Universiteit Leiden
The Netherlands

Opleiding Informatica

Classifying citizen complaints

using pre-trained language models

Nicolaas van der Plas

Supervisors: Suzan Verberne & Paul Veger

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

19/06/2020

Abstract

This thesis shows the effectiveness of state-of-the-art Natural Language Models on a customer complaint data-set owned by the company `Decos` which consists of municipalities, their self-chosen categories and the issues which their citizens submitted. The model used for this classification problem is `BERT` which was introduced by researchers of Google in late 2018. To achieve the highest model quality, parameters such as the learning rate, the learning policy, the maximum length of tokens and the amount of epochs are experimented with while fine-tuning a pre-trained Dutch model. The data-set poses a lot of challenges such as short and under-specified issues, imbalanced data, miss-labelled issues and per municipality unique categories. The model trained with the most optimal variables achieved an 86% F_1 score on the original data-set consisting of 145,000 records using cross-validation. Finally it is discussed what there could've been done better and what could be improved on in further research.

Acknowledgements

I want to thank Suzan Verberne for her help, ideas and time to guide me through this entire process. She was always able to point me into the right direction and stimulate me to go on the way I did. I also want to thank her for the final assessment for this thesis. Secondly, I want to thank Paul Veger for providing me with the tools and brainpower to achieve what I did. The inspiration and genuine enthusiasm he had positively influenced me to continue and try to perfect everything I did. Finally, I would like to thank Alain van Hanegem for providing me with data and guiding me through the technical aspect to use my research for a real business product.

Contents

1 Introduction

With the increasing popularity of Machine Learning (ML) and Artificial Intelligence (AI) comes the demand of the application hereof in municipalities. The motivation for this is the reduced effort it takes to get something done and remove manual errors made by humans. Essentially, the general consensus is that there is a need to move towards a 'smart-city' set-up. In this, the application of ML or AI in services municipalities use becomes more and more demanded and expected. An application where it is exceptionally useful and which a lot of municipalities use is acting on citizen complaints. There are a lot of providers for this and the leading most innovative provider for this service 'Fixi'. 'Fixi' was developed by a Noordwijk based Software company called 'Decos'.

Figure 1: Decos headquarters in Noordwijk. Source: www.decos.com.

Fixi is a platform on which citizens can send issues about their neighbourhood to their municipality so they can quickly 'fix' the problem. Fixi has an app and a desktop site where the citizens are able to send their issues. They can add a location, an explanation upon this location, the description and a photo to further explain what is wrong in their neighbourhood. These issues are stored in a database along with all the information the citizens provided.

In Fixi, classification is useful because it removes an extra step for the citizen to self-classify their complaint and it speeds up the internal work because a complaint can instantly get forwarded to the responsible party. For example: "The whole street is in the dark because the lights here are off." would be classified as 'Street lighting' and should be forwarded to the people who are responsible for fixing these lights in the specific region. Classification of these issues or complaints is hard because they can be underspecified, ill-defined or wrongly labelled, for example: 'Verkeersbord slecht leesbaar' is categorized by a citizen as 'Wegen, troittoirs & fietspaden' while it should have been classified as 'Verkeer, bebording & parkeren'.

Fine-tuning a BERT model on the Fixi data can be of great business-value to Fixi as more and more municipalities want to make it easier for citizens to submit issues in their neighbourhood and a quicker way it comes to the right organisations. The classification would mean the issue can be send to the correct party right away.

Next to the issues described above, anyone can imagine there are more issues about garbage or street lighting then there are about cemeteries which leads to class imbalance. Quality is also impacted by the learning rate, the learning policy, the batch size and the amount of epochs. An optimum set-up needs to be found to achieve the highest model quality. This results in my research question: *With what quality, expressed in the F_1 score, can an optimized fine-tuned BERT model classify very short and cryptic complaints?*

1.1 Thesis overview

T his chapter contains the introduction. Section 2 includes the background of the model, the method and the related work; Section 3 includes the data, which pre-processing, which models, the training methods and every experiment I've ran; Section 4 includes the outcome of the experiments I ran; 5 includes reflecting back on my experiments; 7 includes answering my research question and talking about how someone could improve on my research.

This bachelor thesis was created under the supervision of the LIACS institute, Suzan Verberne and Paul Veger from Decos.

2 Background

2.1 BERT

In late 2018 a paper was published about a new state-of-the-art Natural Language Processing paradigm. BERT, proposed by Jacob Devlin, Ming-wei Chang, Kenton Lee and Kristina Toutanova from Google AI Language [DCLT19], has had some significant impact on the NLP landscape. With pre-training on a large unlabelled data-set it was able to achieve amazing results on all of the 11 NLP tasks, with relatively little to no fine-tuning for each task. How is this possible?

BERT makes use of transfer learning. This means that BERT can gain knowledge on one problem and apply this knowledge on another problem. In practice this results into a pre-training step and a fine-tuning step. Pre-training is done on a large unlabeled corpus, like the entire wikipedia or a huge book corpus. BERT is trained via two unsupervised tasks, Masked Language Modelling(MLM) and Next Sentence Prediction(NSP).

BERT is a deep bi-directional language model which is in essence stronger than a left-to-right of right-to-left model. However, this proves a challenge as training bi-directional it would allow each word to see itself and result in wrong prediction of the target word. To solve this Masked Language Modelling was applied on the corpus. A certain percentage of all words will be masked and the model has to guess what words were supposed to be there. This method dates back all the way to 1953 when Taylor introduced the Cloze task [Tay53].

A lot of downstream tasks such as Question-Answering require to know the relation between sentences. These relationships are important to understand the context and relevance of preceding or proceeding lines. Naturally, a language model does not comprehend relations between sentences right away and requires it to be taught as well. BERT solves this by applying the NSP task which basically means providing two sentences and asking the model to predict whether or not this is the subsequent sentence. While this might seem very easy, this task proves to be very useful. The techniques are trained together, thus minimizing the combined loss function.

