University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Document Structure Analysis By Means Of Sentence Classification

by

Maarten de Jonge

10002109

May 3, 2018

42 EC
01 July 2017, 31 December 2017

Supervisor:
Dr Maarten Marx

Assessor:
Dr To Be Determined

Information and Language Processing Systems Group
Informatics Institute

Contents

1 Introduction

The Political Mashup project[1] aims to digitize the world's political proceedings in order to make them easily accessible and searchable. Unfortunately, the published documents are often primarily intended to be human-readable, without the embedded semantic structure required to properly index this data in a digital way. This semantic information is currently recovered using rule-based methods. Since the data gets transcribed by a human typist, compiled to a PDF, and then goes back into an imperfect PDF decompiler, there is a lot of room for

[1] http://search.politicalmashup.nl/about.html

minor variations in the output even though the layout of the document itself is consistent. Dealing with this in a rule-based system entails using either broad rules that lead to a larger probability of false positives, or a large amount of narrow rules which can quickly lead to a spaghetti-like mess of special cases and is very fragile to unseen issues.

I propose that by using a small number of manually annotated documents as a dataset, a machine learning algorithm can learn to classify sentences in a way that allows it to segment a document into its constituent parts, while being more robust to noise than its rule-based counterpart. The common ways to do sentence classification (e.g. convolutional neural networks [1] , recurrent neural networks or the simpler bag-of-words models) operate on sentences in a vacuum, considering only their linguistic contents and ignoring any contextual information that might be present. This is to be expected considering that most of the common datasets in this area really *are* just small bits of text in a vacuum; often-used datasets involve Twitter messages or short product reviews. In this case however, the sentences come from a document with a rich structure providing a lot of context. Anecdotally, as a human it is trivial to discern section headers in a document even when the document is in a foreign language; simply the fact that the section header might be printed in bold and centered rather than left-aligned gives it away. Incorporating this structural data into the learning process will hopefully increase the performance of the system, either by simply scoring better on the used metrics, or perhaps more indirectly by requiring less data or training time to achieve the same score.

2 Related Works

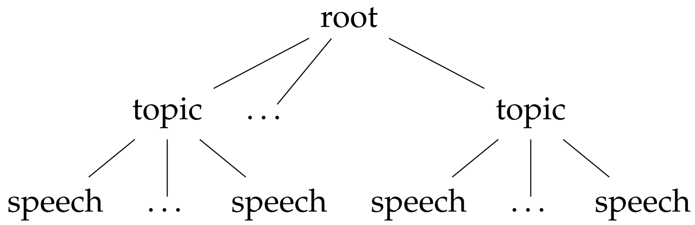
Todo: Expand section

Various forms of convolutional neural networks are commonly used for text classification. The most basic architecture is described by Kim [1], where the input words are tokenized and embedded before passing them to the convolutional neural network. Additional exploration of the parameter space and its effect on various datasets is done by Zhang & Wallace [2]. Comparable results are achieved by Zhang *et al.* [3] by operating on the character-level rather than the word-level, bypassing the issues overhead of using word embedding (either in extra training time or in finding suitable pretrained embeddings). All the previously mentioned architectures use a single convolutional layer; this is somewhat contrary to current trends in computer vision, where popular models such as ResNet[4] go as deep as 152 layers. This difference is explored by Conneau *et al.* [5], who take a character-level CNN and show that adding more layers improves performance, before leveling out at 29 layers. They hypothesize that the difference in effective depth between computer vision and language processing might be due to the difference in datasets. The common ImageNet dataset used in computer vision deals with 1000 classes; in contrast, sentiment analysis datasets vary between 2 and 25 classes. In addition, they note that the deeper networks do require a larger amount of data to train.

In terms of analyzing document structure, Klampfl *et al.* [6] introduce a method to analyze scientific articles, detecting blocks of text, labeling them (as e.g. section headers, tables or references) and determining the reading order — all in an unsupervised manner. While their approach to block detection forms an integral part of this thesis, the rest is too specifically tied to the format of scientific articles to be applicable in this scenario.

3 Problem Statement

The thesis will focus on parliamentary proceedings from the German *Bundestag*. These proceedings are available online as PDF files dating back to 1949, and have used essentially the same document layout and conventions since the start. Semantically, the layout of these documents forms a shallow tree:



Each document consists of a series of topics to discuss, where each topic contains a series of speeches made by the present politicians. The task to be solved is retrieving these speeches as accurately and robustly as possible. Due to the shallow nature of the document's layout, marking where a new speech begins is sufficient for retrieving the layout to a sufficient degree. Although this discards information about the overarching topics, this information is both more difficult to extract (it is represented in a somewhat more free-form manner in the text) and less fundamental to what the dataset might end up being used for. Luckily, in this case a rule-based system for segmenting the files already exists and was used to create a fairly large training set. Since in general such a system will not exist and annotating the data by hand is expensive, a big focus is on limiting the amount of required training data as much as possible; it would be preferable if the system was able to learn sufficiently from a handful (say, less than 5) of hand-annotated files.

3.1 Dataset

PDF files are unfortunately rather difficult to work with; being a vector-based format, they have no internal concept of words or sentences. All that's available are instructions for drawing a certain character at a certain position. This means that even something as seemingly trivial as obtaining the lines of text from a PDF requires some fairly involved logic and heuristics (for instance, one would think that simply taking all characters on a page with the same y coordinate would be sufficient until realising that many documents have a layout with two columns of text). This is dealt with by using the `pdftohtml` script from the Poppler PDF rendering library[2]. This script converts a PDF file to an XML file containing logical lines of text along with the coordinates and size of the line. Figure 1 shows an example of a portion of a PDF file and the corresponding XML produced by `pdftohtml`.

