Classifying Dutch municipality 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

Classifying Dutch municaplity documents using qestions from parliament

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

Axel Hirschel
Universiteit van Amsterdam
Amsterdam, Netherlands
axelhirschel@gmail.com

ABSTRACT

An increasing amount of municipality documents are released as open data. To easily find relevant information these documents need to be classified with labels describing their content. However, no labeled municipality data is available thus data from parliament is used for training. A domain shift between the source domain and target domain is thus present. This research evaluates how the algorithms Par2Vec, Logistic Regression, Convolutional Neural Networks (CNN), Support Vector Machines and Random Forest are capable of overcoming the domain shift. Moreover, the effects of importance sampling on these algorithms are examined. This research discovered that CNNs are better equipped to overcome bigger domain shifts, largely because the word embedding representation of texts mitigates the effect of domain shift. However, both Logistic Regression and Support Vector Machines have outperformed the CNNs on smaller domain shifts. Importance sampling has a negative effect on CNNs but a positive effect on the other algorithms.

KEYWORDS

Text classification, Convolutional Neural Networks, Domain Adaptation, Multilabel classification

1 INTRODUCTION

Since 2014 a part of all documents produced by Dutch municipalities are published as open data on www.zoek.openraadsinformatie.nl, a website dedicated to this cause. Over time more municipalities have joined, and in 2018 some provinces have also decided to participate. Although it is possible to search on specific words, more could be done to allow users to effectively query relevant information. This research is therefore dedicated to classifying documents with labels from the TaxonomieBeleidsagenda [14]. These labels describe which broad topics, such as education or healthcare, are discussed. Users can then select labels and retrieve documents belonging to that label on the website.

The municipality data is unlabeled and thus the classifiers cannot be trained on the municipality data itself. This problem is considered a domain adaptation problem; the target data, in this case the municipality data, must be trained on source data from a different domain. The source and target data differ because they are not drawn from the same feature space or the same distribution. This means different words are used in the data-sets, and the same words are not necessarily predictive for the same labels in both data-sets. In general, when data from a different domain is used for training, the classification algorithms are likely to have a lower performance than when it would have been possible to use data from the municipality domain. [13]

To retain good performance source data similar to the target data should be selected, which ensures a small domain shift. Therefore the classifiers within this research are trained on documents from a related domain to the municipality documents, namely questions asked within the Dutch national parliament. Although the source data is similar, still classifiers must adapt to the discrepancies in style, length and vocabulary between the two types of documents. In addition to choosing similar data, methods for domain adaptation are developed to overcome the domain shift. One type of methodology to do domain adaptation is the transfer of feature representation, which is aimed at finding a common representation for the features of the source and target set [13]. Examples of feature representation transfers are dimensionality reductions of documents, removing domain specific words from both datasets and the use of word embeddings.

Another method of domain adaptation is to transfer the knowledge of instances, and this is aimed at re-weighting samples in the source set based on its predictive value for the target set [13]. The most suitable technique is importance sampling, which is used to assign weights to samples during the training based on how similar the train samples in the source domain are to the test samples in the target domain [13].

Both techniques are studied with a number of classifiers and many of these use the bag-of-words (BoW) representation of documents as input. When BoW is employed, documents are represented as vectors, which contain the counts of words within those documents. Examples of classifiers that employ BoW are Support Vector Machines (SVM), Naive Bayes (NB), Logistic Regression (LG) and Random Forest (RF) [1].

Other document representations have also been developed which instead create low dimensional vectors for words paragraphs and documents which capture their semantic meaning. Examples of these low dimensional vectors are Word2Vec [9] and Paragraph2Vec [7]. Low dimensional word vectors are also known as word embeddings and the word embeddings are necceasary to employ deep neural networks, such as Convolutional Neural Networks (CNN), on text as well. The CNNs employ convolutional filters, which shift over the documents and detect patterns within documents [5, 6]. Since the CNNs use word embeddings they are expected to are

better suited for domain adaptation, as they already use a transfer of feature representations between the source and target set. Previous research also suggested that algorithms based on word embeddings perform better because the embeddings capture the more nuanced meaning of words and can also use words that are not within the training data [10, 12]. Within this research both the transfer of instances and the transfer of feature representation are examined with a variety of text classifiers to understand what the effect of domain adaptation is on each algorithm. These effects are studied on two different domain shifts 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 all algorithms?
  - Are the best methods consistent regardless of the degree of domain shift?

