Classifying Dutch municaplity documents using questions from parliament

A comparative study on the domain adaptivity of Convolutional Neural Networks, Support Vector Machines, Random Forest and Par2Vec

submitted in partial fulfillment for the degree of master of science

Axel Hirschel
10656146

master information studies
data science
faculty of science
university of Amsterdam

2018-06-21

	Internal Supervisor	External Supervisor
Title, Name	Dr Maarten Marx	Tom Kunzler
Affiliation	UvA, FNWI, IvI	Open State Foundation
Email	maartenmarx@uva.nl	tom@openstate.eu

Universiteit van Amsterdam

Contents

Abstract

Thesis requirements

- Your thesis is written in ACM style with two columns (`documentclass[sigconf]acmart`).
- It is maximally 10 pages long, excluding the title page and the appendix, but including references, figures, etc

1 Introduction

Since 2014 a part of all documents produced by Dutch municipalities are published as open data on http://zoek.openraadsinformatie.nl/, a website dedicated to this cause. Over time more municipalities have joined, and in 2018 also some provinces have also decided to participate. Although it is possible to search on specific words, more could be done in order to allow users to effectively query relevant information. This research is therefore dedicated to classify documents with labels describing their content, which users can use to only retrieve document belonging to that label.
These municipality documents have not been classified before, which means that it is not possible to train classifiers on data belonging to the domain. This means that the classifiers are trained on documents from a related domain, which are questions asked within national parlement. Although this data is similar, still classifiers have to adapt to the discrepencies in style, length and vocabulary between the two types of documents. Multiple classifiers are examined in order to evaluate how well the classifiers are suited for domain adaptation.
Many of these classifiers use the bag-of-words (BoW) representation of documents as input. This means relative occurences of words within labels are calculated and the documents of the muncipalities are classified based on these occurences. Examples of these classifers are Support Vector Machines (SVM), Naive Bayes (NB), Logistic Regression (LG) and Random Forest (RF) (Aggarwal & Zhai, 2012).
In contrast to the BoW new document representations have been developed which instead create multidimensional vectors for words which capture their semantic meaning such as Word2Vec (Mikolov, Chen, Corrado, & Dean, 2013) and Paragraph2Vec (Le & Mikolov, 2014). This type of representation is also know as word embeddings and they allow the use of deep neural networks, such as Covolutional Neural Networks (CNN), on text as well. The CNNs employ convolutional filters, which shift over the documents and detect patterns within documents (Kim, 2014) (Kalchbrenner, Grefenstette, & Blunsom, 2014).
The mentioned classifiers are all evaluated in how well they adapt to the new domain. Previous research suggested that algorithms based on word embeddings perform better because they partly capture the more nuanced meaning of words and also are able to use less frequently used words (Nguyen & Grishman, 2015) (Mou et al., 2016). Still methods exist in order to further increase performance of those classifiers. The most common technique is impotance sampling, which assigns weights to samples during the training based on how much the samples are similar to the test data (Pan & Yang, 2010). The effect of this technique on all classifiers is also studied which leads to the following research questions:

- How can Dutch municipality documents be classified with machine learning techniques?
  - How well can the various classifiers categorize data of parliament?
  - How well does that performance generalize to the municipality dataset?
  - What is the influence of importance sampling on the domain adaptivity?

- What type of classifiers are better suited for categorizing municipality documents?
- What are the characteristics of misclassified documents per classifier?

Within the next chapter, the literature review, current approaches to the mentioned challenges and the general idea behind the algorithms is further explained. Then the specific set-up for this research is discussed within the methodology, which also provides information on how the research questions are answered. The results of this research is described next, and provides a detailed overview of the performance of all algorithms with various evaluation metrics. Lastly, the answers to the research questions are formulated and the conclusions from this article are discussed.

2 Related Work

Classifcation has been a widely studied information problem and its various solutions are discussed within this section. The goal of classification is to assign labels to documents based on its contents. Depending on the task, this is either a binary classification or a multiclass classification problem. This research focusses on multiclass classification and even more specifically, multi-label classification. This means that documents can have multiple labels instead of merely one.
All classification is contingent on labeled train data, which is used to train a classifier in distinquishing between various classes. This means that classification of new documents is based on previous experience on other data. Often, a number of pre-processing steps are executed, in order to engineer a document representation which contains relevant content. A common representation is the bag-of-words representation, which is used to transform documents into a vector which indicates how many times words of the vocabulary occur within the text.
One of the algorithms that uses bag-of-words as input is the miltinominal naive bayes classifier, which uses probabilities of words occuring within a specific topic in order to classify unseen documents. One of the ways to increase performance is to extend the probabilities based on word occurences with other features, such as which organization published a document or on what kind of domain names documents were found (Sahami, Dumais, Heckerman, & Horvitz, 1998). Another version, very similar to Naive Bayes, includes document frequency within the bag-of-words vector, and thus divide the word-occurences within that specific document by the amount of documents in which that word occurs (Joachims, 1996).
Another frequently used algorithm for text classification is decision tree, which is an algorithm which establishes a hierarcal division based on textual features. These splits are based on features spaces which have a more skewed distribution of the classes, for example document in which certain words occur are often from a certain class. Multiple trees can be created as an ensemble, each on part of the data, to prevent overfitting to the train data and these are called random forest classifiers. Although the algorithm and its outcome are easily understandable its performance is often worse than other methods (Li & Jain, 1998).
The last method based on the bag-of-words implementation is the SVM, which