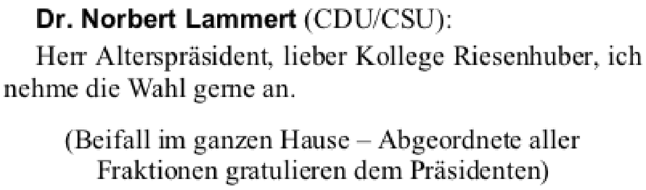
The dataset was obtained through a rule-based system as described in the introduction, which annotates each `<text>` element of the XML files with a boolean flag indicating whether said element starts a new speech. The system was run on documents from the 18th electoral period of the *Bundestag*, consisting of 211 documents dating from 2013 to 2017. Together these documents contain 43,252 `<text>` elements indicating the start of speeches (that is, positive training samples), and 2,602,793 other elements (negative samples). This adds to a total of 2,646,045 training samples, taking up 503 MiB. This is a rather lopsided distribution (there are roughly 60 negative samples for each positive sample), which might have to be accounted for by, for instance, subsampling negative samples before training. Figure 2 shows the distribution of the number of positive samples per file, giving a guideline as to how many files would have to be annotated to reach a desired amount of positive samples.

3.2 Research Question

As stated in the introduction, the research done in this thesis is about augmenting the data with the spatial information that is usually left out in sentence classification. In doing so, the following research questions will be considered:

1. Does this increase the learning performance?
2. Does this allow for reaching the same performance using less data?
3. Does this make the training converge to the optimal performance quicker?

[2] https://poppler.freedesktop.org/



Dr. Norbert Lammert (CDU/CSU):

Herr Alterspräsident, lieber Kollege Riesenhuber, ich nehme die Wahl gerne an.

(Beifall im ganzen Hause – Abgeordnete aller Fraktionen gratulieren dem Präsidenten)

(a) The source PDF

```
<text top="122" left="125" width="143" height="16" font="3">
    <b>Dr. Norbert Lammert </b>
</text>
<text top="122" left="269" width="83" height="17" font="4">
    (CDU/CSU) :
</text>
<text top="142" left="125" width="328" height="17" font="4">
    Herr Alterspr sident , lieber Kollege Riesenhuber , ich
</text>
<text top="158" left="108" width="156" height="17" font="4">
    nehme die Wahl gerne an .
</text>
<text top="186" left="141" width="278" height="17" font="4">
    ( Beifall im ganzen Hause   Abgeordnete aller
</text>
<text top="203" left="158" width="242" height="17" font="4">
    Fraktionen gratulieren dem Pr sidenten )
</text>
```

(b) XML

Figure 1: The data representations

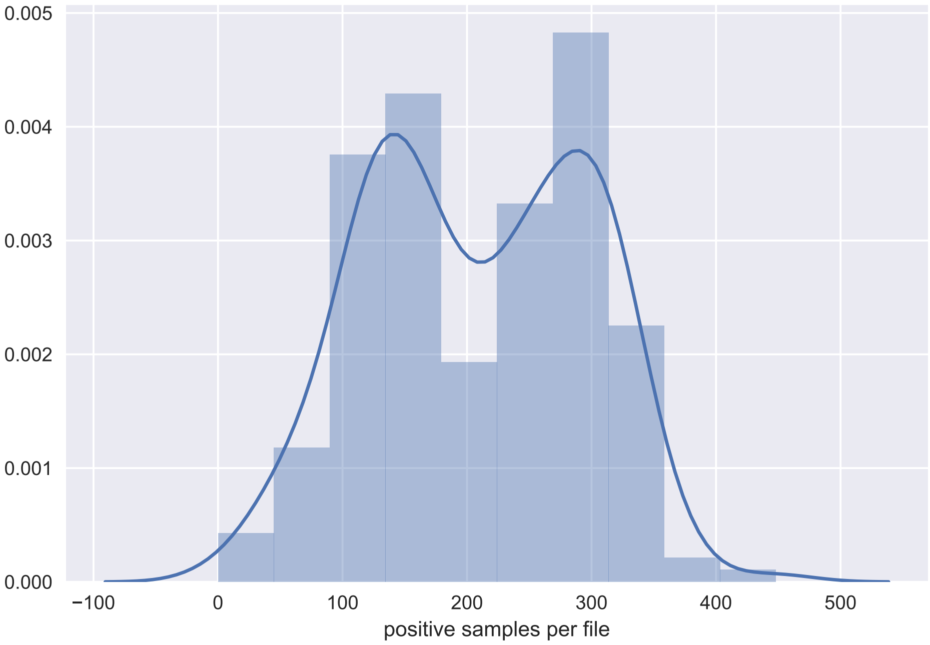


Figure 2: Distribution of the number of positive samples per file

4 Methodology

The system consists of two separate parts; an unsupervised algorithm for augmenting data, and a supervised algorithm for classifying the data (Figure 3). The unsupervised portion attempts to augment the data with additional structural information. It could be considered a preprocessing step, with the choice of parameters acting as a way to inject some amount of domain knowledge into the data. The supervised portion consists of a regular classification algorithm.

Todo: Rewrite this and add all the details about how the data is used (sliding window, stratified sampling, etc)