Within the next chapter, relevant literature to this research is examined which provides a more detailed overview of the classification task and the employed techniques. Afterwards the methodology is discussed, which also provides information on how the research questions are answered and how the algorithm are exactly implemented. The results are described next and provide a detailed overview of the performance of all algorithms. Lastly, the answers to the research questions are formulated and the conclusions are discussed.

2 BACKGROUND

Within this section related research is discussed. The task of classification is discussed first, together with the regularly used classification algorithms. Secondly, it is explained what multilabel classification is and how the algorithms should be adapted. Lastly, the relevance of domain adaptation for this research is examined.

2.1 Classification

Within classification a training set D of length N is present, and each sample has a label from a finite set of labels L assigned to it. This data is used to construct a classification model (also known as a classifier) using classification algorithms, which relate features of samples within D to labels in L [1].

To evaluate how well the classifiers can predict labels, a part of D is used as train data. The data which was not used for training, also known as test data, is now fed into the established classifiers. For evaluation these classifiers predict the samples in the test data, and the predictions are compared to the labels that were assigned to the samples. The classifiers are optimized to maximize the correct predictions which means that the predicted label of a document corresponds to the label which was assigned to it [1].

Two approaches are used for classification, namely semantic and lexical methods. Lexical classifiers are based on the occurrences of words, and thus rely on exact matches of words in test and train to predict certain labels. In contrast, semantic classifiers try to capture the meaning of documents, and base their prediction on the semantic meaning of documents. Both types of classifiers are contingent on a specific representation of documents, which are used to train the classifiers. For lexical classifiers BoW representations are used and semantical classifiers employ word embeddings.

2.1.1 *Lexical classifiers.*

Lexical classifiers employ the BoW representation of documents, which implies that documents in D are transformed into vectors which indicate how many times words of the vocabulary occur within the documents [1]. Often these vectors are then modified with TF-IDF, which scales the BoW-reprentation in such a way that the relative occurences of words are collected instead of raw counts. One of the algorithms that uses bag-of-word vectors as input is the the decision tree classifier, which is an algorithm that establishes a hierarchal division based on textual features. These splits are based on feature spaces which have a more skewed distribution of the classes. 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 interpreted its performance is often worse than other methods [8].

Logistic regression is also used as classifier, and it employs a statistical approach for classification. Its prediction is based on the multiplication of parameters and variables. During training these parameters are optimized with gradient descent in such a way that as many samples are classified correctly with high probability. To predict unseen samples, it takes a logistic function of the multiplication of the parameters with the variables of the unseen sample, which is a number between 0 and 1. A label can then be assigned to samples when the number between 0 or 1 is higher than a certain threshold [1].

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 SVMs can perform well on textual data, since few features are relevant though those which are relevant correlate. This allows SVM classifiers to easily distinguish between various classes [4].

Although SVM and logistic regression perform well, they are naturally binary classifiers. This means these classifiers can only be used to distinguish between two labels instead of having any amount in labels within L. However, this problem can be easily solved by constructing one-vs-rest classifiers, one for every label in L. All these classifiers produce a new binary model for one label against all other labels. This means that all samples in D with that label are considered as positive samples and all the other samples without the label as negative samples for the training of the model. After training these models can be used to assign a label to unseen samples by applying all classifiers to new samples and predicting a label from L for the classifier which reports the highest score [1].

The algorithms based on bag-of-words have performed well on many classification tasks. However, the bag-of-words representation has a few detriments. Foremost, the representation is unable to see the relation between "strong" and "powerful" as these words are equally far apart as they are different from any other word such

as "flower". Moreover, the representation treats documents as a collection of words instead of an ordered sequence of words. These two problems refrain algorithms to identify specific patterns and nuances in texts [9].

2.1.2 *Semantic classifiers.*

Recently, new document representations have been developed which use word embeddings. Word embeddings are low dimensional vectors that represent the semantical meaning of words. Word2Vec, one of this methodologies, is created using a neighbourhood approach, and therefore words that appear in similar contexts are located near each other within the multidimensional space. [9] Therefore the distance of between the vectors of similar words is smaller than those of words with another meaning. This means that embeddings of the words "strong" and "powerful" are very similar. Documents can then be represented by ordering these word vectors in the same sequence as original sentences.

