken van het bestuur, namelijk in Vlaams-Brabant. Op zaterdag 15 december 2012 legde Luc Robijns de eed af als provinciaal gedeputeerde. Hij werd in september 2014 opgevolgd door Tie Roefs. Na de verkiezingen in 2018 werd Riet Gillis de eerste gedeputeerde van Groen in het provinciebestuur van Oost-Vlaanderen. Ondanks electorale vooruitgang in Vlaams-Brabant werd Groen er niet meer opgenomen in de nieuwe provinciale bestuurscoalitie.

| Sinds begin 2019 waren milieu en ecologie uitgegroeid tot een van de belangrijkste politieke thema's met bijzonder veel persaandacht voor onder meer de klimaatstakingen en diverse klimaatbetogingen. De partij had de beweging ontstaan rond Anuna De Wever ook van bij de start ondersteund en aangemoedigd. Toen peilingen in het voorjaar van 2019 Groen tot 16% deden pieken, met eventuele uitschieters richting 19% rekening houdend met de foutenmarge, waren de verwachtingen zeer hoog gespannen. Die verwachtingen werden echter niet ingelost: ongeveer 10 procent in het Vlaams Parlement en net geen 10 procent in de Kamer. De partij boekte daarmee een winst van iets meer dan een procentpunt op Vlaams niveau en ongeveer een procentpunt op federaal niveau. Ten opzichte van de lokale verkiezingen van 14 oktober 2018 ging de partij zelfs achteruit. Analisten linkten dit verlies mede aan twee dossiers waar de partij niet lang voor de stemslag in het defensief werd gedrongen: het afschaffen van het fiscaal interessante statuut van de salariswagen enerzijds en het invoeren van een vermogenskadaster anderzijds. Dat laatste werd op basis van de tekst die de partij in de memorie van toelichting neerschreef door Het Laatste Nieuws omschreven als: "Je moet elke fles wijn in je kelder en elk boek in je kast aangeven", wat nuancering verdiende. | Toch was de balans positief : met in totaal 28 parlementsleden, waarvan 11 nieuwkomers, had Groen nog nooit zo veel verkozenen. Ook de Kamerfractie gebundeld met de Franstalige zusterpartij Ecolo, heeft met 21 leden, de grootste groene fractie ooit. Ze maakt deel uit van de meerderheid van de nieuwe federale regering-De Croo die aantrad op 1 oktober 2020. In het Brussels Hoofdstedelijk Gewest was Groen de grootste Vlaamse partij geworden in Brussel en de partij trad toe tot de regering-Vervoort III, waarbij minister Elke Van den Brandt tevens het ministerieel college van de Vlaamse Gemeenschapscommissie voorzit.

| Agalev, zoals de partij tot 2003 heette, kende geen echte partijvoorzitter. Zowel het wekelijkse Uitvoerend Comité (partijbestuur) als de maandelijkse Stuurgroep (partijraad) werden voorgezeten door een "gespreksleider". Het dagelijks bestuur lag bij de secretaris(sen) enerzijds en de fractieleiders in de parlementen anderzijds. De voorzitter en ondervoorzitter worden door de leden op een congres verkozen voor een termijn van vijf jaar. Zij moeten een verschillende genderidentiteit hebben.