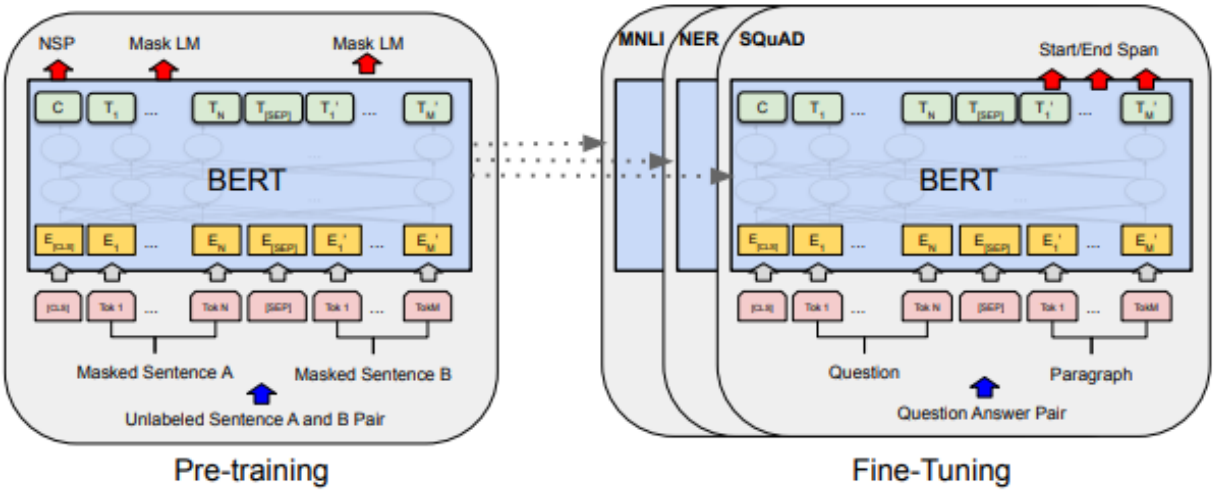


Figure 2: Example of the training process. Source:[DCLT19]

Everyone with enough resources can pre-train their own BERT model. A lot of models trained by individuals or researchers are uploaded to and contained in the Huggingface library and are free to download for everyone. On these models, fine-tuning afterwards is relatively inexpensive and quick. Because of transfer learning it is easy to fine-tune the model to a specific downstream task, in my case, classification of citizen complaints.

2.2 Huggingface Transformers

Fine-tuning a model naturally requires a pre-trained language model to start with. These models can be downloaded and loaded in by anyone via The Huggingface Transformers. Huggingface Transformers is a popular library for state-of-the-art NLP which provides pre-trained language models as well as a lot of examples, tutorials and scripts to pre-train your own model or to create a downstream task like Named Entity Recognition. The library can be used for a variety of tasks from QA to Text Classification. The project has a very large community which is very active, they help develop the library. According to the paper:"Its capabilities, performances and unified API make it easy for both practitioners and researchers to access various large-scale language models, build and experiment on top of them and use them in downstream task with state-of-the-art performance"[WDS+19]. Not only was it via this library easy to load in a model from the cloud, but it solved another problem within this research: a Dutch BERT model, called BERTje[dVvCB+19], was uploaded to their vast amount of pre-trained models.

2.3 Related work

This research problem is about a text complaint belonging to one of a number of categories or classes, which makes it a multi-class text classification problem. Multi-class text classification is a very common operation and there are a lot of examples on the internet. One very similar example is on the the Machine learning and Data Science platform 'Kaggle', where multi-class text classification is done on consumer finance complaints. The key difference with my research, next to different data obviously, is the use of different methods. While classification in this research is done with the use of a state-of-the-art BERT language model, in the example several traditional models are tested such as 'LinearSVC'. The classification quality will of course differ between these models so while data preparation and the result format will almost be the same, in essence the classification is done differently.[1].

The developer of the data-set also provided a notebook with his results; In his case, the author had 1.42 million records spanning thirteen categories. These are his results on several models, as well as a more in-dept result on one of them.

Another example uses a Hybrid-attention GRU neural network [WWWT19]. In this paper the authors propose the hybrid-attention GRU neural network(HATT-GRU) for complaint classification. Their experiments are conducted on different complaint databases from different industries.

[1] https://www.kaggle.com/selener/multi-class-text-classification-tfidf

model_name	Mean Accuracy	Standard deviation
LinearSVC	0.779103	0.005528
LogisticRegression	0.758114	0.009978
MultinomialNB	0.647500	0.004733
RandomForestClassifier	0.387005	0.010992

Table 1: Results of the Kaggle notebook on 4 different models

	Precision	Recall	F_1 score	Support
micro avg	0.78	0.78	0.78	2500
macro avg	0.62	0.51	0.54	2500
weighted avg	0.77	0.78	0.77	2500

Table 2: In-dept results of the LinearSVC model

The model is trained and evaluated on the multiple complaint data-sets and their quality is measured with the F_1 score. In their research they compare many models and three different data-sets, but for the sake of simplicity I will only display the results of their proposed model and the Decision tree on the credit reporting complaint data-set.

	Precision	Recall	F_1 score
Decision tree	0.765	0.795	0.779
HATT-GRU	0.890	0.926	0.907

Table 3: Results of the two models on a credit reporting data-set

Another example closer to home is the software made by a former employee of Decos who already tried to solve this classification problem a few years ago. A few notes to make is that because it was done a while ago, fewer municipalities were connected, fewer categories were available and also fewer complaints were issued. His approach was a bag-of-words model with TF-IDF weighting and a MultinomialNB. In his solution he used TF-IDF to determine the important words in a specific class and stored those. He used those words and run them through the MultinomialNB model to classify the category. With the disadvantage of fewer data and sparse word features, he had an advantage in the fewer classes and municipalities. More classes result naturally in more fine-grained classes making it harder to predict the right class. He was able to get an F_1 score of 56-83% depending on the region with an overall average of around 70%. However, a lot of regions couldn't be classified using this model because he had set a threshold of 200 issues. The results of his research are in Appendix A.

A limitation both of the approaches have is the use of an older non-state-of-the-art model with sparse features which has limitations concerning context. BERT uses dense semantic feature representations and does not rely on specific words to be present in the issue. With the use of this language model predictions are made by more than just the occurrence of keywords.

3 Methods

To eventually get a fine-tuned model with a decent quality out of a pre-trained language model, a lot of background research is needed; a language model needs to be found that suits the needs of the problem and software to be written to work with this model or to use a wrapper readily made available by someone else. When first analyzing the problem, it was quite clear this was a Multi-class text classification problem with fine-grained classes and most likely very short texts. Looking up similar work gave me the one found on Kaggle mentioned in section 2. This gave some insight in how to use Sci-kit learn to prepare my train and test set, how to extract the useful columns of the complete data-set and also how to view wrongly categorized results as shown in section 4.