This representation allows successful employment of deep neural networks for text classification, and two distinct types can be distinguished: 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, CNNs are expected to have better performance on classification tasks where pattern recognition is vital [18]. Moreover, CNNs can be trained faster than RNNs, because it can be processed in parallel on the GPU. Since key phrases and similar structures seem to be important for this specific task, and because computational speed is important, CNNs have been further explored within this research.

Within CNNs filters are applied to local features and this has originally been used for processing image data. However, recent research has shown that CNNs can also be used for natural language processing tasks and CNNs with Word2Vec-embeddings achieve excellent results. This suggests that the embeddings are good feature extractors. A common architecture consists 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.

Although this architecture has performed well, it is limited to classifying documents with the same length. Most researchers have solved this problem with padding and splitting documents. This means that a fixed length is chosen as parameter, and sentences that are longer are cut off at that point. Vectors with only zeros, and thus meaning, are appended to sentences that are shorter to get to the fixed length [6, 19].

Another architecture, which deal with different document's lengths in a more sophisticated way, is the Dynamic CNN, which allows different sentence-sizes as input. This is possible because both convolution and pooling layer do not need a fixed input, but the dense layers do. Therefore the Dynamic CNN employ k-max pooling layers, which are very similar to max pooling layer. The k-max pooling layer pools the k most active features whereas the normal pooling layer only transfers one. However, this k-max pooling layer is even further developed into the dynamic k-max pooling layer [5]. This means that the amount of features pooled is decided by a function based on the document's length as shown in equation 1:

$$k_l = max(k_{top}, \lceil \frac{L-l}{L} \rceil s) \tag{1}$$

Within this function k_l is the amount of features pooled in this layer. This is decided by the max-function of k_{top}, which is the pre-determined amount of features pooled after the last convolutional layer. Then L is the total amount of layers, l is the current layers and s is the length of the input document. Using this formula ensures that a fixed number of features is used in the latter dense activation function, which is similar to the activation function of the previously discussed CNN [5, 6].

2.2 Adaptions for multilabel classification

The challenge within this research is a multilabel classification task. Instead of giving one label from the finite set L now any number of labels can be assigned, including no label at all. This means that the training data D also contains samples with varying numbers of labels assigned to it. All previously discussed algorithms can be used for multilabel classification as well when adapted [2].

The algorithms within this research all output probability scores between 0 and 1 for every sample. So instead of choosing the label with the highest score, a threshold can be defined, such as 0.5. Now samples are considered part of a class if the independent probability of it belonging to that class is higher than the threshold.

In addition to this, for CNNs the final activation of the last connected layer should be changed from "softmax" to "sigmoid". The softmax function sees the outputs of the last layer as dependent, which means that the total probability is always 1. With the sigmoid function independent probabilities for each label can be calculated. Many people also employ a different loss function to optimize multiclass neural networks, such as binary cross-entropy instead of multiclass cross-entropy [11, 15].

2.3 Domain adaptation

Transfer learning is the discipline dedicated to transferring models from one domain to another. One typical problem of transfer learning is transductive transfer learning or domain adaptation, which formally can be described as learning a task on the source training set D_S. The goal is then to create a model that minimizes the error of the same task on a target test set D_T. D_S and D_T are not from the same domain which means that either the feature space is not the same or the probability distribution of the features is not the same. Without adaptation classification models perform relatively bad on the D_T, luckily a number of methods exist to mitigate these effects. It is assumed that unlabeled documents form D_T are available during training [13].

One of these methods is a transfer of feature representations, and the use of Word2Vec can already be seen as an example of this type of adaptation [12]. The reason why this is effective is because semantic methods do not merely rely on string matches of documents but instead are able to match based on semantic meanings of documents. Other examples of feature representation transfers are some dimensionality reduction algorithms employed on the bag-of-words representations and finding domain specific features and removing those. These methods ensure that algorithms can be trained on features that are present in both set and thus can be employed to overcome part of the domain shift [13].

Another way to do feature representations transfers is the autoencoder, which is an unsupervised neural network. The idea of the autoencoder is to find a lower dimensionality space with which you can construct the original sample. However, the local structure is lost during the process, which is why it is less suited for combination with CNNs [3].

Another type of transfers suited for domain adaptation is transferring the knowledge of instances. The most common way to do this is with importance sampling, which implies that training samples in D_S that are similar to test samples in D_T are more important during training. This is done by constructing a classifier which recognizes whether samples come from D_S or D_T, and this is possible because some unlabeled documents from D_T are available. There are many ways to determine which set a sample belongs to, but for this research a simple logistic regression model is used.

