



UNIVERSITY OF AMSTERDAM

MSC ARTIFICIAL INTELLIGENCE
MASTER THESIS

Document Structure Analysis By Means Of Sentence Classification

by
MAARTEN DE JONGE
10002109

May 2, 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	2
2	Related Works	4
3	Problem Statement	5
3.1	Dataset	6
3.2	Research Question	6
4	Methodology	9
4.1	Unsupervised	9
4.1.1	Hierarchical Agglomerative Clustering	9
4.1.2	Classical Clustering	12
4.2	Supervised	12
4.2.1	Difference between convolutional and recurrent neural networks	14
5	Experimental Setup	16
5.1	Dataset	16
5.2	Testing performance	17
5.3	CNN performance	17
5.4	Generalisation	17
6	Evaluation	18
6.1	400 samples, local max pooling	18
7	Conclusion	22

Chapter 1

Introduction

The Political Mashup project¹ 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 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

¹<http://search.politicalmashup.nl/about.html>

29 rather than left-aligned gives it away. Incorporating this structural data into the
30 learning process will hopefully increase the performance of the system, either
31 by simply scoring better on the used metrics, or perhaps more indirectly by
32 requiring less data or training time to achieve the same score.

Chapter 2

Related Works

Todo: Expand section

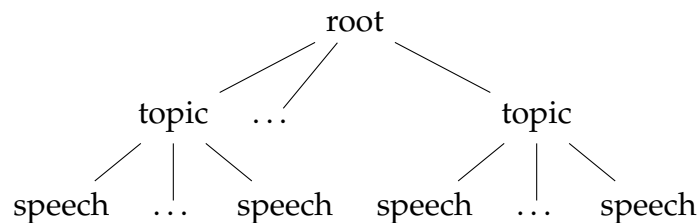
Various forms of convolutional neural networks are commonly used for text classification. The most basic architecture is described by Kim [1], where the input words are tokenized and embedded before passing them to the convolutional neural network. Additional exploration of the parameter space and its 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.

Chapter 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 library¹. 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 3.1 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 3.2 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?