3.1 Data

The original data-set contained ∼145,000 records with the first complaint issued in late 2017. It contained data from 19 organizations (table 6, which are basically groups of nearby municipalities, with a total of 64 municipalities/regions(table 5) and a total amount of 177 unique categories. The reason why there are so many categories is that the municipalities, next to the few main categories, are able to make their own categories. One reason this is the case is that, for example, municipalities near the coast have to deal with Seagulls while municipalities more in-land have to deal with the Processionary Caterpillar. An example of an organisation is the 'buch-gemeenten' which includes Bergen, Castricum, Uitgeest and Heiloo. Other than the category, region and organisation, the data-set contained a lot of other labels but with those being abundant, the only focus was on the issue description. The issue description can be of variable length from 0 to 999 characters.

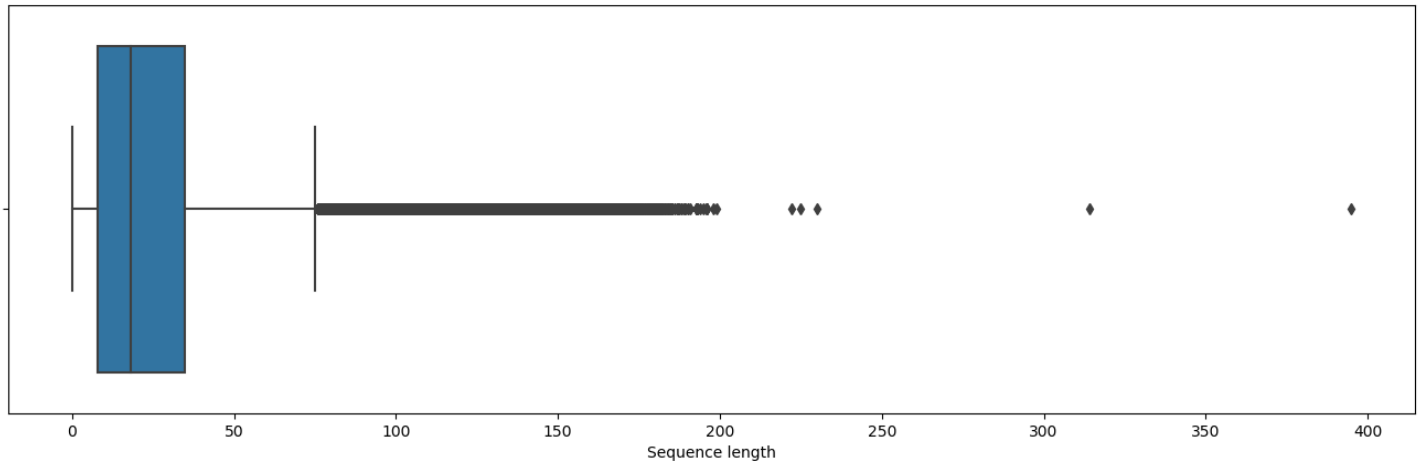


Figure 3: Boxplot of the sequence length of the issues. The mean sequence length of a description is 27 characters while the 95$^{\text{th}}$ and 99$^{\text{th}}$ percentile are respectively 88 and 154.

With this many categories and only a few categories shared between all municipalities, there is a large class imbalance because it makes sense there would be more complaints regarding trash or street lighting then there are complaints about cemeteries. For example on the category 'Afval' are 18035 complaints while on the category 'Winkel deurbeleid' only 2 as shown in table 4.

Category	**issues**
(Verkeers)borden, hekjes, palen, banken	37
(Verkeers)borden, hekjes, palen, banken	1380
Afval	18035
Afval	280
Afval (dumpingen en zwerfafval)	164
...	...
Wegen, straten, trottoirs en bermen	324
Winkel deurbeleid	2
Zwerfafval	813
Zwerfafval / dumpingen	2
Zwerfvuil/afvalbakken in de openbare ruimte	606

Table 4: Categories with their ammount of issues

Region	**issues**
Alblasserdam	1114
Arnhem	643
Bergen-nh	545
Best	5546
Binnenmaas	5197
...	...
Wijk7-defryskemarren	59
Wijk8-defryskemarren	46
Zuidoost-Arnhem	2076
Zuidwest-Arnhem	1690
Zwijndrecht	3331

Table 5: Regions with their amount of issues

Organisation	**issues**
DuivenWestervoort	9988
Arnhem	10639
Beekdaelen	7339
Best	5546
Brummen	5045
Buch-gemeenten	2427
Bunschoten	276
Defryskemarren	1041
Doesburg	1490
Drechtsteden	23671
Echtsusteren	3124
GemeenteEijsdenmargraten	1634
Goes	5032
Haarennb	2
HLT	24603
Hoekschewaard	13329
Kapelle	805
Katwijk	7690

Table 6: Organisations with their amount of issues

Category	issues
Begraafplaatsen	48
Dieren	3639
Handhaving & overlast	2074
Milieu	661
Openbaar groen	21736
Openbare verlichting	17631
Overig	6927
Schade	478
Speelplaatsen	1654
Vandalisme	124
Verkeer, bebording & parkeren	15044
Vuil	39433
Vuurwerk	241
Water, riolering & bruggen	11248
Wegen, troittoirs & fietspaden	23937

Table 7: Superset support before balancing

Category	issues
Begraafplaatsen	7331
Dieren	8662
Handhaving & overlast	7420
Milieu	7060
Openbaar groen	10867
Openbare verlichting	8808
Overig	5079
Schade	5135
Speelplaatsen	6327
Vandalisme	5348
Verkeer, bebording & parkeren	6366
Vuil	19715
Vuurwerk	8381
Water, riolering & bruggen	6604
Wegen, troittoirs & fietspaden	11968

Table 8: Superset support after balancing

3.1.1 Superset

Next to the original data-set, an extra data-set was created which consisted of only 15 categories which incorporated all the 177 categories and was not limited to one organization or region.

This data-set is useful for tweaking hyper-parameters like the learning rate and policy because the issues are all combined in here making it an excellent canvas. The data-set consists of the aggregated data of all regions in the original data-set. It also deemed useful for creating a training, test and validation set for comparing models, split in respectively 80%, 10% and 10%.