During training on D_S for the classifier of the eventual task, samples in D_S are weighted based on how much the previously constructed classifier predicts them to be part of D_T [13]. This is successful because features in D_S which are similar to D_T are more important in training.

Although normally no labeled data from D_T is available, some research is done exploring the effect of retraining some layers within CNNs with small amount of data from D_T. These findings prove that directly transferring the models trained on D_S to D_T is less effective than training those models on a small set of labeled data in D_T. However, they also show that initializing the model on D_S and then fine-tuning the layers of the CNN on D_T actually works best [10]. For this research this is unfortunately not possible, as not enough labeled data from D_T is available.

3 METHODOLOGY

Within this research the final goal is to construct a classifier which is able to classify municipality documents correctly. In order to do this, various algorithms are examined. Also, the influence of the domain adaptivity and importance sampling are researched. This section introduces the datasets, experiments and further detail about the implementation of the algorithms.

3.1 Description of the data

Two datasets have been used within this research which both include documents from the government of the Netherlands. The first set, called PAR_ALL, consists of over 50,000 questions and answers that have been asked within the Dutch national parliament during 2001-2017 and these questions were scraped from www.zoek.officielebekendmakingen.nl. The content of these questions and answers ranges from critical examination of proposed laws to requests of more information about ongoing affairs currently within the news. The second set consist of 20,000 documents from Dutch municipalities, such as items on the agenda and notifications of commissions. This set is retrieved with an API of www.zoek.openraadsinformatie.nl and is called MUN_ALL.

Although both sets are political and Dutch, they do vary in content. One of the big differences is that the set of the national government only entails questions with answers, whereas the municipality dataset consist of a greater variety of document types. The themes discussed are also different because within municipalities only local policy is discussed such as the construction of local infrastructure and the re-allocation of local sport clubs. This is in contrast to the parliament, because in parliament broader themes such as criminal law and measure for social security are examined.

All of the parliament data is annoted with any number of labels denoting their content. These labels have two levels of detail; one broad category such as healthcare or law and one detailed category such as healthcare for the elderly or criminal law. For this research only the seventeen broad labels are used, the detailed 118 labels are dropped. The municipality set was not labeled, but for this research a part was manually annotated using the framework of the TaxonomieBeleidsagenda [**?**]. This taxonomy has also been used to annotate the parliament data. The labeled municipality dataset is named MUN_LABELED.

For one experiment the dataset with questions of parliaments has been split in two parts, based on when these questions have been asked. The first set consists of all questions from 2001-2009 (PAR_EARLY) and the second set composed of the questions of 2015-2017 (PAR_LATE). Figure 1 shows the relative occurences of labels within all datasets. Within Table 1 the datasets are summarized, as it shows some characteristics of these sets.

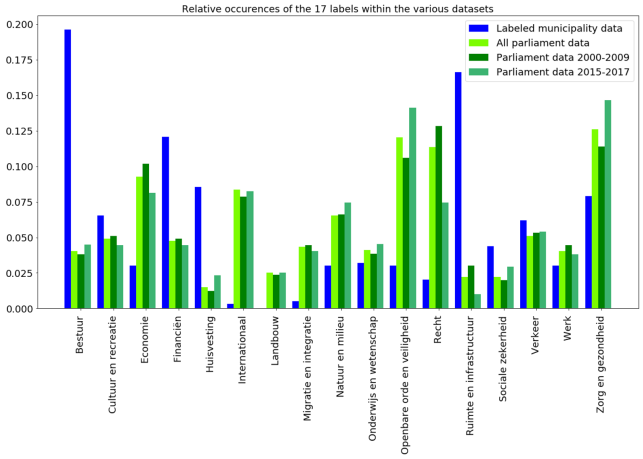

Figure 1: Relative occurences of labels within various datasets

3.2 Experiments

Within this research three experiments are carried out, which are all set-up via the following method. For all experiments a source and a target domain is defined. Around 80 percent of the source data is used for training and 20 percent for selecting the best hyperparameters. The final evaluation is done on the target set. For each experiment the evaluation is based on precision, recall, F1 and accuracy. The accuracy is defined as the fraction of samples that are entirely correctly classified which means all labels for one sample need to be correctly predicted. The equations of precision, recall and F1 are shown in equations 2, 3 and 4 [17].