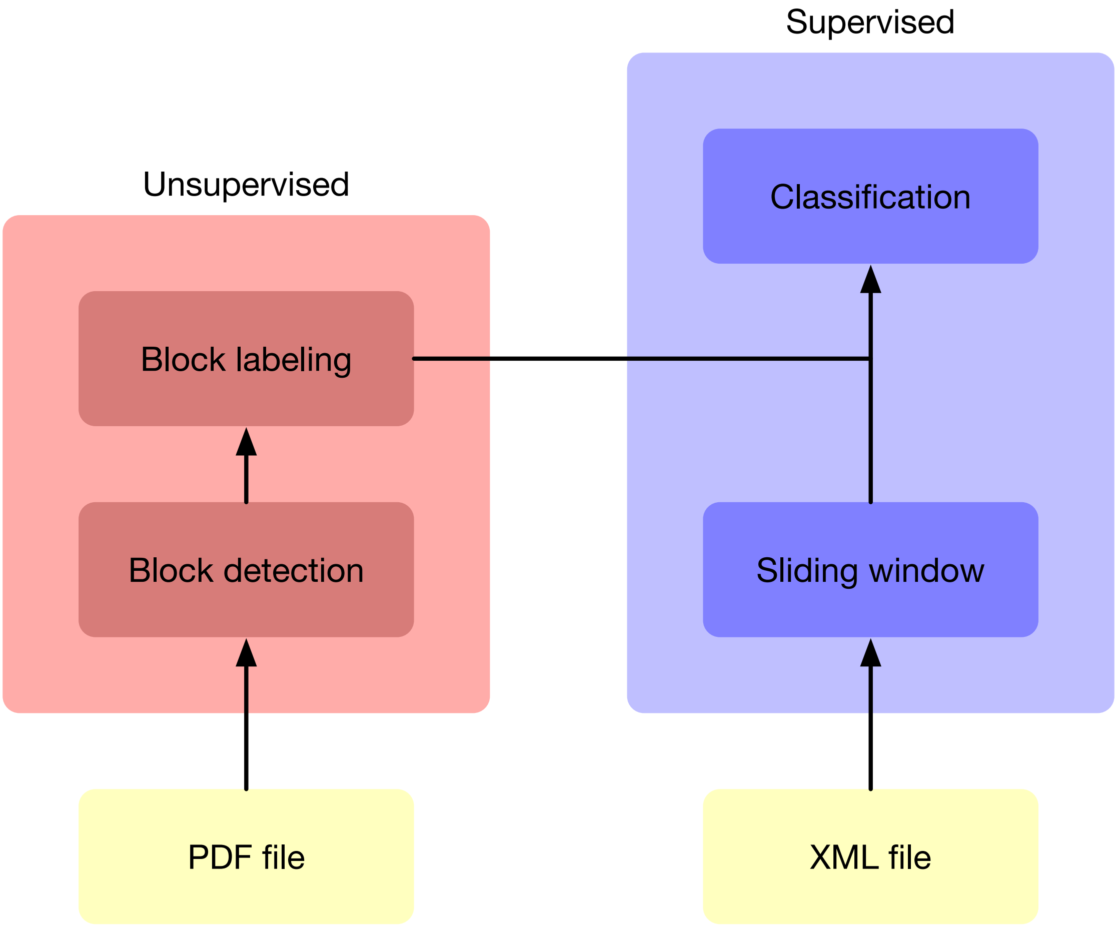


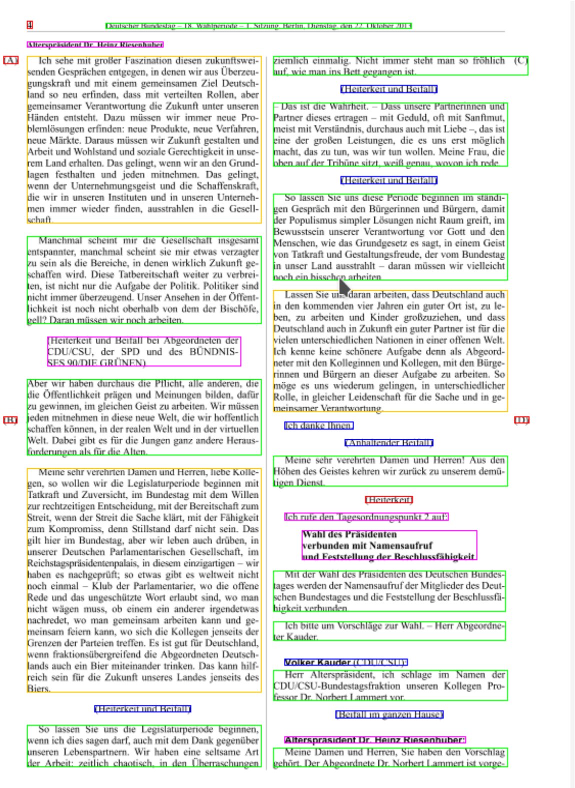
Figure 3: A high-level overview of the system

4.1 Unsupervised

The unsupervised algorithm attempts to detect and label blocks of text in the PDF file, as shown in Figure 4 for an example). This approach is based on work by Klampfl *et al.* [6], and consists of two clustering steps:

1. Individual letters are clustered together into blocks of semantically relevant text (e.g. a full paragraph, or a section header).

2. These blocks are labeled by a different clustering algorithm.



4 Deutscher Bundestag – 18. Wahlperiode – 1. Sitzung. Berlin, Dienstag, den 22. Oktober 2013

Alterspräsident Dr. Heinz Riesenhuber

(A) Ich sehe mit großer Faszination diesen zukunftsweisenden Gesprächen entgegen, in denen wir aus Überzeugungskraft und mit einem gemeinsamen Ziel Deutschland so neu erfinden, dass mit verteilten Rollen, aber gemeinsamer Verantwortung die Zukunft unter unseren Händen entsteht. Dazu müssen wir immer neue Problemlösungen erfinden: neue Produkte, neue Verfahren, neue Märkte. Daraus müssen wir Zukunft gestalten und Arbeit und Wohlstand und soziale Gerechtigkeit in unserem Land erhalten. Das gelingt, wenn wir an den Grundlagen festhalten und jeden mitnehmen. Das gelingt, wenn der Unternehmungsgeist und die Schaffenskraft, die wir in unseren Instituten und in unseren Unternehmen immer wieder finden, ausstrahlen in die Gesellschaft.

Manchmal scheint mir die Gesellschaft insgesamt entspannter, manchmal scheint sie mir etwas verzagter zu sein als die Bereiche, in denen wirklich Zukunft geschaffen wird. Diese Tatbereitschaft weiter zu verbreiten, ist nicht nur die Aufgabe der Politik. Politiker sind nicht immer überzeugend. Unser Ansehen in der Öffentlichkeit ist noch nicht oberhalb von dem der Bischöfe, gell? Daran müssen wir noch arbeiten.

(Heiterkeit und Beifall bei Abgeordneten der CDU/CSU, der SPD und des BÜNDNISSES 90/DIE GRÜNEN)

(B) Aber wir haben durchaus die Pflicht, alle anderen, die die Öffentlichkeit prägen und Meinungen bilden, dafür zu gewinnen, im gleichen Geist zu arbeiten. Wir müssen jeden mitnehmen in diese neue Welt, die wir hoffentlich schaffen können, in der realen Welt und in der virtuellen Welt. Dabei gibt es für die Jungen ganz andere Herausforderungen als für die Alten.