3.1.2 Imbalance

As shown in table 7, there is a large class imbalance present in the data-set which is bad for the overall quality of the model as little supported classes are out-weighted by the large ones. To combat there are two ways to balance the data. One possibility is over-sampling using SMOTE or just copying records from under-balanced classes and using them again. One reason to over-sample with the latter instead of the former and why oversampling is necessary in this case, is discussed in section 4. To random over-sample a function was written(3.1.2) which would find the largest supported class and for every class would check whether the support for that class was lower than 30% of the largest one, if it was it would be appended by random samples of its original records otherwise it would randomly select 70% of the class.

```
def balance_data(data):
    max_size = data['category'].value_counts().max()
    lst = []
    for _, group in data.groupby('category'):
        if len(group) < 0.3 * max_size:
            lst.append(group.sample(
                int(random.randint(int(max_size / 2), max_size) / 4), replace=True))
        else:
            lst.append(group.sample(frac=0.7))
    frame_new = pd.concat(lst)
    return frame_new.sample(frac=1)
```

Another addition to further balance the data is to add class weights. This would give more weight to less supported classes to increase their impact when training instead of being neglected by the model.

3.2 Pre-processing

Before fine-tuning a model on data, pre-processing is required. A data-set contains a lot of columns which have to be filtered and special characters or empty string which have to be replaced or deleted. The classes might also be non-numerical and have to be converted to the numerical labels. These are my actions:

- Filter out the necessary columns, like the issue-description and the category as well as the organisation for creating a model per organisation.
- Cleaning up the data by getting rid of every special character as well as weird no-sense-making descriptions, which is done via Regex matching. For example an empty description or a description in which the citizen only put his phone number for contact won't help anything for predicting the category and are thus excluded from the data-set.
- Convert the categories to numerical labels by encoding them using Sklearn's pre-processing library[2], with the LabelEncoder which is implemented in this library the classes are first fit and then transformed into numeric labels.

[2] https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

- Creating a super-set because this way the classes would be less fine-grained and a very accurate model could be trained. Because this is not region specific, it posed a problem to the municipalities because in this set-up they can't have their own categories which they really wanted to have, however it is a very good citizen complaint baseline model.

Having prepared the data, every issue has to be further pre-processed for BERT. Pre-processing for BERT is done in a few steps; first of all the input is tokenized using the model's tokenizer, in other words converting the input text to Wordpiece embeddings and then converting them to token ids. After the input is tokenized, the input is padded to the maximum length the user gave to the model with a padding token, which is default '0'. Instead of coding all that myself I decided to use a Huggingface wrapper made available online called Ktrain [Mai20]. Ktrain is a Python library which is essentially a wrapper for the Tensorflow library.

3.3 Model training

A BERT model has a lot of parameters to optimize its performance. A few of these are fixed in my experiments while others are experimented with to figure out what the impact of altering these parameters is to the final quality of the model. A few of these fixed parameters are:

- A maximum sequence length of 200, because a higher maximum length will most likely result in running out of memory and because the 99th percentile of length of issues is <200.
- A pre-trained Dutch language model with a L-12_H-768_A-12 configuration.

To determine further parameters such as the learning policy and the rate, Ktrain again proved useful with built-in features such as:

- Computing weight classes for class imbalance
- Learning rate finder to estimate optimal learning rate
- Different learning rate policies to minimize loss
- Easy loading and saving of trained models
- Perform multi-lingual text classification

With the help of the multiple learning schedules and the learning rate plotter the learning rate and the best learning schedule can be determined to further increase the quality of my model.

3.4 Baselines

Multi-class text classification is often done using a word vectorizer with TF-IDF weighting and a classification model like LinearSVC. In this Kaggle notebook[3] a user analyses multiple multi-classification models on a Finance consumer complaint database. The models the author compares are:

- **Random forest classifier**
  The Random forest is a classification model consisting of many decision trees which all return a prediction, the prediction with the most hits will be the end prediction. The classifier is incorporated in the Sklearn library and has a lot of parameters however, I used the same parameters as the author of the Kaggle notebook.
- **LinearSVC**
  The Linear support vector is a faster implementation of the regular SVC. According to Wikipedia[4], a SVM model is "a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible." New examples are then mapped into this space and the prediction will be based on the point where they are placed by the algorithm.
- **MultinomialNB**
  The multinomial Naive Bayes is used for classification with discrete features like word counts in text classification. It is an instance of the normal Naive Bayes classifier but it uses a multinomial function. The model uses the frequencies of the features to judge to which of the classes the input text belongs.
- **Logistic Regression**
  Logistic regression is used to assign observations to a discrete set of classes and is often used for classification problems. It is one of the basic and most well-known algorithms to solve classification problems. It's underlying technique is near as the same as Linear Regression.

While the author classifies on different data, the concept is the same. There is a text issue and a category to which it belongs. His analysis of the LinearSVM, which is none-state-of-the-art, next to his other models will be useful as a baseline.

Another comparison I can instantly make because it is also used as one of the models inside the Kaggle notebook, is the one my former colleague created using a TF-IDF vectorizer and the MultinomialNB model.

3.5 validation

After training and testing on a data-set, validation is the next step. The reason for creating another set is that this is 'new' data which the trained model has mostly not seen before. This means that the scores that are generated by this set determine the actual quality of the model. All the models, also the baseline follow the same steps:

[3] https://www.kaggle.com/selener/multi-class-text-classification-tfidf
[4] https://en.wikipedia.org/wiki/Support_vector_machine

- The data-set is split in to a training, a development and a test set of respectively 80%, 10% and 10%.
- For calculating the quality of a model I create a classification report using Sklearn's metrics library. The score I am comparing between models is the F_1 score.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

For comparing the BERT model against the other models, the same data-sets are used for the train, test and validation set and the quality is determined by the F_1 score.

3.6 Named Entity Recognition

BERT can be used for another downstream task called Named Entity Recognition(NER). When masking the entities like names or locations, an assumption is that the model would give more attention to the rest of the text which could improve the quality of the classification model. So during fine-tuning, another downstream task of BERT was trained on the SoNaR-500 corpus[OHJ+14] as well as the CoNLL-2003 corpus[SDM03] using the run_ner.py script which is a script inside the Huggingface GitHub library. With the fine-tuned model, every issue in the data-set was masked after which the normal procedure of classifying followed.

Due to the recognition of entities such as personal names, locations and organisations, these entities can be masked to anonymize text. This trained model thus added another business value as a bonus to not only Fixi but many other services Decos provided and will be as a result deployed the same way the classificatoin model will be.