Table 1: Summary of the datasets used within this research

Name Dataset	PAR_ALL	PAR_EARLY	PAR_LATE	MUN_ALL	MUN_LABELED
Amount samples	52397	28063	8202	20886	439
Average amount words	688	648	834	119	137
Standard deviation words	429	382	554	53	107
Median amount words	549	549	723	106	112
Average amount labels	1.62	1.85	1.32	-	1.36
Standard deviation labels	0.76	0.83	0.52	-	0.89

$$recall = \frac{truepositives}{truepositives + falsenegatives} \tag{2}$$

$$precision = \frac{truepositives}{truepositives + falsepositives} \tag{3}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{4}$$

True positives are samples which are correctly identified as being part of a label. False negatives are samples identified as being not part of a label, but in reality they are part of that label. False positives are samples that are wrongly classified as being part of a label. Notice that recall, precision and F1 are all binary classification evaluation metrics, which means merely one label at a time is evaluated. Therefore for the evaluation of the entire models the micro-average of these metrics is calculated, which is a method of aggregating the multiple scores into one score. The formula for micro-average is given in equation 5 [17].

$$microaverage(f(), D_T) = \frac{f(D_T^1) + f(D_T^2) + ... + f(D_T^{\|L\|})}{\|L\|} \tag{5}$$

f() can be any binary classification function, such as recall, precision or F1. D_T is the test data set on which the evaluation is done. D_T^n is the test set when specifically looking at label n, which thus turns the entire set into a binary classification set. L is the set of labels within D_S [17].

The first experiment is aimed at finding out how the classifiers are capable of classifying without a domain shift. This means that the PAR_ALL dataset is shuffled and part is used as source and part is used as the target domain. These results show the potential of the algorithms given the multilabel task.

For the second experiment the models are trained on the PAR_EARLY data as source domain and tested on the PAR_LATE data. This experiment is done twice per algorithm, as it is executed once with and once without importance sampling. For the importance sampling a logistic regression classifier is used which uses samples from PAR_EARLY and PAR_LATE. This classifier is then employed to weigh the remainder of the training data. The second experiment gives insight how well the classifiers adapt to a relatively small domain shift. Moreover, it shows how importance sampling influences the performance of the classifiers.

Thirdly, an experiment very similar to the second experiment is carried out. However, the data is different, because now the source domain is all PAR_ALL and the target domain is the parliament data from MUN_LABELED. Importance sampling is also used for this experiment, but now the classifier is trained on data from PAR_ALL and all data within MUN_ALL. Maximizing performance on this algorithm is the goal of this research, as this shows how the domain shift between the two dataset is mitigated. In addition to experiment 2 this also provides insight whether the degree of domain shift influences which classifiers perform best.

3.3 Algorithms

Within this research newer CNN-classifiers are compared to traditional classifiers such as Support Vector Machines (SVM), Logistic Regression (LR) and Random Forest (RF). For all these implementations the input is a bag-of-words representation of the various samples combined with TF-IDF. The implementation of Scikit-Learn is employed for these algorithms and the optimal hyper-parameters are chosen based on a grid-search. Moreover, when needed, the one-versus-all classifier is used to make the classifiers multi-class. To ensure multi-label outcomes multiple thresholds for prediction are experimented with.

Instead of bag-of-words CNN uses word embeddings as input. This research experiments with two Word2Vec embeddings, namely pre-trained embeddings retrieved trained on a variety of Dutch resources [16] and self created embeddings using Word2Vec-implementations from Gensim trained on PAR_ALL and MUN_ALL.

The established CNNs are implemented using the Keras library in Python. One of the employed CNN within this research is similar to earlier architectures [6]. This means that an embedding layer is used to transform sentences to a multi-dimensional space using one of the two Word2Vec-embeddings. Both embedding spaces have been tested with static and non-static initializations which indicates whether the embeddings can alter during training.

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 [19].

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 [11, 15]. 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 2

Table 2: 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 with one filter	3,5,7,11
Filter sizes with three filters	[2,3,4],[3,4,5],[7,8,9]
Threshold of prediction	0.3,0.4,0.5,0.6,0.7

An addition of this research is a slight variation to Kim's architecture, which tries to deal with the length variance of samples in a different way. Instead of padding and cutting to a pre-defined length this research splits up samples into blocks of 200 words each. The last block of each sample has zeros appended to, in order to make this block also a fixed length of 200. Also, for each block it is assumed that it has the similar labels as the entire samples. All these blocks of the training set are used for training the CNN model, which is similar to the previously explained CNN.
The optimized models are then tested on the test set. The test samples are also cut into blocks of 200, and it is marked to which sample it belongs. The final prediction per sample is established by classifying each individual block of 200 and then aggregating those individual predictions into one prediction per sample. This aggregation is done by averaging or maxing the prediction of every block belonging to a certain sample. This algorithm has the same parameters as Kim's model but it has additional parameters too which are listed in Table 3.

