Mateusz Wozakowski

Automatic Detection of German Dialects Around the World

BSc Computer Science

25/03/2022

I certify that the material contained in this dissertation is my own work and does not contain unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation.

Regarding the electronically submitted version of this submitted work, I consent to this being stored electronically and copied for assessment purposes, including the school's use of plagiarism detection systems in order to check the integrity of assessed work.

I agree to my dissertation being placed in the public domain, with my name explicitly included as the author of the work.

Name: Mateusz Wozakowski

Date: 24/03/2022

Abstract

This project investigates the efficacy of a range of different machine learning models at detecting unseen texts of 12 different German dialects from around the world, using data provided in the AGD (Archiv für Gesprochenes Deutsch/Archive for Spoken German). The model results are then analysed to extract useful information about the similarities and differences of these dialects, such as exploring the relationship between distance and dialect similarity.

The project will involve the design, creation and testing of multiple classical, and word embedding models, with the use of performance measurement and data analysis techniques to compare both the models and the dialects.

The results found that the classical models generally performed better. The best model, when tested on all the dialects, achieved an F1-score of 0.70. This was achieved using a classical TF-IDF model with a Support Vector Machine algorithm. The data also showed there to be a weak negative relationship between dialect similarity and distance.

Contents

1. Introduction

Natural Language Processing (NLP) can be defined as a theoretically motivated range of computational techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications (Liddy, 2001). It is a powerful tool within artificial intelligence with many uses, such as sentiment analysis, machine translation, and for this project, natural language detection.

German is a language which has hundreds of different dialects (Gross, 2009), spoken not only in Germany and neighbouring countries, but also less commonly, across multiple continents, largely as a result of colonialisation, historical immigration and asylum seeking of German speaking citizens abroad (United States Holocaust Memorial Museum, n.d.).

One of the difficulties in dialect detection, unlike language detection, is the relative scarcity of available datasets with distinct dialects of a language which must be used for training the model. A reason for this in the case of German, is that most formal communication is done in standard high German, meaning that most of the data available of regional dialects in written form will come from transcribed speech, as is the case in the dataset used for this project – in which spoken German from different parts of the world is transcribed from various studies and projects.

Another challenge within dialect detection is the relative similarity of dialects, as compared to language detection, in which languages are usually much more distinct, this makes dialect detection more difficult as different dialects of one language are likely have more common features than completely separate languages, meaning that there are fewer features by which we can distinguish them.

An investigation into how these less common dialects of German compare to standard German spoken in central Europe and other lesser-known dialects from other parts of the world may prove useful in linguistic research, in addition to this, this project could have uses in fields such as location detection, based on the regional dialect which a user may input.

In this project, I will investigate the efficacy of different NLP models at correctly classifying provided text using a supervised machine learning approach, as well as comparing the timeliness of these models. In addition to this, I will analyse the results to find useful information about the dialects in the dataset, such as comparing the similarity between them, and investigate how they vary based on the location from which they originate, I predict that there will be a negative relationship between the distance between dialect origin, and dialect similarity.

The structure of this report will be as follows:

In the Background chapter (2), I will discuss previous work done into both language detection and dialect detection.

The Data Collection chapter (3) will describe the dataset used for this project, as well as how the data will be used.

The Method chapter (4) will outline and explain the methodology used to create the dialect detection models, and the data analysis tools which will be used.

In the Results chapter (5), I will show the results from the experimentation described in the previous chapter.

The discussion chapter (6) will discuss, analyse, and put into context the results of the experimentation.

The conclusion (7) will reflect on the successes and failures of the project, in comparison to the original aims and objectives originally set out.

2. Background

2.1. Classification

Text classification is one of the largest applications of Natural Language Processing, related applications to this project include topics such as:

Sentiment Analysis, in which NLP techniques are used to determine whether pieces of text are positive, negative, or neutral, or a quantified amount, this can, for example be used to detect the sentiment within an online review for a product, allowing a business to numerically gauge the reception of their product.

Spam detection, using language to identify the likelihood of e.g., an e-mail containing spam, or malicious content, allowing e-mail service providers to filter out received e-mail, which is likely to be spam, or harmful to its users.

News Classification – Some news websites use NLP models to save time on manually classifying news articles into different categories, such as technology, politics, business etc. In recent years, there has also been an effort made by social media sites to detect fake or misinformative news shared by its users using NLP, and either tagging or completely removing it to stop the spread of misinformation on their sites.

2.2. Language Detection

Language detection, or language identification is another field under text classification, in which NLP techniques are used to determine which natural language a given piece of data is in. Today, this has many uses, such as within machine translation, and web search optimisation, allowing end users to receive search results which are most relevant to them, or their language. Furthermore, language detection also has uses within some of the other applications of text classification, such as spam detection services which have multi-lingual support must first use natural language detection to identify the language in which content is written in, before being able to apply spam detection filters. Essentially, language detection is an extremely important component within text classification and can be applied in almost every context in which multiple natural languages are observed.