¹<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 Alterspräsident, 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 Präsidenten)
</text>
```

(b) XML

Figure 3.1: The data representations

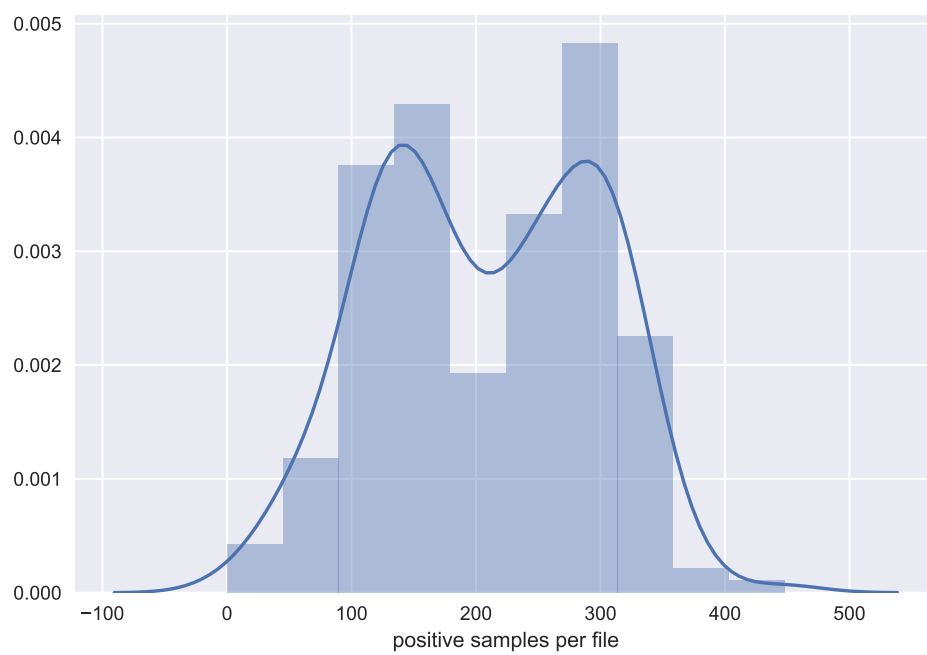


Figure 3.2: Distribution of the number of positive samples per file

Chapter 4

Methodology

The system consists of two separate parts; an unsupervised algorithm for augmenting data, and a supervised algorithm for classifying the data (Figure 4.1). 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)

4.1 Unsupervised

The unsupervised algorithm attempts to detect and label blocks of text in the PDF file, as shown in Figure 4.2 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.

4.1.1 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 4.3. The algorithm gets fed the individual characters present in

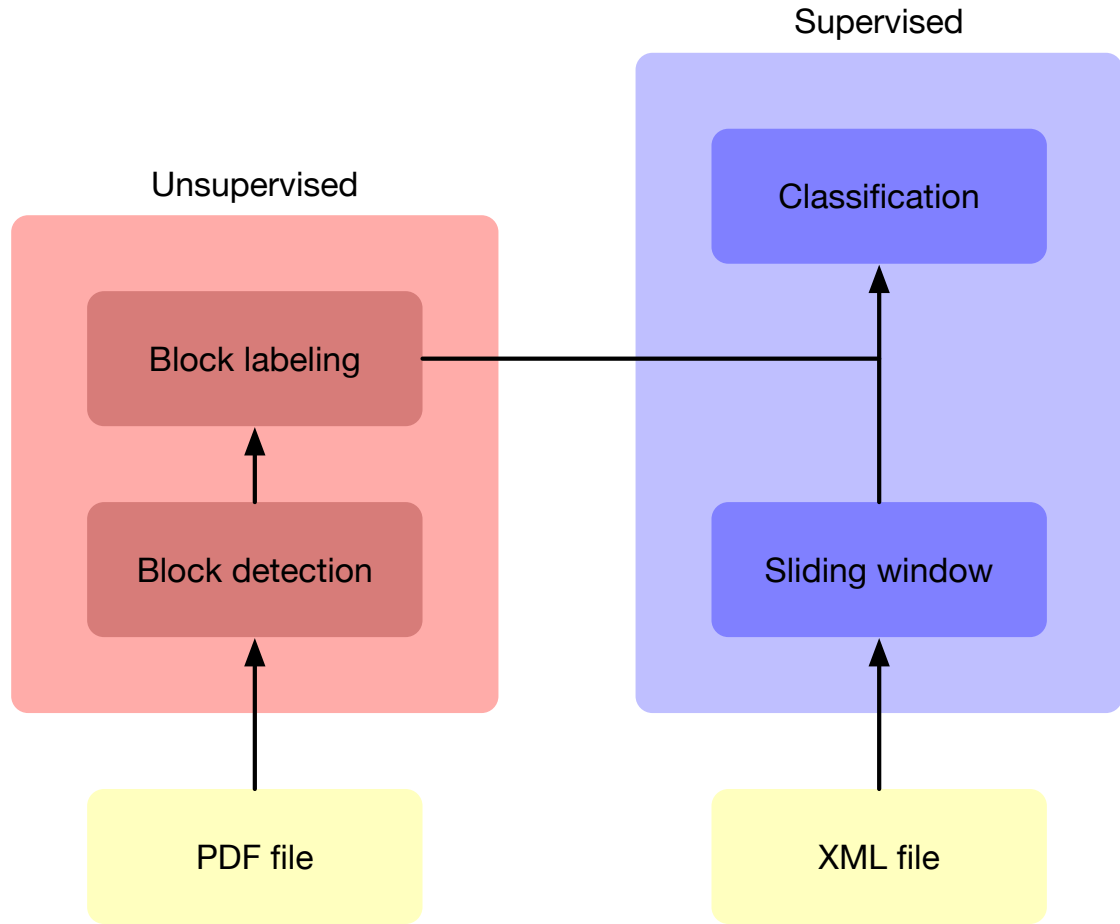


Figure 4.1: A high-level overview of the system

135 the PDF files, then iteratively groups the two closest clusters (the initial inputs
 136 being regarded as clusters of one element) together until only a single cluster
 137 remains. This process involves two parameters:

- 138 1. The distance function between two characters.
- 139 2. The distance function between two groups of characters.

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

- 143 • Single-linkage: The distance between groups is based on the closest two
 144 elements:

$$d(A, B) = \min\{d(a, b) : a \in A, b \in B\}$$

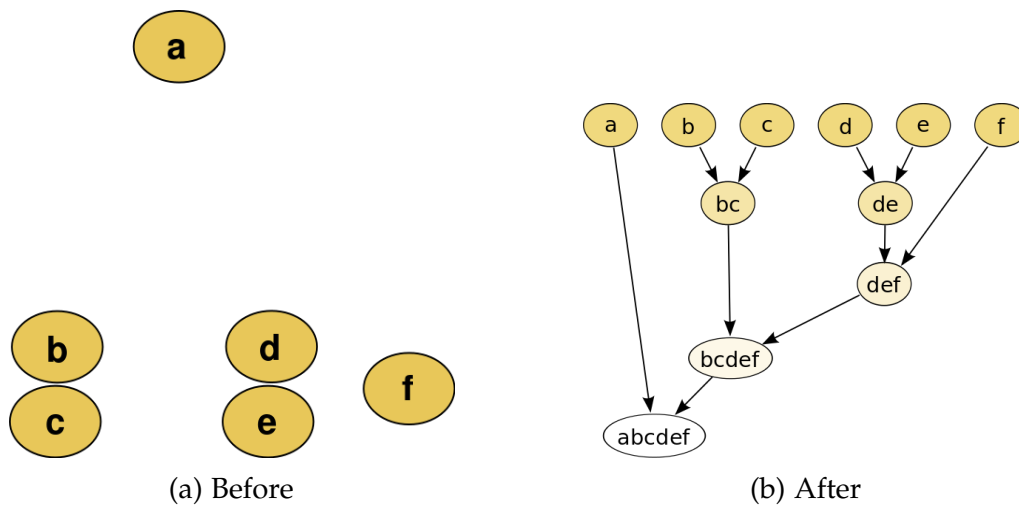


Figure 4.3: An example of hierarchical agglomerative clustering.