Table 3: Extra parameters and values within CNN-split

Parameter	Values
Block size	200
Aggregation	mean, max

4 EVALUATION

Within this section the results to the experiments are examined. This means it is evaluated which classifiers perform well on overcoming the domain shifts explained in section 3. Moreover, the effects of importance sampling are also analyzed.

4.1 Experiment 1: D_s = PAR_ALL, D_t = PAR_ALL

Within Table 4 the results for the first experiment is visualized, it shows the best performing algorithm of every type with its best parameters. The algorithms are optimized on F1-score, and the corresponding Precision, Recall and Accuracy for that initialization are also given. Table 4 shows that on the task of classifying parliament data SVM performs best with an F1-score of 0.784. However,logistic regression performs almost equally well with a F1-score of 0.779. The RF, CNN - Kim and split-CNN achieve lower results compared to the SVM. What can also be noticed is the relatively low accuracy of all algorithms in respect to the F1-score, this is because all seventeen labels need to be correct.

4.2 Experiment 2: D_s = PAR_EARLY, D_t = PAR_LATE

Experiment 2 has results similar to experiment 1 and these are listed in Tables 5 and 6. Once again the LG and SVM outperform the other algorithms with F1-scores of 0.623 and 0.614 without importance sampling. This also shows the influence of the domain shift for the classifiers, as both algorithms drop about 0.15 in F1-score. The RF performs better than the CNNs, which perform almost equally well within experiment 2.
The effects of importance sampling are not established within experiment 2, as most algorithms have almost equal performance with and without importance sampling. For CNN - Kim the importance sampling has most effect, although it does decrease performance.

4.3 Experiment 3: D_s = PAR_ALL, D_t = MUN_LABELED

Within Table 7 and 8 the results for the third experiment are shown. Table 7 shows the results of all algorithms without importance sampling, and this time the split-CNN has the best performance with an F1-score of 0.475. CNN-Kim does perform worse, but it shows that the semantic classifiers outperform the lexical classifiers on experiment 3. The F1 of the best classifier for experiment 3 is 0.3 lower than the best classifier for experiment 1. Importance sampling has a negative effect on the CNNs, but this time around it does positively influence the lexical methods.

The CNNs have shown quite a big discrepancy in which parameters are best for what experiment. Although this is quite surprising, some research has looked into parameter optimization for this kind of CNNs. They have also seen variations in what parameters perform best, so these results are therefore not problematic **??**.

5 DISCUSSION AND CONCLUSIONS

This research has provided information on the domain adaptivity of a number of algorithms and the effects of importance sampling. When experiment 1 is taken into account BoW-classifiers have outperformed the word embedding classifiers. This is a surprising result, as in much of the research on these word embedding classifiers better performances have been witnessed instead. Although surprising, on a few baseline multiclass classification tasks the word embedding classifiers were worse than traditional methods too [5, 6]. Overall this research shows that an F1-score of 0.78 can be achieved given this classification task when there is no domain shift present.
Experiment 2 shows how these various classifiers perform when a domain shift does occur. Importance weighing has little effect on the performance of the classifiers or even a negative effect for the CNN-Kim. Experiment 2 furthermore shows that lexical methods

Table 4: Results of algorithms without importance weighing trained on PAR_ALL and tested on PAR_ALL

Algorithm	Parameters	F1-score	Precision	Recall	Accuracy
LR	C = 10000 penalty = l2 solver = sag	0.779	0.781	0.777	0.509
SVM	alpha = 1e-5 penalty = l2 threshold = 0.4	0.784	0.783	0.785	0.519
RF	split criterion = entropy max_depth = 100 n_estimators = 30 threshold = 0.2	0.686	0.620	0.768	0.310
CNN - Kim	filter = [2,3,4] W2V = pre-initialized W2V init = dynamic Threshold = 0.4	0.727	0.745	0.711	0.444
CNN - split	filter = [3] W2V = pre-initialized W2V init = Dynamic Threshold = 0.4 Aggregation = mean	0.752	0.766	0.738	0.479
CNN - Kalchbrenner					
Par2Vec					