3.7 Deploying

Deploying a classification or NER predictor can be done in several ways but for these models the best way to deploy it is as a REST API in this case using Flask and a docker-compose Python 3.7 container in a Linux environment. A Python file had to be created containing the routes the API is able to be called at. For example a POST request to /api/classify with an issue sent in the text body will return the category classified by the model.

4 Results

While finding the optimal learning rate, the best learning policy, the amount of epochs and whether or not to balance classes with extra oversampling a useful data-set was the super-set data. The reason for this is that in this data set the data of all regions have been aggregated resulting in a large training and test set giving extra validation if manipulating these hyper-parameters has a big impact on fine-tuning. In the BERT paper is mentioned that "For fine-tuning, most model hyper-parameters are the same as in pre-training, with the exception of the batch size, learning rate, and number of training epochs." So a good starting point was deciding the optimal learning rate for the data using the built-in learning rate finder from Ktrain on 2 epochs and a batch size of 16.

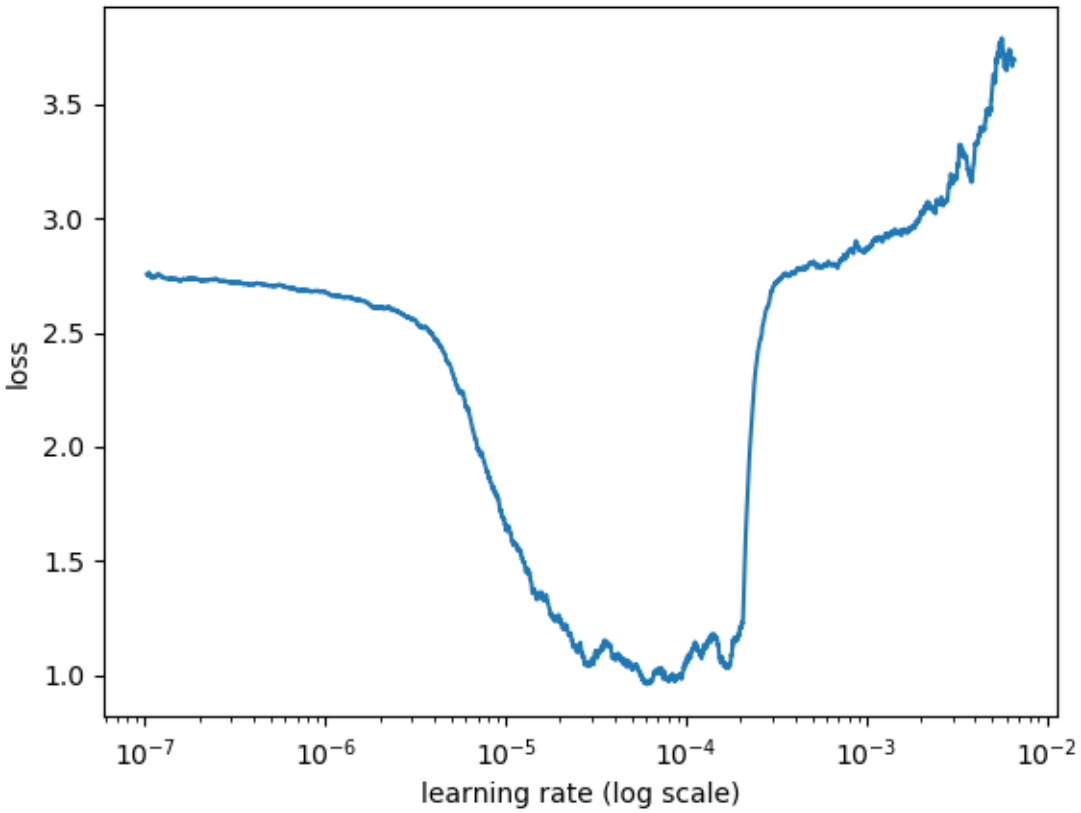


Figure 4: The plot for the learning rate on my data

4.1 Learning rate and policy

The optimal learning rate for which the loss is lowest. The loss is calculated on both the training and validation data-set and its value is based how well the model predicts both of these sets and is actually the sum of errors made in one of these sets. The value implies how poorly or well a model behaves after each epoch. A lower loss value means in this case a higher quality. Graph 4 has a few

troughs where the loss is at one of its lowest points for this data-set, one of these troughs is at 10^{-4}. To make sure this is the best performing learning rate for my data I evaluated four values around this value on the validation set using three different learning rate policies. With the use of the SGDR (Stochastic Gradient Descent with Restart), the onecycle and the triangular learning rate policies a comparison was made on the respective classification reports per policy and primarily the F_1 score on the impact of different learning rates on the exact same validation set.

	10^{-5}	10^{-4}	20^{-4}	10^{-3}
Onecycle	0.708	0.856	0.824	0.161
SGDR	0.710	0.831	0.155	0.048
Triangular	0.706	0.847	0.163	0.162

Table 9: The F_1 score score on the policies with different learning rates

As table 9 shows, a learning rate of 10^{-4} has the highest F_1 score as the graph in figure 4 already predicted. With this information the learning rate of 10^{-4} was selected as the default learning rate for the best model quality on this data-set. Another point learnt from this experiment is that not only 10^{-4} was the optimal learning rate, but also that the Onecycle policy was the best learning policy for this data-set. While the Triangular policy was a very close competitor the decision was made to continue with the Onecycle policy due to quicker training times, the Stochastic Gradient Descent with Restart and the Triangular policy however fell of with a increased learning rate while the Onecycle policy was able to keep up for a while with the learning rate of 20^{-4} this could be due to the fact that the Onecycle policy runs a single triangular cycle over the course of training and then annihilates the learning rate to a near-zero value towards the end making it more capable of performing on learning rates closer to 0.

4.2 Epochs

Having decided the best learning rate and policy, another parameter to further improve quality are the different amount of epochs. An epoch indicates the number of passes through the entire training data-set and can be as large as one wants. However, an epoch on this large of a data-set can take a very long time and also can result in over-fitting. Over-fitting means that whenever you train on specific data too much, the model will depend on this data and fail to fit additional data. For example, you can achieve 99% F_1 score on your train data by doing a lot of epochs over this data-set but when, after that, you test your model against a new test set you only achieve 60%. This is the result of your model becoming to reliant on your original data and thus not generalizing well to unseen data.