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

157 4.1.2 Classical Clustering

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

161 Todo: Either expand this section or just integrate it with the previous sub-section.

162 4.2 Supervised

163 This is written as a draft

164 Cite a survey mentioning these methods

165 After the data is augmented by the previously described clustering algorithm,
 166 it's fed into a supervised classifier. For the purposes of machine learning, text
 167 produces 3-dimensional data: there is a feature vector for each word, and
 168 some (either variable or predetermined through padding) amount of words
 169 per text. This makes each training sample a 2-dimensional matrix, which then
 170 gets stacked in the depth dimension to produce 3-dimensional training data.

171 This is troublesome as the standard machine learning algorithms work on
172 2-dimensional data, assuming a feature vector for each sample rather than a
173 matrix. There are three common methods to deal with this:

- 174 • Bag of words
- 175 • Convolutional neural networks
- 176 • Recurrent neural networks

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

186 Convolutional neural networks (CNNs) work by taking a number of filters
187 (sometimes called kernels or feature maps) of a specified size and convolving
188 these over the input data. A simplified example using one filter is shown in
Figure 4.4. In this example, the input text is convolved with a filter with a

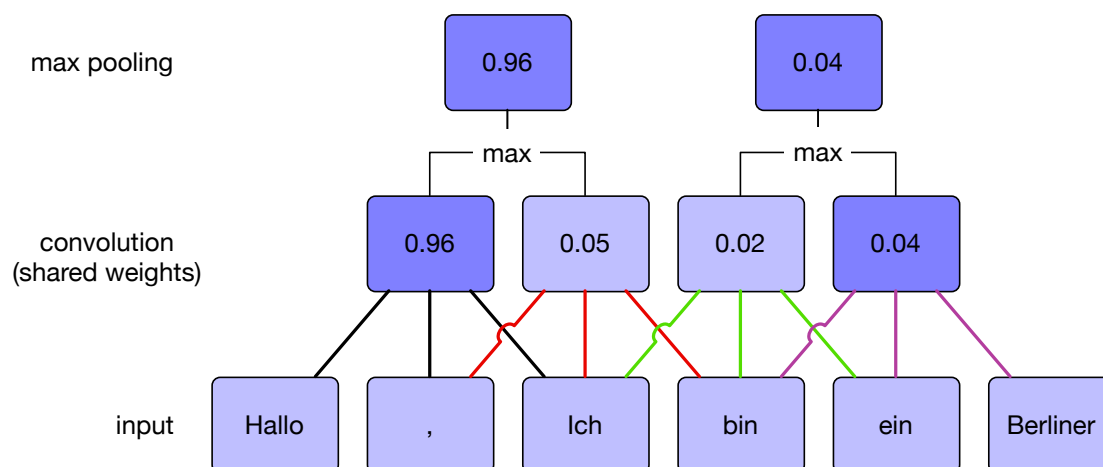


Figure 4.4: A simplified convolutional neural network