Table 5: Results of algorithms without importance weighing trained on PAR_ALL and tested on MUN_LABELED

Algorithm	Parameters	F1-score	Precision	Recall	Accuracy
LR	C = 10000 penalty = l2 solver = sag threshold = 0.3	0.623	0.576	0.677	0.345
SVM	alpha = 1e-5 penalty = l2 threshold = 0.3	0.614	0.540	0.711	0.31
RF	split criterion = entropy max_depth = 100 n_estimators = 30 threshold = 0.3	0.589	0.574	0.606	0.321
CNN - Kim	filter = [3] W2V = trained on corpus W2V init = static Threshold = 0.4	0.532	0.509	0.558	0.281
CNN - split	filter = [7,8,9] W2V = trained on corpus W2V init = Static Threshold = 0.4 Aggregation = mean	0.537	0.540	0.534	0.296
CNN - Kalchbrenner					
Par2Vec					

Table 6: Results of algorithms with importance weighing trained on PAR_ALL and tested on MUN_LABELED

Algorithm	Parameters	F1-score	Precision	Recall	Accuracy
LR	C = 10000 penalty = l2 solver = sag threshold = 0.2	0.624	0.58	0.675	0.348
SVM	alpha = 1e-5 penalty = l2 threshold = 0.4	0.630	0.569	0.705	0.34
RF	split criterion = entropy max_depth = 100 n_estimators = 30 threshold = 0.3	0.578	0.586	0.571	0.332
CNN - Kim	filter = [3] W2V = trained on corpus W2V init = dynamic Threshold = 0.4	0.505	0.511	0.498	0.280
CNN - split	filter = [7] W2V = trained on corpus W2V init = Static Threshold = 0.3 Aggregation = mean	0.537	0.540	0.534	0.296
CNN - Kalchbrenner					

Table 7: Results of algorithms without importance weighing trained on PAR_ALL and tested on MUN_LABELED

Algorithm	Parameters	F1-score	Precision	Recall	Accuracy
LR	C = 10000 penalty = l2 solver = sag Threshold = 0.2	0.310	0.371	0.267	0.146
SVM	alpha = 1e-5 penalty = l2 threshold = 0.3	0.317	0.263	0.401	0.075
RF	split criterion = entropy max_depth = 100 n_estimators = 30 threshold = 0.1	0.253	0.194	0.361	0.009
CNN - Kim	filter = [3] W2V = pre-initialized W2V init = dynamic Threshold = 0.4	0.415	0.447	0.388	0.267
CNN - split	filter = [2,3,4] W2V = trained on corpus W2V init = Dynamic Threshold = 0.4 Aggregation = mean	0.475	0.454	0.498	0.262
CNN - Kalchbrenner					
Par2Vec					

Table 8: Results of algorithms with importance weighing trained on PAR_ALL and tested on MUN_LABELED

Algorithm	Parameters	F1-score	Precision	Recall	Accuracy
LR	C = 10000 penalty = l2 solver = sag	0.338	0.414	0.285	0.166
SVM	alpha = 1e-5 penalty = l2 threshold = 0.4	0.368	0.340	0.401	0.130
RF	split criterion = entropy max_depth = 100 n_estimators = 30 threshold = 0.1	0.263	0.214	0.341	0.107
CNN - Kim	filter = [2,3,4] W2V = pre-initialized W2V init = dynamic Threshold = 0.4	0.387	0.494	0.319	0.289
CNN - split	filter = [2,3,4] W2V = pre-initialized W2V init = Static Threshold = 0.4 Aggregation = mean	0.454	0.421	0.493	0.246
CNN - Kalchbrenner					

are not outperformed by semantic methods on every domain adaptation problem.

The results on experiment 3 show how these various classifiers perform when a bigger domain shift does occur. Most notably, the BoW-classifiers, SVM and LG, are less adapted to deal with the domain shift than the various CNN-versions. This shows that feature transformation transfer approaches are suitable for overcoming a part of the domain shift when transferring to municipality data. Specifically the split-CNN shows good performance, as it outperforms the other word embedding classifiers by a large margin too with an F1-score of 0.48. The importance sampling does not have an universal effect, as it only increases performance for the BoW-classifiers.

These experiments have shown four interesting results which need to be further explained. Firstly, Random Forest and Par2Vec do not perform best on any experiment. Par2Vec most probably suffers from a lack of data, as it is trained on relatively little data compared to the research in which it was introduced [7]. The lack of performance of decision tree-based classifiers such as random forest has also been witnessed in other research before [1].