Epochs	1	2	3	4
F_1 score	0.828	0.856	0.874	0.883

Table 10: The epochs with their F_1 score after validating

As table 10 shows, more epochs improves the model quality but the gains are very minor. In this case, the quality from one to two epochs improves quite substantially but after that it improves per epoch by only one percent. It would be ideal to train with four or more epochs for the highest

quality achievable, but an epoch takes 4500 seconds and it is thus not viable to do so. Therefore, I will be training from here forward, with two epochs taking in mind the model quality and the time it takes.

4.3 Batch size

Another parameter which greatly influences quality is the batch size. It defines the number of samples that will be run through the model per iteration. For example, with a total training-set of 12000 and a batch size of 64, there would be 188 iterations where 187 iterations would have 64 samples and the last one 32. Having a larger batch size reduces the amount of iterations and thus time. The batch size is usually a multiple of 2 and while the intention was to test many different batch sizes, the machine the experiments were ran on limited the batch size to a maximum of 16 because the graphics card ran out of memory. The batch size of 2 was also excluded because of the very poor performance of the batch size of 4 and the long time it would have taken, the results in table 11 are concluded from validation on the validation set.

Batch size	4	8	16
F_1 score	0.161	0.856	0.867

Table 11: The batch sizes with their resulting F_1 scores on the validation set

4.4 Comparison

With all the hyper-parameters tuned to their best performance, the model could be compared against the one of my former colleague had developed as well as the baseline which was set. The countvectorizer was set to a maximum of 20,000 features and a min_df of 10, this was to avoid running out of memory. From the best performing model a confusion matrix was made and compared with BERT.

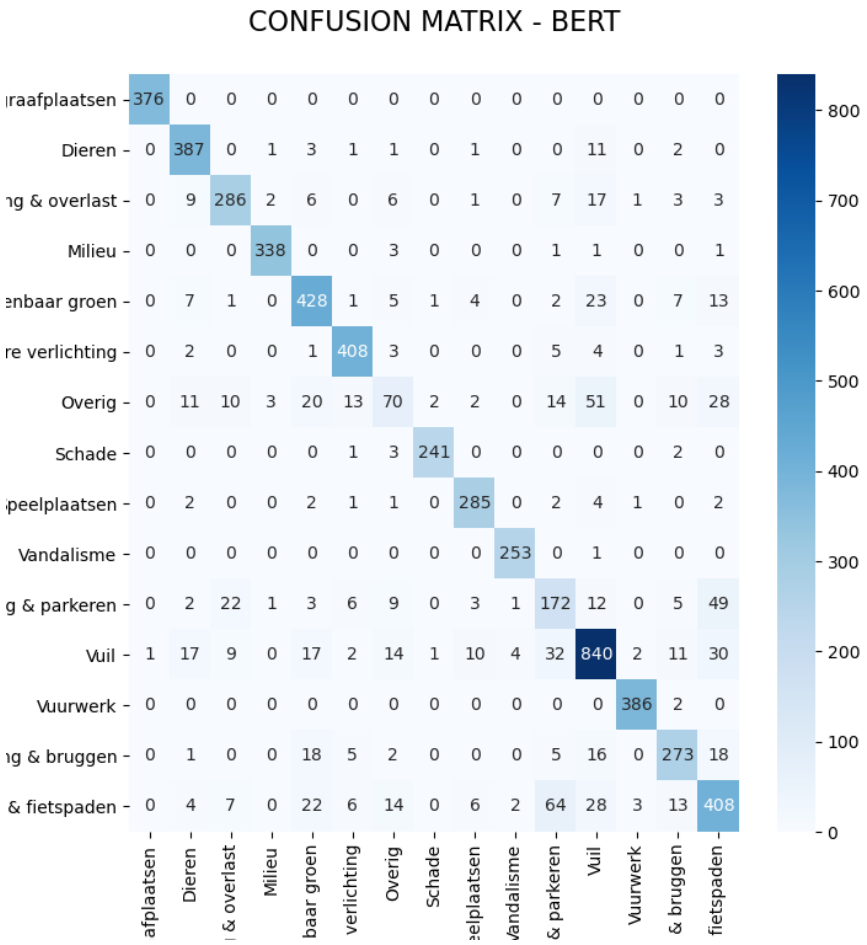
Model name	**Precision**	**Recall**	**F_1 score**
LinearSVC	0.831	0.838	0.833
LogisticRegression	0.797	0.808	0.797
MultinomialNB - Former colleague	0.786	0.775	0.759
RandomForestClassifier	0.616	0.296	0.244
fine-tuned BERT	0.858	0.863	0.860
fine-tuned BERT with masked named entities	0.846	0.857	0.851

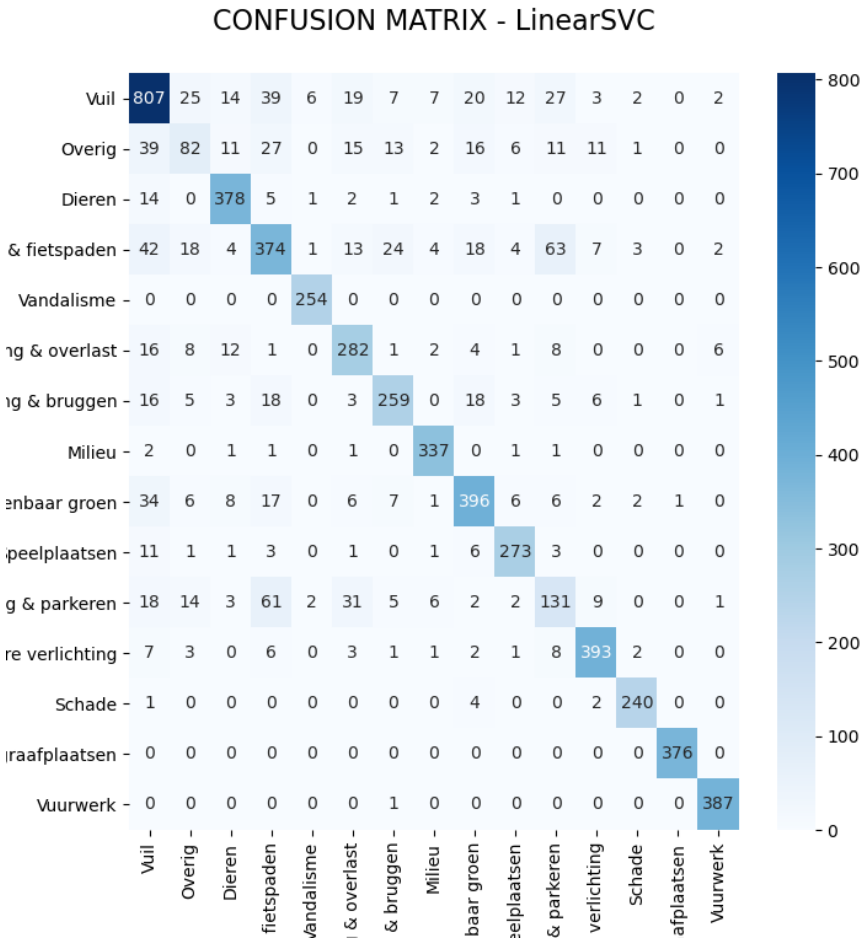
Table 12: The weighted averages of different measurements of the models