Meine sehr verehrten Damen und Herren, liebe Kollegen, so wollen wir die Legislaturperiode beginnen mit Tatkraft und Zuversicht, im Bundestag mit dem Willen zur rechtzeitigen Entscheidung, mit der Bereitschaft zum Streit, wenn der Streit die Sache klärt, mit der Fähigkeit zum Kompromiss, denn Stillstand darf nicht sein. Das gilt hier im Bundestag, aber wir leben auch drüben, in unserer Deutschen Parlamentarischen Gesellschaft, im Reichstagspräsidentenpalais, in diesem einzigartigen – wir haben es nachgeprüft; so etwas gibt es weltweit nicht noch einmal – Klub der Parlamentarier, wo die offene Rede und das ungeschützte Wort erlaubt sind, wo man nicht wägen muss, ob einem ein anderer irgendetwas nachredet, wo man gemeinsam arbeiten kann und gemeinsam feiern kann, wo sich die Kollegen jenseits der Grenzen der Parteien treffen. Es ist gut für Deutschland, wenn fraktionsübergreifend die Abgeordneten Deutschlands auch ein Bier miteinander trinken. Das kann hilfreich sein für die Zukunft unseres Landes jenseits des Biers.

(Heiterkeit und Beifall)

So lassen Sie uns die Legislaturperiode beginnen, wenn ich dies sagen darf, auch mit dem Dank gegenüber unseren Lebenspartnern. Wir haben eine seltsame Art der Arbeit: zeitlich chaotisch, in den Überraschungen ziemlich einmalig. Nicht immer steht man so fröhlich auf, wie man ins Bett gegangen ist. (C)

(Heiterkeit und Beifall)

– Das ist die Wahrheit. – Dass unsere Partnerinnen und Partner dieses ertragen – mit Geduld, oft mit Sanftmut, meist mit Verständnis, durchaus auch mit Liebe –, das ist eine der großen Leistungen, die es uns erst möglich macht, das zu tun, was wir tun wollen. Meine Frau, die oben auf der Tribüne sitzt, weiß genau, wovon ich rede.

(Heiterkeit und Beifall)

So lassen Sie uns diese Periode beginnen im ständigen Gespräch mit den Bürgerinnen und Bürgern, damit der Populismus simpler Lösungen nicht Raum greift, im Bewusstsein unserer Verantwortung vor Gott und den Menschen, wie das Grundgesetz es sagt, in einem Geist von Tatkraft und Gestaltungsfreude, der vom Bundestag in unser Land ausstrahlt – daran müssen wir vielleicht noch ein bisschen arbeiten.

Lassen Sie uns daran arbeiten, dass Deutschland auch in den kommenden vier Jahren ein guter Ort ist, zu leben, zu arbeiten und Kinder großzuziehen, und dass Deutschland auch in Zukunft ein guter Partner ist für die vielen unterschiedlichen Nationen in einer offenen Welt. Ich kenne keine schönere Aufgabe denn als Abgeordneter mit den Kolleginnen und Kollegen, mit den Bürgerinnen und Bürgern an dieser Aufgabe zu arbeiten. So möge es uns wiederum gelingen, in unterschiedlicher Rolle, in gleicher Leidenschaft für die Sache und in gemeinsamer Verantwortung.

Ich danke Ihnen. (D)

(Anhaltender Beifall)

Meine sehr verehrten Damen und Herren! Aus den Höhen des Geistes kehren wir zurück zu unserem demütigen Dienst.

(Heiterkeit)

Ich rufe den Tagesordnungspunkt 2 auf:

Wahl des Präsidenten
verbunden mit Namensaufruf
und Feststellung der Beschlussfähigkeit

Mit der Wahl des Präsidenten des Deutschen Bundestages werden der Namensaufruf der Mitglieder des Deutschen Bundestages und die Feststellung der Beschlussfähigkeit verbunden.

Ich bitte um Vorschläge zur Wahl. – Herr Abgeordneter Kauder.

Volker Kauder (CDU/CSU):

Herr Alterspräsident, ich schlage im Namen der CDU/CSU-Bundestagsfraktion unseren Kollegen Professor Dr. Norbert Lammert vor.

(Beifall im ganzen Hause)

Alterspräsident Dr. Heinz Riesenhuber:

Meine Damen und Herren, Sie haben den Vorschlag gehört. Der Abgeordnete Dr. Norbert Lammert ist vorge-

Figure 4: An example of clustered blocks of text, blocks with the same outline color belonging to the same cluster.

4.1.1 Hierarchical Agglomerative Clustering

The first step is performed using hierarchical agglomerative clustering (HAC), an unsupervised bottom-up clustering algorithm that constructs a hierarchical tree of clusters (in this context referred to as a *dendrogram*). An example is shown in Figure 5. The algorithm gets fed the individual characters present in the PDF files, then iteratively groups the two closest clusters (the initial inputs being regarded as clusters of one element) together until only a single cluster remains. This process involves two parameters:

1. The distance function between two characters.
2. The distance function between two groups of characters.

The first parameter is trivially chosen to be the Euclidian distance between the coordinates of the two characters. The second parameter is called the *linkage* and has several common options, the most basic of which are:

- Single-linkage: The distance between groups is based on the closest two elements:
$$d(A,B) = \min\{d(a,b) : a \in A, b \in B\}$$
- Maximum-linkage: The distance between groups is based on the furthest two elements:
$$d(A,B) = \max\{d(a,b) : a \in A, b \in B\}$$
- Average-linkage: The distance between groups is based on the average distance of its elements:
$$d(A,B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b)$$