Secondly, the CNNs do outperform the BoW-classifiers on experiment 3, but not on experiment 2. This implies that when domain shifts are bigger more emphasis needs to be put on feature transformation transfers, as less of the feature space of the source domain exist within the target domain. As seen with experiment 2, when much of the feature space is still similar normal BoW-representation can still suffice. This effect is similar for the importance sampling too, which also increase performance for SVM and Logistic Regression more on experiment 3 than in experiment 2.

Thirdly, importance sampling has a negative effect on the CNN-classifiers whereas it does increase performance for the BoW-classifiers. This can largely be explained by weighing of the samples, which is done using a logistic regression which distinguishes between the source and target domain based on the BoW-representation. This means that for BoW-classifiers the weighing is based on relevant features, whereas these features are not for CNNs.

Fourthly, split-CNN outperformed the CNN-Kim on two out of three tasks. For the municipality data this is quite logical, as the split-CNN looks at smaller blocks and the municipality documents have an average length smaller than 200. Kim's implementation instead is trained to extract features in the back of longer documents too. Why the split-CNN performed better on experiment 1 is quite unclear.

In conclusion, the performances of the various algorithms on the parliamentary data do not generalize well on municipality data. This implies that the current dataset does not suffice for creating a decent classifier for municipality data. This is also the reason why the developed methodologies will not be implemented on www.openraadsinformatie.nl. The research has also shown that when the domain shift increases it is more beneficial to employ domain adaptation methods. Lastly, importance sampling has a negative effect on CNNs, but this can be explained by how the weighing is calculated. Importance weighing is an effective technique to increase performance of RF, SVM and LR in domain adaptation problems.

For future research on creating a good classifier for municipality data finding new data with a smaller domain shift is recommended. Given that relevant data can be found, an F1-score between the

results of experiment 1 and 2 should be achievable as the domain shift will likely be smaller than the gap of 6 years. Even if not enough data can be found to train entire classifiers still it can be used for the effective retraining of classifiers as demonstrated in earlier research [10].

An alternative for finding new data is further developing the current methodologies on this dataset. One of the ways further progress can be made is the use of ensembles which combines algorithms for a prediction of samples. Secondly, importance weighing specifically for CNNs can be established, so instead create classifier which distinguishes between the train and test data based word embeddings. Lastly, the split-CNN has been tested with only block sizes of 200, perhaps more parameter training can further improve performance.

6 ACKNOWLEDGEMENTS

I would like to thank everyone from Open State foundation for their support and friendliness during my internship. I have always gone to OSF with great pleasure partly due to the good discussions and atmosphere. Moreover, I have greatly appreciated the help of Maarten Marx and Tom Kunzler, as they constantly proposed new ideas and critically looked at mine. Lastly, thanks to my parents Diederik and Moniek, siblings Max and Iris, girlfriend Amber and friends Stefan, Roos, Joppe, Simon, Ethan and Josse for keeping me motivated and supporting me during the entirety of my studies.

REFERENCES

[1] Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*. Springer, 163–222.

[2] Wei Bi and James T Kwok. 2011. Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 17–24.

[3] Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* (2014).

[4] Thorsten Joachims. 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*. ACM, 128–136.

[5] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

[6] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[7] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[8] Yong H Li and Anil K Jain. 1998. Classification of text documents. *Comput. J.* 41, 8 (1998), 537–546.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[10] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? *arXiv preprint arXiv:1603.06111* (2016).

[11] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 437–452.

[12] Thien Huu Nguyen and Ralph Grishman. 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 International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Vol. 2. 365–371.

[13] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[14] Nederlandse Rijksoverheid. 2017. *TaxonomieBeleidsAgenda*. http://www.standaarden.overheid.nl/owms/4.0/doc/waardelijsten/overheid.taxonomiebeleidsagenda

[15] Tobias Sterbak. 2017. Guide To Multi-Class Multi-Label Classification With Neural Networks In Python. https://www.depends-on-the-definition.com/guide-to-multi-label-classification-with-neural-networks/. (2017). Accessed: 2018-05-23.

[16] Stephan Tulkens, Chris Emmery, and Walter Daelemans. 2016. Evaluating Unsupervised Dutch Word Embeddings as a Linguistic Resource. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* (23-28), Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association (ELRA), Paris, France.

[17] Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information retrieval* 1, 1-2 (1999), 69–90.

[18] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923* (2017).

[19] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).