First thing to notice when looking at this graph is the very poor performance of the Random Forest Classifier. This is very odd because usually it is a very well-performing model. I could not figure out why it performed the way it did and even after experimenting some more with the parameters I could not get a different result, comparing with the results of the Kaggle author he also performed poorly with the RandomForestClassifier. It is also worth mentioning that all the models were trained on a maximum of 20,000 features, this sure has a small impact on the quality of the model but this is nullified as all the models have this limitation and thus it is still a valid comparison.

Another noticeable point is the performance of the masked BERT model against the non-masked one. It seems that BERT does not give much or any weight to personal names, organisations or locations when classifying issues.

Below are two confusion matrices of the BERT and the Linear SVC model evaluated on the exact same 6,000 issues. As is shown in the matrices, the 'Overig' class is widely spread because of the ambiguity of the issues in this class. Looking more closely shows that the confusion matrix of the BERT model is less spread and has more accurate predictions. Overall, comparing the matrices shows that the fine-tuned BERT model can more accurately predict the correct category.





4.5 Misclassifications

Obviously, when training on user-generated data, there will be some misclassifications. This can be because of the model wrongly predicting an issue and it being actually wrong because of a very non-informative issue, or because a model predicts the right category however the issue is categorized wrongly. A large reason for this is that many complaints are in a grey area and can for example be in both the 'Openbaar Groen' as in the 'Wegen, troittoirs & fietspaden' category. In a large data-set as the one this research is done on, there are a lot of these misclassifications. A few examples are:

Issue	**Predicted as**	**Actual category**
Foto spreekt voor zich	Vuil	Verkeer, bebording & parkeren
Zucht	Verkeer, bebording & parkeren	Wegen, troittoirs & fietspaden
Paal omver gereden	Wegen, troittoirs & fietspaden	Vuil
Wederom zand en modder op de weg	Wegen, troittoirs & fietspaden	Verkeer, bebording & parkeren
Verkeersbord slecht leesbaar	Verkeer, bebording & parkeren	Wegen, troittoirs & fietspaden

Table 13: Issues with their BERT predictions and their actual category

The first two are issues clearly non-informative and the latter are wrongly categorized by the citizen.

In this data-set, there are a lot of these examples and because of the over-sampling through copying issues there are even more present. This means that the quality of the data-set is not 100%, but on an estimation based on 100 random examples from the set around 90%.

Category	Database	Human	Predicted
Begraafplaatsen	-	-	-
Dieren	2	3	3
Handhaving & overlast	2	6	5
Milieu	1	-	-
Openbaar groen	11	10	12
Openbare verlichting	10	8	10
Overig	5	4	3
Schade	-	2	1
Speelplaatsen	-	-	-
Vandalisme	-	1	-
Verkeer, bebording & parkeren	13	14	12
Vuil	27	26	27
Vuurwerk	-	-	-
Water, riolering & bruggen	13	11	10
Wegen, troittoirs & fietspaden	16	15	17

Table 14: Value count per category with Human and BERT prediction

As shown in table 14, in this random sample of 100 issues comparing the categories in the database against the human classification, a few of the issues are wrongly categorized. As a result, classifying with BERT will prove more of a challenge because BERT will make a connection between the issue and the wrong category. This will improve misclassifications of the model itself. Another factor which reinforces this is random over-sampling, this method can actually increase the amount of wrongly categorized issues because they might be over-sampled.
Luckily, the issues which are rightly classified largely outweigh these issues so it will mostly likely only hurt the quality of the model because of these false negatives.

5 Discussion

While the quality of the fine-tuned BERT model is already very decent, there are still some improvements to be made. A lot of issues in the database just contain gibberish or even consists of some spaces or tabs and are thus not technically 'empty' and as a result not filtered out. These leave the model to just guess the class and that will in most cases result in a large loss. There are a lot of these records in the database and they have to be filtered more carefully, to then train a new model which does not contain these issues.

Another reason the quality is not as high as it could be is due to citizens wrongly classifying issues as seen in table 14. While their issue might be about tree branches which fell on the road, they might classify them as 'Openbaar Groen' while they actually belong to 'Wegen, troittoirs & fietspaden' in the case of the super-set categories. This example is in a grey area as the issue can belong in both these categories however there are more clear examples out there.

As shown in this paper by PhD student at the Khmelnytskyi National University in Ukraine[Rad17] a larger batch size is tied to a higher F_1 score and thus a better model quality and speeds up training times, however this requires more RAM and also more computational power. So to achieve even better results, I would have to do my training again on a different machine with more RAM and a more powerful GPU.

Another possible benefit is to have more RAM in the machine. With more RAM we could improve the comparison experiment by training, testing and evaluating on more features improving quality of the models. This would give an even better view on the results of the different methods.

I also could have done my experiments better, this because I balanced the whole data-set instead of just the training set. This resulted in reoccurring values of the training set in the test and validation set which may have influenced the quality of the fine-tuned model.

6 Realising business value

The business value of my results is divided into two parts. On the one end there is the fine-tuned classification model for Fixi and on the other there is the Named Entity Recognition. The classification model will make it easier for citizens to submit their issues and for the municipalities to process these issues. The apparent goal is to more quickly solve the issues and to create a higher level of contentness in the municipality, with the predicted category and a certain threshold set on the probability, say 90%, the municipality could instantly send the issue forward to the organisation which deals with these issues instead of the time it takes to review it and manually forward it. Another option is to show the top three most likely categories to the user making it easier and quicker to select the right category. This feature would increase the value of Fixi for the current municipalities using Fixi and would make it more appealing for the ones which are not.