has been the state-of-the-art for many years. Within SVM hyperplanes are constructed that split datapoints within the multidimensional space. It is argued that SVM can perform well on textual data, since few features are relevant though those which are relevant correlate. This allows SVM classifiers to easily distinquish between various classes (Joachims, 2001).
All these algorithms based on bag-of-words have had specific classification problems in which they have excelled. However, the bag-of-words representation has a number of detriments. Foremost, the representation is unable to see the relation between "strong" and "powerful". The words are equally far apart as any random word such as "flower". Moreover, the representation treats documents as a collection of words instead of an ordered sequence. These two problems refrain algorithms to identify specific patterns and nuances in texts.
Recently, new document representations have been developed which use word embeddings. Word embeddings are multi-dimensional vectors that represent the semantical meaning of words. These embeddings are created using a neighbourhood approach, and therefore words that appear in similar contexts also are located closeby within the multi-dimensional space. (Mikolov et al., 2013) Documents can then be represented by ordering these word-vectors in the same sequence as original sentences.
This representation allows successful employment of neural networks for text classification, and two distinct types can be distinquished: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Although both types have achieved state-of-the-art results within various natural language processing tasks, CNN is expected to have better performance on classification tasks where pattern recognition is vital (Yin, Kann, Yu, & Schütze, 2017). Moreover, CNN can be trained faster than RNN, because it can be parallized on the GPU. Since keyphrases and similar structures seem to be important for this specific task, and because computational speed is important, CNN have been further explored.
CNN employ filters that are applied to local features and this has originally been used for processing image data. However, recent research has shown that CNN can also be used for natural language processing tasks. Kim et al. (2014) show that a CNN with Word2Vec-embeddings achieves excellent results, which suggests that the embeddings are good feature extractors. Their architecture consist of multiple pooling layers and 1D-convolutional layers and then two fully connected layers which produce the final output. They also experiment with multiple filter sizes in the convolutional layers, which also increases performance. Their last finding is that dynamically adjusting the embeddings boosts performance in many tasks.
CNN are limited to processing sentences with the same length, and most research solves this problem with padding and splitting sentences. This means that a fixed length is chosen as parameter, and sentences that are longer are cut off at that point. Empty vectors are padded to shorter sentences in order to reach the fixed length (Kim, 2014) (Zhang & Wallace, 2015). Another approach to deal with different sentence length is the Dynamic Convolutional Neural Network, which allows different sentence-sizes as input. After convolution layers a dynamic k-max-pooling layer, which computes the amount of features pooled to the next layerly based on the sentence's length. Only within the last pooling layer a fixed number of features is pooled and used within the last dense activation layer (Kalchbrenner et al., 2014). This architecture is only

suitable for sentences but has been extendend to entire documents as well. This algorithm uses the output from the last layer of the Dynamic Convolutional Neural Network employed on each sentence. Then based on the length of the document these features are then once again used in a Dynamic Convolutional Neural Network (Denil, Demiraj, Kalchbrenner, Blunsom, & de Freitas, 2014). Domain adaptation.....................

3 Methodology

This section is specifically aimed at explaining how the differences in performance between a CNN-classifier and other methods are measured. This entails that the data, the experiments and the algorithms are all examined in order to show how the research question is answered.

3.1 Description of the data

The data used within this project consist of two types of data, both from the government of the Netherlands. The first set consists of over 50,000 questions with answers that have been asked within the Dutch national parliament. Each question-answer pair is annotated with a number of labels. Each of the labels then consists of two levels of detail; it has one of the 17 undetailed labels, such as law or education, and one of the 118 more detailed labels, such as criminal law or primary education. Figure 1 shows the distributions of labels within the dataset and figure 2 shows how many labels each document contains.