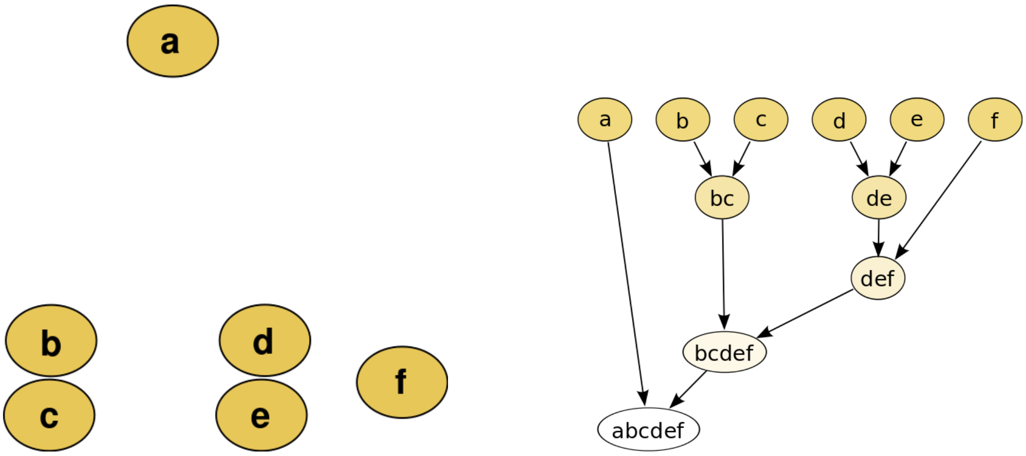
As per Klampfl *et al.* [6], single-linkage clustering performs best for this task due to its tendency to form long thin clusters, mimicking the structure of sentences. As an additional bonus, while the general time complexity for HAC is in $\mathcal{O}(n^3)$, single-linkage clustering can be done in $\mathcal{O}(n^2)$ [7], making it far more usable on realistic datasets.

After the dendrogram is constructed, the only choice left is at which level to cut the tree to obtain the desired blocks of text. This is left as a parameter to be manually fine-tuned.

4.1.2 Classical Clustering

The extracted blocks from the previous step are then clustered according to the similarity of their shapes (width and height). This is done using K-means clustering for some chosen K, or with the DBSCAN algorithm.

Todo: Either expand this section or just integrate it with the previous subsection.



(a) Before

(b) After

Figure 5: An example of hierarical agglomerative clustering.

4.2 Supervised

This is written as a draft

Cite a survey mentioning these methods

After the data is augmented by the previously described clustering algorithm, it's fed into a supervised classifier. For the purposes of machine learning, text produces 3-dimensional data: there is a feature vector for each word, and some (either variable or predetermined through padding) amount of words per text. This makes each training sample a 2-dimensional matrix, which then gets stacked in the depth dimension to produce 3-dimensional training data. This is troublesome as the standard machine learning algorithms work on 2-dimensional data, assuming a feature vector for each sample rather than a matrix. There are three common methods to deal with this:

- Bag of words
- Convolutional neural networks
- Recurrent neural networks

Aside from being completely different methods, they differ in a major way in how they handle the sequential nature of text. The bag of words approach is the simplest in that it simply disregards this sequential nature, instead creating what is essentially a histogram of word occurrences. This downsamples each sample from a feature matrix to a feature vector, allowing the use of normal machine learning algorithms (commonly support vector machines). While the sequential information can be kept to some degree by use histograms of n-grams rather than words (unigrams), this causes the size of the input data to scale exponentially with the value of n.

Convolutional neural networks (CNNs) work by taking a number of filters (sometimes called kernels or feature maps) of a specified size and convolving these over the input data. A simplified example using one filter is shown in Figure 6. In this example, the input text is convolved with a filter with a width of 3 and a stride of 1 — that is, each application of the filter considers three subsequent input elements, after which the window is shifted one space to the right. This filter is essentially a small neural network mapping three items to one output value, whose weights are reused for each application of the filter. Reusing the weights in this way (weight sharing) prevents the number of parameters in the network from spiraling out of control. [8] After the application of this convolution layer, the responses of the filter form a new sequence of roughly the same size as the input (minus a few due to missing the edges). The next step is to downsample this sequence by means of a *max pooling* layer, which maps all values within a window to the maximum value amongst those values. While conceptually similar to a convolution, this step generally does not involve overlap, instead either setting the stride to the same value as the window size (usually 2) reducing the entire sequence to 1 value (1-max pooling). The reason for this is twofold:

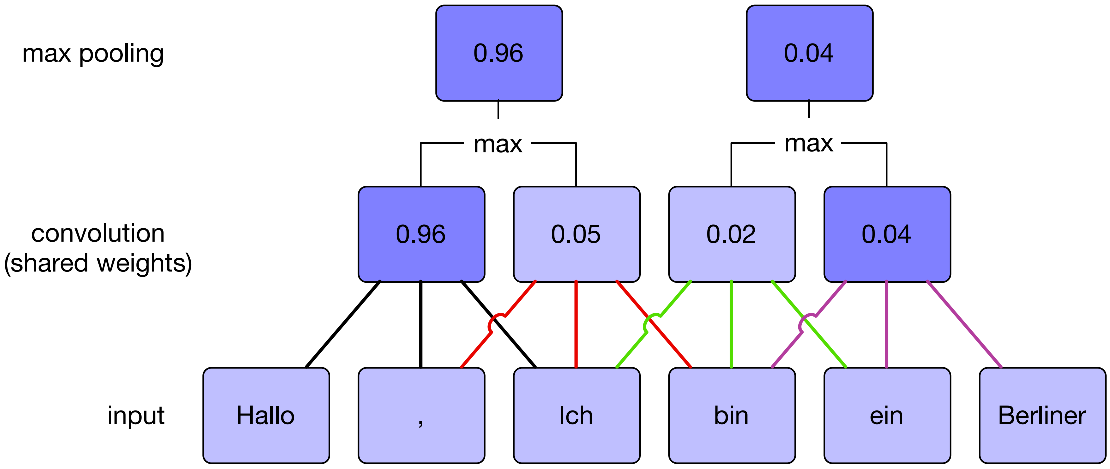


Figure 6: A simplified convolutional neural network

1. It downsamples the number of inputs, reducing the amount of parameters required further on in the network.

2. It adds translation invariance to the feature detected by this filter. The example filter of Figure 6 appears to react strongly to the presence of the word “Hallo”. Without the pooling layer, changing the location of the word “Hallo” in the input would similarly change the location of the high activation in the intermediate representation; this would be *equivariance*. The more aggressively the pooling is applied, the higher the degree of invariance.