189 width of 3 and a stride of 1 — that is, each application of the filter considers
190 three subsequent input elements, after which the window is shifted one space
191 to the right. This filter is essentially a small neural network mapping three
192 items to one output value, whose weights are reused for each application of the
193

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:

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 4.4 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:

- 231 1. In practise, the performance for classification tasks does not differ between
232 the two types of networks.[9]
- 233 2. The computations in convolutional networks are highly independent of
234 each other, allowing for great paralellization (in particular with regards to
235 running on a GPU). In contrast, LSTMs are bottlenecked by the fact that
236 each calculation is dependent on the previous calculations. As a result,
237 CNNs achieve far higher training speeds.[10]

238 Diagram/description of the full model with cluster labels added

Chapter 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 augmented with clustering information (which will be referred 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 3.2, 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,

263 the number of cluster types (the k in k -means) will be varied to examine its
264 impact on the performance.

265 5.2 Testing performance

266 All models will be tested on a test set containing 1000 positive samples and
267 10000 negative samples, all of which are guaranteed not to be in the training set.
268 Performance on this set is measured by constructing a precision-recall curve,
269 and calculating two values:

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

272 5.3 CNN performance

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

279 5.4 Generalisation

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

- 286 1. whether the CNN models handle this better than the rule-based system
287 does.
- 288 2. Whether the clustering-augmented CNN model performs better on this
289 task than the baseline CNN.

Chapter 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 6.1: Performance metrics on a small dataset (200 positive samples)

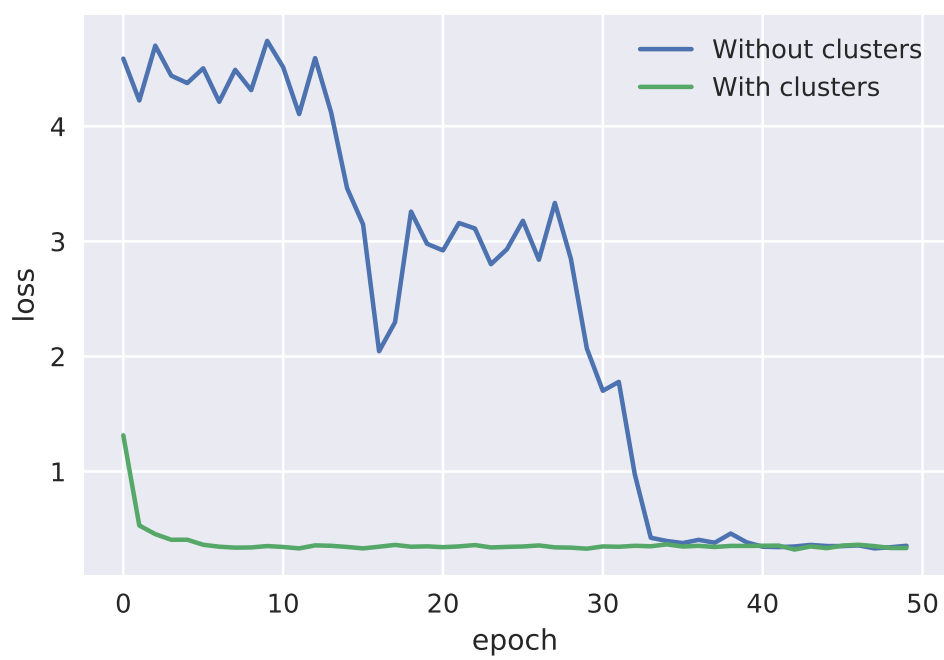


Figure 6.1: The convergence speed

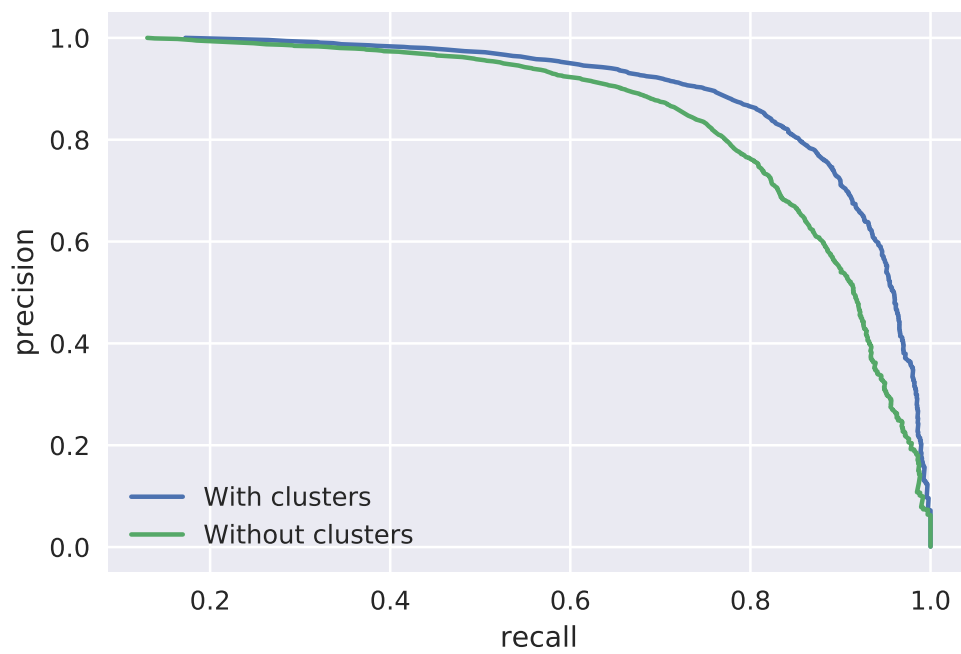
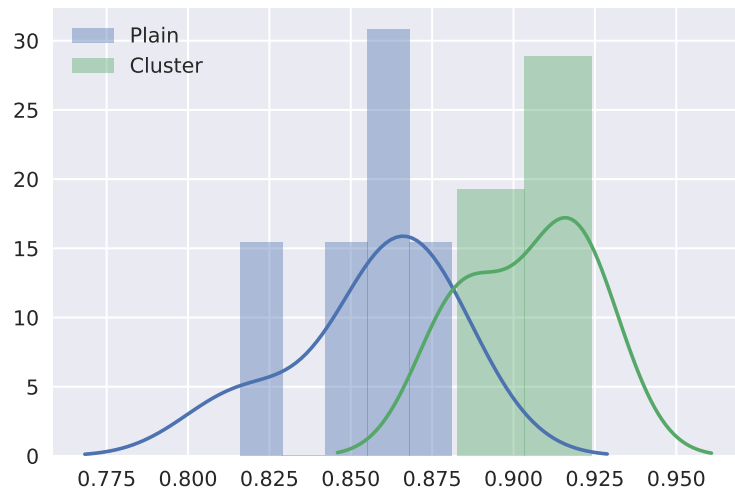
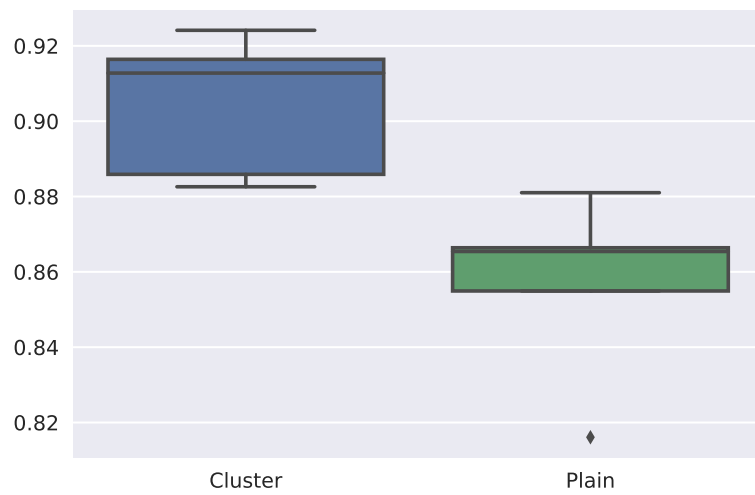


Figure 6.2: The precision/recall curves



(a) Kernel density estimation



(b) Boxplot

Figure 6.3: The distribution of the cross validation results

296 **Chapter 7**

297 **Conclusion**

298 Todo: conclusion

Bibliography

- 300 1. Kim, Y. Convolutional Neural Networks for Sentence Classification. *CoRR*
301 **abs/1408.5882**. arXiv: 1408.5882. <http://arxiv.org/abs/1408.5882>
302 (2014).
- 303 2. Zhang, Y. & Wallace, B. C. A Sensitivity Analysis of (and Practitioners'
304 Guide to) Convolutional Neural Networks for Sentence Classification.
305 *CoRR* **abs/1510.03820**. arXiv: 1510.03820. [http://arxiv.org/abs/1510.](http://arxiv.org/abs/1510.03820)
306 03820 (2015).