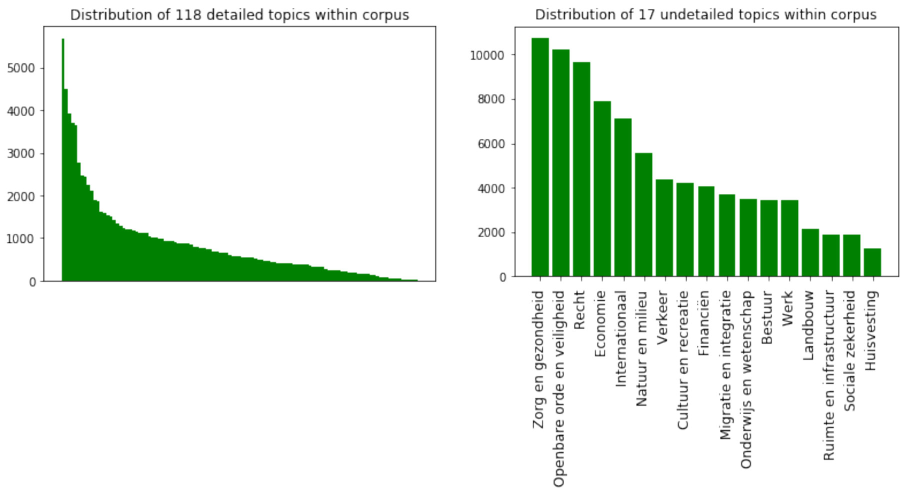

Figure 1: Distribution of labels within the parlement data

The questions are collected from www.zoek.officielebekendmakingen.nl with a scraper and the set consist of all question asked from 2001 to 2017. This means all of the labels have been discussed in a wide variety of manners. Moreover, the documents vary on length as can be seen in Figure 3 and which politicians have asked and answered these questions.

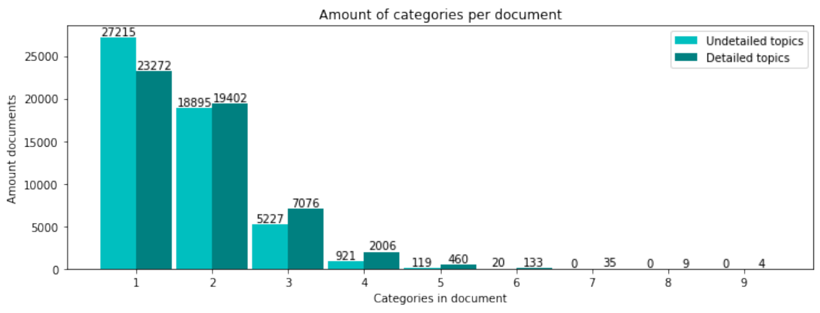

Figure 2: Amount of labels per document

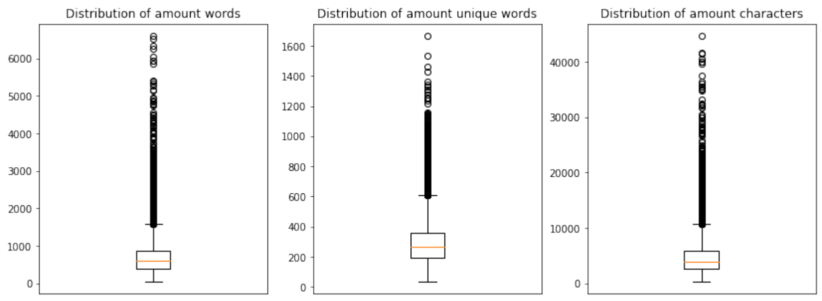

Figure 3: Box plots of the amount of words, unique words and characters within the train data.

The second part of the data is dat of municipalities retrieved from www.openraadsinformatie.nl. ....

3.2 Algorithms

The CNN of this research are compared to a number of traditional text classifiers; Support Vector Machines (SVM), Logistic Regression(LR), Random Forrest (RF) and Multinominal Naive Bayes (NB). For all these implementations the input is a bag-of-words representation of the various documents. The implementation of Scikit-Learn is employed for these algorithms and the optimal hyper-parameters are chosen based on a grid-search. Moreover, since SVM, LG and NB are not suited for multilabel they are employed with a one-versus-all classifier as well. This means that per label one classifier is built which independently of the other classifiers and labels predicts whether a sample belongs to that label.
The CNN within this research are similar to earlier architectures (Kim, 2014).

This means that an embedding layer is used to transform sentences to a multidimensional space using Word2Vec-embeddings. This research experiments with two embeddings, namely pre-trained embeddings retrieved trainedon a variety of Dutch resources (Tulkens, Emmery, & Daelemans, 2016) and embeddings trained on the data described in section 3.1. Both embedding spaces have been tested with static and non-static initializations. Thereafter three convolutional layers and three max-pooling layers are alternated between. For the convolutional layers multiple filter sizes have been tested and in addition also multiple filter sizes in one layer are experimented with.
Then the multidimensional is flattened and a dropout layer is used to prevent overfitting within the network. In some architectures also L1-regularization has been used, however, later research demonstrated the minimal effect of this regularization (Zhang & Wallace, 2015). The last two layers are fully connected layers in order to gain the final prediction. Since the classification task is multilabel, binary crossentropy is used as loss function in combination with the sigmoid function as activation within the final dense layer (Nam, Kim, Mencía, Gurevych, & Fürnkranz, 2014). The output of the sigmoid function is a number between 0 and 1 per label, and multiple thresholds are tested in order to determine when to predict a certain label. All the parameters of this standard version of CNN are listed in Table 1