Another task of BERT which can be of great value not only for Fixi but for Decos as a whole is Named Entity Recognition. Privacy is a very important value and is much demanded by customers and their users. This applies to all businesses and especially for one that works closely with government instances. With the help of the NER model, entities in a string of text can be recognised and then masked by replacing them with another string for example 'XXX'. Again, this would

greatly increase the value of services Decos supplies to existing customers and make it more appealing for those who are not.

6.1 Suggestions for deployment

The municipalities that use Fixi are able to choose their own categories. Because of this, many different categories are used for different purposes. A way to limit the municipalities doing this is to internally take the structure of the super categories and make the municipalities or tenants create one mapping value for each of these categories. The reason I am bringing this up is that some municipalities take a different structure for example; some have categories as 'Verkeer & wegen' or 'Openbaar groen en verlichting' which when mapped to one of the super categories always rules one of the two out. This way every issue is always linked to one of the super categories, with zero to no ambiguity, and every municipality has at most one value mapped to one of these categories while at the moment, with my own created mapping scheme, some municipalities have more. With this environment in place, the quality of the model will eventually improve as more and more issues will come in and be correctly mapped.

Another feature that will improve model quality is the introduction of a continuous deployment and training mechanic. This means that the model is always to be trained when a specific threshold, say 1000 new issues, is reached and automatically substituted with the old model. This will further increase quality as the new issues are automatically classified. The development of this would require a pipeline, with a check on the amount of new issues, which would set up the environment and libraries for the training, then collect the top 140,000 issues, fine-tune the model on this data-set and then substituting the old model with the newly fine-tuned one. The reason of the 140,000 issues and not just all of them is that, with this method, in 140 deployments, all old issues which are classified by citizens are not trained with anymore resulting in less losses and thus a higher quality model.

The last improvement to be made is to create an image classification model of the issues. A lot of issues also contain, next to the description, an image, for example of a fallen tree or a flooded sewer. A model could be trained on these images and applied in combination with the BERT model to more accurately predict the right category.

7 Conclusions

This research aimed to identify what the quality is of an optimized fine-tuned BERT model for classifying citizen reports. Based on the experiments regarding the optimal settings for the hyper-parameters, the pre-processing of the data and the comparison between BERT and other language models, it can be concluded that a pre-trained BERT model can be fine-tuned with an F_1 score of at least 85% and that it out-performs non state-of-the-art models even with noisy and/or short issues.

The fine-tuned model will be of great value for the Fixi team when implemented in their service and the side product, the NER model, which at first was just another test method will prove even more useful for the whole of Decos when implemented. These models will create whole new value for Fixi and other services Decos provides.

References

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[dVvCB+19] Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, 2019.

[Mai20] Arun S. Maiya. ktrain: A low-code library for augmented machine learning. *arXiv*, arXiv:2004.10703 [cs.LG], 2020.

[OHJ+14] NHJ Oostdijk, Veronique Hoste, F de Jong, Martin Reynaert, O De Clercq, H Heuvel, and B Desmet. Sonar-500. 2014.

[Rad17] Pavlo M Radiuk. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, 20(1):20–24, 2017.

[SDM03] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

[Tay53] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[WDS+19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[WWWT19] Shuyang Wang, Bin Wu, Bai Wang, and Xuesong Tong. Complaint classification using hybrid-attention gru neural network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 251–262. Springer, 2019.

A Results of colleague

Regio	Aantal meldingen	Aantal categorieën	Correct voorspeld
lisse	4240	15	72%
teylingen-sassenheim-warmond	2474	5	81%
teylingen	1702	8	83%
hillegom	4241	16	73%
teylingen-voorhout	2474	5	81%
cgm	2869	17	57%
zevenaar	897	6	69%
montferland	0	0	%
medemblik-medemblik	38	6	%
wervershoof-medemblik	62	5	%
wognum-medemblik	62	5	%
werf-medemblik-medemblik	55	5	%
duiven	3192	9	68%
brummen	902	15	58%
pauwhof	2	2	%
oegstgeest	1945	9	70%
Stadskanaal-Waterland	1945	13	69%
Maarsveld	1945	13	69%
Maarsstee	1945	13	68%
Ceresdorp	1945	13	69%
De-Borgen	1945	13	67%
Maarswold	1945	13	67%
Dideldom-Vleddermond	1945	13	69%
Noord-Oost	1945	13	67%
Stadskanaal-Centrum	1945	13	68%
cereshof	1945	13	70%
stadskanaal	1235	8	92%
De-Hagen-Vogelwijk	1945	13	68%
Musselkanaal	1945	13	68%
Parkwijk	1945	13	67%
weesp	1477	15	56%
sv	8577	10	76%
wijk4-defryskemarren	113	5	%
wijk2-defryskemarren	113	5	%
wijk3-defryskemarren	113	5	%
wijk1-defryskemarren	113	5	%
de-fryske-marren-defryskemarren	61	5	%
reimerswaal	147	10	%
hollandskroon	6717	10	69%
haarennb	2	1	%

B Computer specifications

- Intel Core i5-4690K @ 3.50Ghz
- Nvidia GeForce GTX 1070 Ti
- 12GB DDR3 RAM @ 1333MHz
- Windows 10 Pro x64

C Models for every organisation

Organisation	F_1 **score**	**macro** F_1 **score**	**weighted** F_1 **score**	**Support**
Arnhem	0.97	0.96	0.97	4040
Beekdaelen	0.87	0.90	0.87	1400
Best	0.84	0.86	0.84	895
Brummen	0.84	0.85	0.83	1069
Buch-gemeenten	0.90	0.89	0.90	520
Bunschoten	0.89	0.89	0.89	65
Defryskemarren	0.87	0.88	0.87	194
Doesburg	0.92	0.92	0.92	298
Drechtsteden	0.91	0.92	0.91	7585
DuivenWestervoort	0.16	0.02	0.05	1844
Echtsusteren	0.85	0.85	0.85	564
Ijsdenmargraten	0.85	0.84	0.84	413
Goes	0.81	0.82	0.81	791
Hlt	0.84	0.85	0.84	6398
Hoekschewaard	0.78	0.80	0.77	2155
Hollandskroon	0.80	0.78	0.79	3742
Kapelle	0.77	0.77	0.77	164
Katwijk	0.79	0.81	0.79	1410

Table 15: Organisations with their fine-tuned model