- 307 3. Zhang, X., Zhao, J. & LeCun, Y. *Character-level convolutional networks for*
308 *text classification* in *Advances in neural information processing systems* (2015),
309 649–657.
- 310 4. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image
311 Recognition. *CoRR* **abs/1512.03385**. arXiv: 1512.03385. [http://arxiv.org/](http://arxiv.org/abs/1512.03385)
312 [abs/1512.03385](http://arxiv.org/abs/1512.03385) (2015).
- 313 5. Conneau, A., Schwenk, H., Barrault, L. & LeCun, Y. Very Deep Convolu-
314 tional Networks for Natural Language Processing. *CoRR* **abs/1606.01781**.
315 arXiv: 1606.01781. <http://arxiv.org/abs/1606.01781> (2016).
- 316 6. Klampfl, S., Granitzer, M., Jack, K. & Kern, R. Unsupervised document
317 structure analysis of digital scientific articles. *International Journal on Digital*
318 *Libraries* **14**, 83–99 (2014).
- 319 7. Sibson, R. SLINK: an optimally efficient algorithm for the single-link cluster
320 method. *The computer journal* **16**, 30–34 (1973).
- 321 8. LeCun, Y., Bengio, Y., *et al.* Convolutional networks for images, speech,
322 and time series. *The handbook of brain theory and neural networks* **3361**, 1995
323 (1995).
- 324 9. Yin, W., Kann, K., Yu, M. & Schütze, H. Comparative Study of CNN
325 and RNN for Natural Language Processing. *CoRR* **abs/1702.01923**. arXiv:
326 1702.01923. <http://arxiv.org/abs/1702.01923> (2017).

- 327 10. Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. Con-
328 volutional Sequence to Sequence Learning. *CoRR* **abs/1705.03122**. arXiv:
329 1705.03122. <http://arxiv.org/abs/1705.03122> (2017).