This combination of convolution followed by pooling can be repeated multiple times as desired or until there is only a single value left as output from the filter. Finally, the outputs of all filters are concatenated and fed into a standard feedforward neural network.

While CNN architectures in computer vision are generally very deep, they tend to be very shallow in natural language processing; commonly just a single convolution followed by 1-max pooling [2]. Since this particular task at first glance appears to be fairly reliant on word position (e.g. a colon at the end of a sentence very often indicates the start of a speech, a colon in the middle almost certainly does not), the degree of pooling will be experimented with.

4.2.1 Difference between convolutional and recurrent neural networks

Recurrent neural networks (in particular LSTMs or GRUs) are seemingly the most natural fit for language processing, since they process an entire sequence and are therefore fully conditioned on the word order (as opposed to the convolutional neural networks which tend to learn translation invariant ngram features). Regardless, convolutional neural networks will be used in this research. This choice is based on two factors:

1. In practise, the performance for classification tasks does not differ between the two types of networks.[9]

2. The computations in convolutional networks are highly independent of each other, allowing for great paralellization (in particular with regards to running on a GPU). In contrast, LSTMs are bottlenecked by the fact that each calculation is dependent on the previous calculations. As a result, CNNs achieve far higher training speeds.[10]

Diagram/description of the full model with cluster labels added

5 Experimental Setup

Experiments are focused on the difference in performance between the baseline CNN model without clustering information (referred to from here on as `CNN`) and the model augemented with clustering information (which will be reffered to as `CNN-cluster`). Performance will be measured with regards to the following three metrics:

1. Number of training epochs until convergence
2. The F1 score or average precision metrics on a test set (see Section 5.2)
3. The number of training samples required to attain a specific F1/average precision score

In each case, the experiment will be repeated 10 times by means of 10-fold cross validation followed by a Student's T-test to gauge the probability of the following null hypothesis being true:

Null Hypothesis 1 *Adding clustering information to the CNN model does not change the performance of the model.*

5.1 Dataset

Referring back to Figure 2, the average document has between 100 and 300 positive samples. Since a secondary concern is to minimize the number of documents that would have to be annotated as training data, the tested dataset sizes will be very low, with the number of positive samples being one of 100, 200, 500 and 1000. Due to the relative abundance of negative samples and to prevent overfitting on the distribution of the labels, stratified sampling will be used to keep a 1:1 ratio of positive to negative samples. In addition to the size, the number of cluster types (the k in k-means) will be varied to examine its impact on the performance.

5.2 Testing performance

All models will be tested on a test set containing 1000 positive samples and 10000 negative samples, all of which are guaranteed not to be in the training set. Performance on this set is measured by constructing a precision-recall curve, and calculating two values:

1. The average precision (which is equivalent to the area under the curve)
2. The F1 score of the point on the curve maximizing the F1 score

5.3 CNN performance

Although less central to the thesis than the difference between the `CNN` and `CNN-cluster` models, some experimentation will be done with the parameters of the convolutional network in an attempt to optimize the performance. These parameters include the dimensionality of the word embeddings, the number of filters, the pooling strategy (1-max versus a smaller region) and the number of convolutional layers.

5.4 Generalisation

This particular dataset has the quirk that the performance of a rule-based system created based on recent documents decreases in performance when used on older documents, the older the document the worse it performs. This occurs despite the layout being visually the same all the way back to the 1950s. A number of files from old election periods has been labeled (and manually verified for correctness) in order to test

1. whether the CNN models handle this better than the rule-based system does.

2. Whether the clustering-augmented CNN model performs better on this task than the baseline CNN.

6 Evaluation

6.1 400 samples, local max pooling

Turn this into actual text

T-test probability that the AoC distributions are the same: 0.009790328758564197
T-test probability that the F1 distributions are the same: 0.011943604700946929

model	F1		AoC	
	mean	std	mean	std
CNN	0.79126	0.01644	0.856768	0.0219711
CNN-clusters	0.83429	0.02047	0.904359	0.0168642

Table 1: Performance metrics on a small dataset (200 positive samples)