Table 1: Parameter and values within standard CNN

Parameter	**Values**
Word2Vec Model	Pre-trained, trained on corpus
Word2Vec Initialization	Dynamic, Static
Amount of filters	1,3
Filter sizes	3,4,5 (combined whem amount of filters = 3)
Threshold of prediction	0.3,0.4,0.5,0.6,0.7

A different approach which deals with the length discrepancy of the documents is splitting the sentences up into smaller parts. Each of these individual parts of the sentence are classified individually. Then these predictions aggregated into one prediction, either by summing or using the max value. Then once again predictions above a certain threshold are predicted to belong to that label, and those below that threshold are considered not to have that label. All the parameters of this variation of CNN are listed in Table 2.

Table 2: Parameter and values within split version of CNN

Parameter	**Values**
Input size	200, 400, 800
Word2Vec Model	Pre-trained, trained on corpus
Word2Vec Initialization	Dynamic, Static
Amount of filters	1,3
Filter sizes	3,4,5 (combined whem amount of filters = 3)
Aggregation	sum, max
Threshold of prediction	0.3,0.4,0.5,0.6,0.7

3.3 Experiments

The various algorithms are evaluated on the basis of two test-sets, both from a different source of data as explained in section 3.1. The first experiment is carried out on the data with question of the Dutch parlement. This set is split into three parts of respectively 70, 20 and 10 percent of the data. The models are firstly trained on 70 percent of the data. Then, the optimal hyperparameters, such as the decision threshold, are chosen by evaluating the performance on the 20 percent of the data. When these parameters have been selected, the final versions is tested with the last 10 percent of the data. This experiment is conducted twice, using both the data with 17 and 118 different topics. Using different amount of topics shows how well algorithms perform depending on the detail of the topics.
Within the second experiment the transfer between different datasets is specifically important. The model is trained on 80 percent of the parlement-data and the remaining parlement data is used as validation data to select the optimal parameters. However, this model is evaluated using the manually labelled dataset of the Dutch municipalities in order to see how well the models transfer to another dataset. In contrast to the first experiment this experiment is merely conducted with 17 topics, as the municipality data is only classified that way.
Success in both experiments is measured using the micro-average F1-score, which balances the precision and recall of the prediction. However, in addition to the F1-scores also confusion matrices are used in order to evaluate what kind of errors are made. Lastly, properties of documents that are missclassified are evaluated per algorithm to better understand how the algorithms perform on specific types of documents.

4 Evaluation

Met een subsectie voor elke deelvraag.
In hoeverre is je vraag beantwoord?
Een mooie graphic/visualisatie is hier heel gewenst.
Hou het kort maar krachtig.

5 Conclusions

Hierin beantwoord je jouw hoofdvraag op basis van het eerder vergaarde bewijs.

Table 3: Performances of algorithms on central government data

Model	**Accuracy**	**Micro-F1**	**Micro-Recall**	**Micro-Precision**
Random Forest	0.27	0.45	0.30	0.92
SVM	0.55	0.79	0.72	0.84
Logistic Regression	0.54	0.77	0.72	0.84
Multinominal Naive Bayes	0.06	0.09	0.05	0.98
CNN	0.40	0.71	0.71	0.70

5.1 Acknowledgements

Hier kan je bedanken wie je maar wilt.

References

Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In *Mining text data* (pp. 163–222). Springer.

Denil, M., Demiraj, A., Kalchbrenner, N., Blunsom, P., & de Freitas, N. (2014). Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*.

Joachims, T. (1996). *A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.* (Tech. Rep.). Carnegie-mellon univ pittsburgh pa dept of computer science.

Joachims, T. (2001). A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 128–136).

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *The Computer Journal*, *41*(8), 537–546.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., & Jin, Z. (2016). How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*.

Nam, J., Kim, J., Mencía, E. L., Gurevych, I., & Fürnkranz, J. (2014). Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 437–452).

Nguyen, T. H., & Grishman, R. (2015). Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th inter-*

national joint conference on natural language processing (volume 2: Short papers) (Vol. 2, pp. 365–371).

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for text categorization: Papers from the 1998 workshop* (Vol. 62, pp. 98–105).

Tulkens, S., Emmery, C., & Daelemans, W. (2016, may). Evaluating unsupervised dutch word embeddings as a linguistic resource. In N. C. C. Chair) et al. (Eds.), *Proceedings of the tenth international conference on language resources and evaluation (lrec 2016).* Paris, France: European Language Resources Association (ELRA).

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.