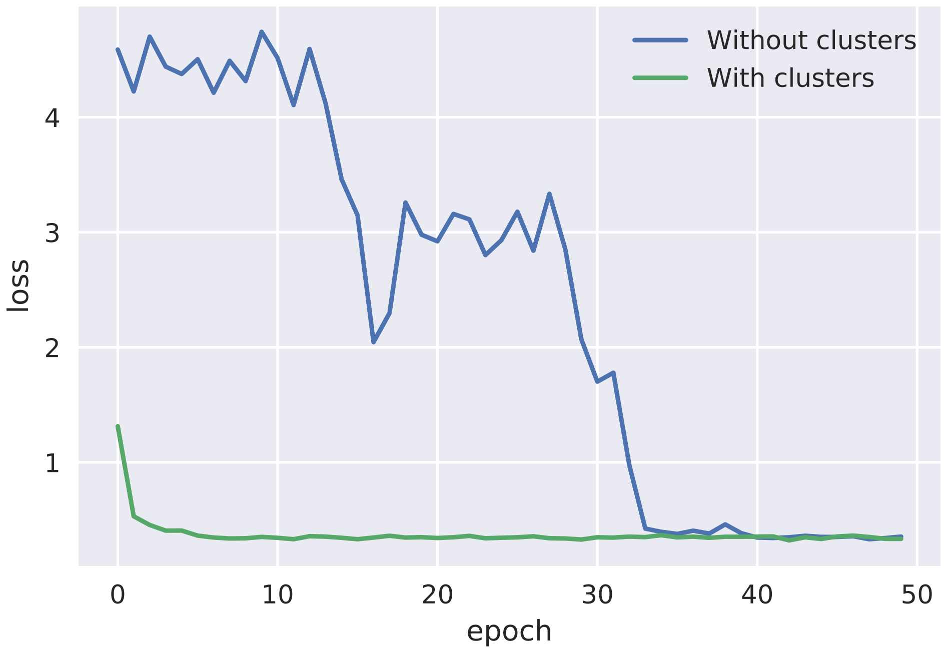


Figure 7: The convergence speed

289 6.2 400 samples, 1-max pooling

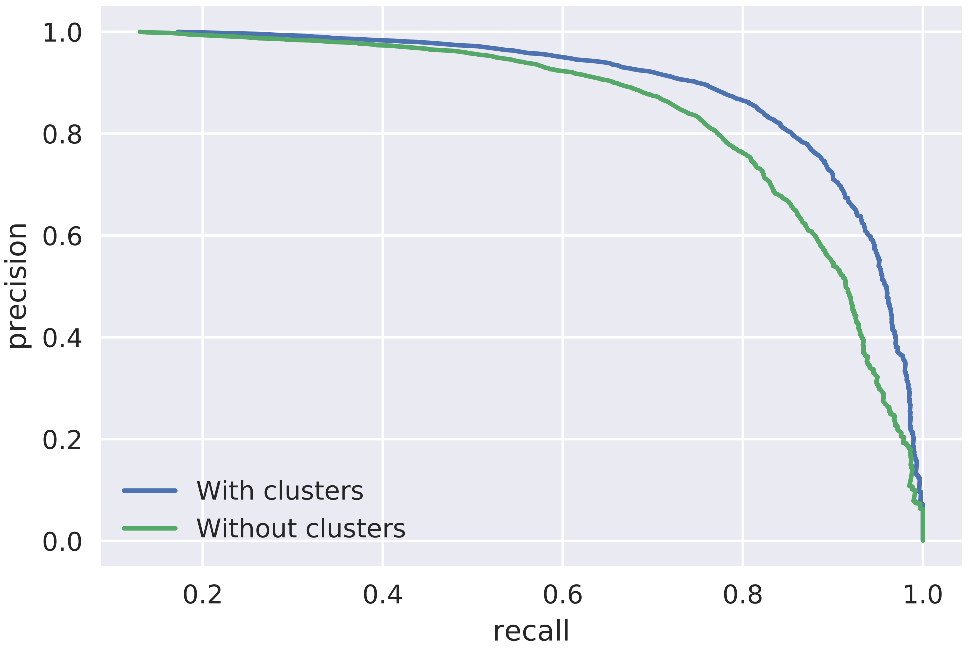
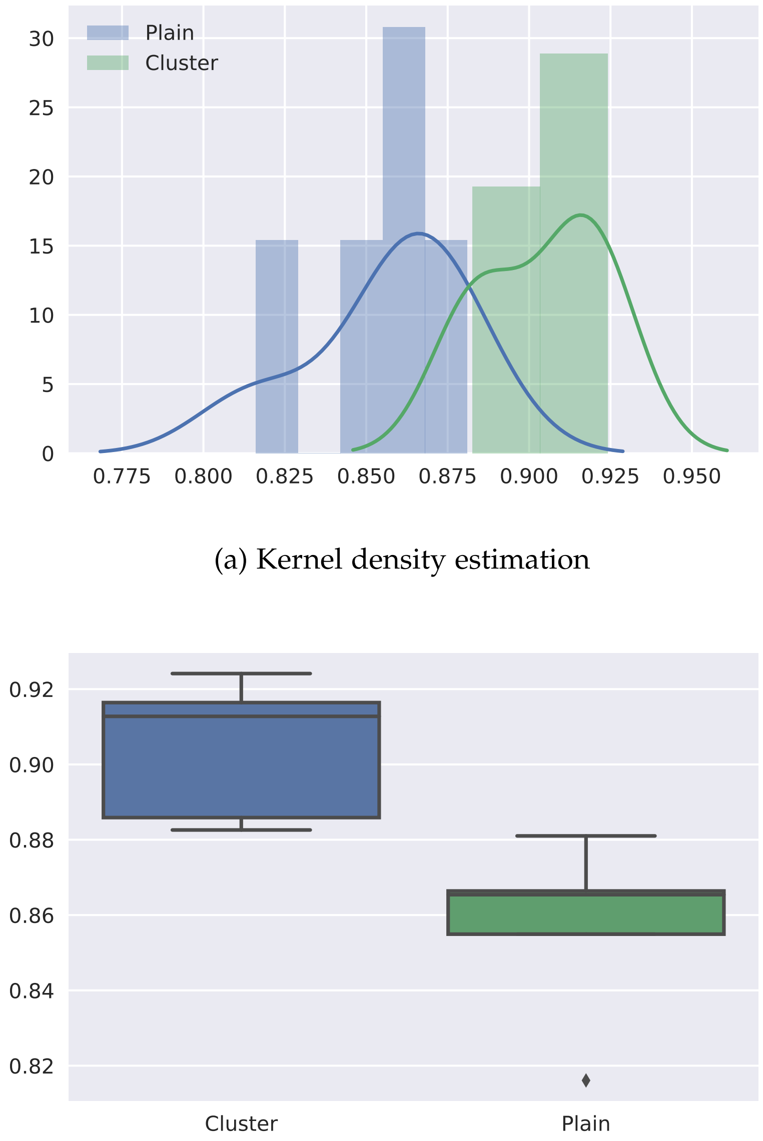


Figure 8: The precision/recall curves

model	F1		AoC	
	mean	std	mean	std
CNN	0.916071	0.00786046	0.934946	0.0183115
CNN-clusters	0.916407	0.00664848	0.931635	0.0270519

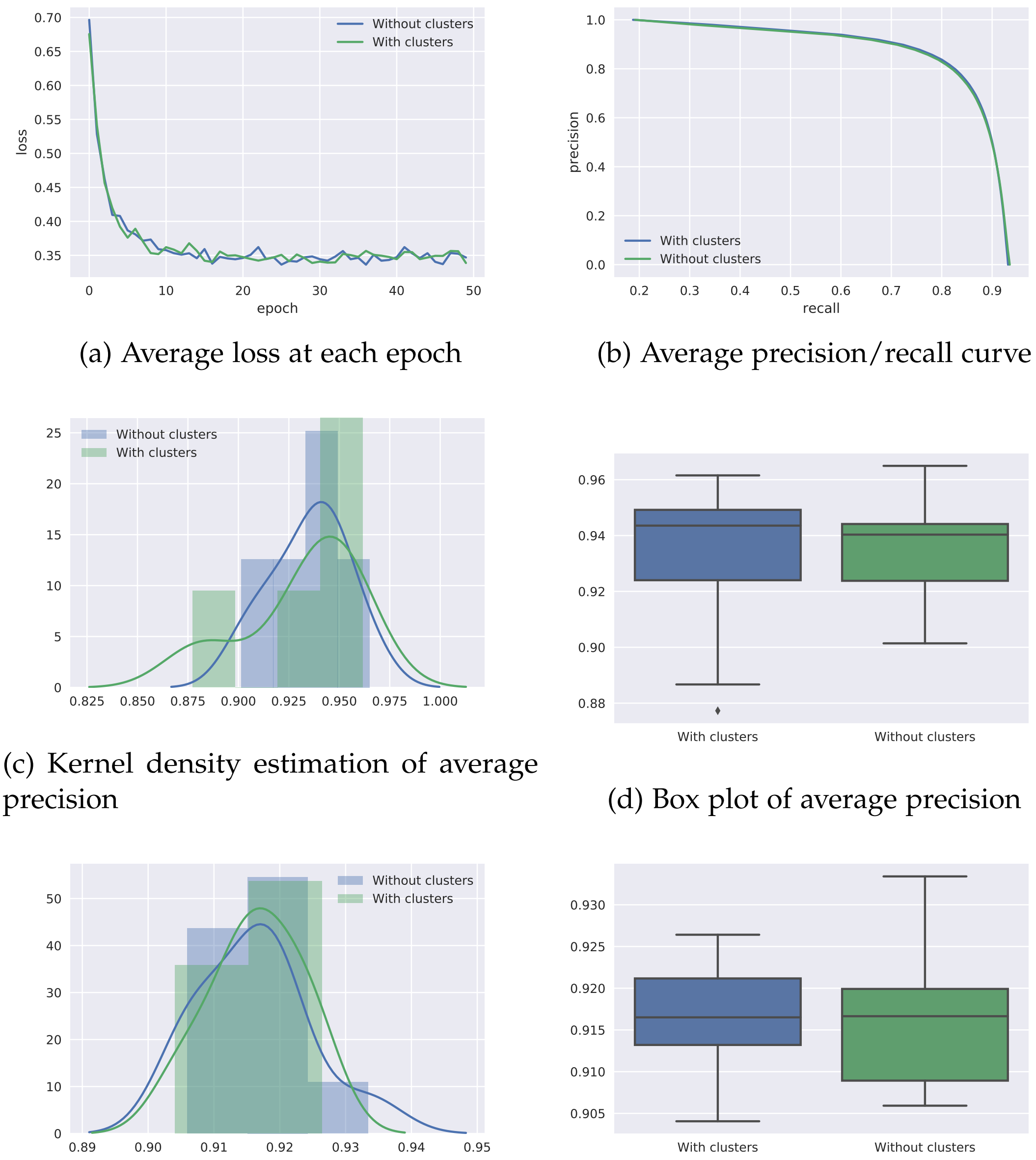
Table 2: Performance metrics on 200 positive and 200 negative samples with 1-max pooling. The T-test probability of the null hypothesis being true for the average precision is 0.765, for the F1 score it is 0.923. This makes it fairly save to say that for this configuration there is no benefit to the addition of clustering data.



(a) Kernel density estimation

(b) Boxplot

Figure 9: The distribution of the cross validation results



(a) Average loss at each epoch

(b) Average precision/recall curve

(c) Kernel density estimation of average precision

(d) Box plot of average precision

(e) Kernel density estimation of F1 score

(f) Box plot of F1 score

Figure 10: At 200 positive and 200 negative samples with 1-max pooling, there is no discernable difference between the two models.

7 Conclusion

Todo: conclusion

References

1. Kim, Y. Convolutional Neural Networks for Sentence Classification. *CoRR* **abs/1408.5882.** arXiv: 1408.5882. http://arxiv.org/abs/1408.5882 (2014).
2. Zhang, Y. & Wallace, B. C. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *CoRR* **abs/1510.03820.** arXiv: 1510.03820. http://arxiv.org/abs/1510.03820 (2015).
3. Zhang, X., Zhao, J. & LeCun, Y. *Character-level convolutional networks for text classification* in *Advances in neural information processing systems* (2015), 649–657.
4. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **abs/1512.03385.** arXiv: 1512.03385. http://arxiv.org/abs/1512.03385 (2015).
5. Conneau, A., Schwenk, H., Barrault, L. & LeCun, Y. Very Deep Convolutional Networks for Natural Language Processing. *CoRR* **abs/1606.01781.** arXiv: 1606.01781. http://arxiv.org/abs/1606.01781 (2016).
6. Klampfl, S., Granitzer, M., Jack, K. & Kern, R. Unsupervised document structure analysis of digital scientific articles. *International Journal on Digital Libraries* **14,** 83–99 (2014).
7. Sibson, R. SLINK: an optimally efficient algorithm for the single-link cluster method. *The computer journal* **16,** 30–34 (1973).
8. LeCun, Y., Bengio, Y., *et al.* Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* **3361,** 1995 (1995).
9. Yin, W., Kann, K., Yu, M. & Schütze, H. Comparative Study of CNN and RNN for Natural Language Processing. *CoRR* **abs/1702.01923.** arXiv: 1702.01923. http://arxiv.org/abs/1702.01923 (2017).
10. Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. Convolutional Sequence to Sequence Learning. *CoRR* **abs/1705.03122.** arXiv: 1705.03122. http://arxiv.org/abs/1705.03122 (2017).