A field very closely related to language detection is Native Language detection, in which the native language of a writer is guessed, based only their writing in their second language. One paper by Chan et al (2017) explores the use of different Ensemble methods (See chapter 4) to detect writers' native languages, based on their written responses to a standardized English proficiency assessment, there were 11 different native languages originating from Europe (E.g. German, Italian, French), Asia (E.g. Japanese, Hindi, Korean), as well as Arabic. Chan et al used different methods combining support vector machines (SVM) and Fully Connected Neural networks (FCNN) to classify their data, since these algorithms consistently outperformed classifiers like Multinomial Naïve Bayes and perceptron. The outputs of the SVM and the FCNN were combined using a voting scheme, a linear discriminant analysis (LDA), and multiple LDA classifiers in combination with a voting scheme. They tested the models with multiple different configurations, such as using different word, character and phoneme N-grams, and Part of Speech (POS). In the end, they found that their best results came from an SVM ensemble based on word bigrams and trigrams, character 4-grams and 5-grams, and phoneme 4-grams and 5-grams, this resulted in an optimal F1 score of 0.8730, showing that the SVM generally outperformed their best FCNN ensemble which got an F1-score of 0.8560, a combination of the main classifiers did not lead to an improvement over pure ensembles of either type.

A paper by Cavnar and Trenkle (2001) studied and compared the use of different N-gram frequencies (See chapter 4) at identifying languages and subjects, by looking at how often different N-grams appeared in each language/subject. They used a dataset comprised of news articles from ten different European languages, although some originating from outside of Europe (E.g., Mexican Spanish). They found that the top ~300 N-grams always correlated to the language, meaning that a long piece of text about one topic in English would have a very large number of common N-grams as another English piece of text about a completely different topic, whereas N-grams in another language would have a very different distribution. At the point past the 300 most common N-grams, the frequency profile started to show N-grams more specific to the subject of the text, representing terms which occur frequently in documents regarding that subject. When it came to their language identification results, they found that their classification process worked only a little better on longer articles, and not as much as they expected. In addition, they found that using the top 400 N-grams from each language, their model could detect the language which an article was in almost perfectly, reaching an accuracy of 99.8%.

Another more classical method of language identification using a statistical classification algorithm is demonstrated by Elworthy (1999), who used Bayes' rule to find the probability of the language given the input tokens for each language, meaning that each token is treated as an independent event. The probability of a token occurring in the language was calculated under the assumption that the probabilities follow a binomial distribution, thus treating token occurrences as binary events. The algorithm made a decision as soon as one could be made reliably over a certain confidence limit. In this paper, Elworthy used a collection of corpora from the European Corpus Initiative which contained 18 different languages, taking data from newspaper corpora where possible, and novels or literature where not, the interesting part of this classifier is that the statistical process provides feedback about itself through the use of the confidence limits (Which are based on the model), meaning that presenting results without adequate evidence is avoided. The algorithm's accuracy ranged from 71.4% using all tokens to a very accurate 99.1%, only slightly lower to Cavnar and Trenkle's N-gram model.

2.3. Dialect Detection

Within the field of language detection, the large bulk of work done on language detection has been based on distinguishing between natural languages as mentioned above. There is a limited amount of work that has been done on dialect detection, which focuses on identifying different dialects of the same language, originating from either different parts of the same country, or from different countries altogether.

Goncalves and Sanchez (2015) partially tackle the issue by using geographically tagged Spanish tweets on twitter. They scraped a sample of all tweets produced between May 2010 and June 2015, they used a language detection tool to filter out all non-Spanish tweets, which left them with 106 million geolocated tweets originating mostly from Europe, Latin America, and North America, with this, they created a corpus and found the geographical distribution of how often different words with the same meaning are used around the world. They then created cells and grouped them into K clusters which were similar to one another, using this, they found that the linguistic differences in tweets around the world can be split into two well defined clusters which they called superdialects α and β. However, these superdialects were not localised around a definite region, since both groups were present in all Spanish speaking countries, but they hypothesised that these superdialects came about as a result of urbanisation, as α was much more common in urban centres, likely relating to official media, thus α was named an "international Spanish", whereas β was more linguistically heterogenous and originated from more rural areas around the world. One of the interesting

things about the corpus used in this paper is that, unlike a lot of language and dialect work which is usually done based on formal, written languages, such as from newspapers and literature, they used tweets which allowed for more informal language to be analysed. However, in this paper they did not attempt to create a dialect detection model using the corpus they created, it would be interesting to see how well different NLP models perform at classifying tweets originating from different countries, or even between their proposed superdialects α and β.

An example of work done on dialect detection is the paper by El-Haj (2020), in which a multi-dialect and multi-national corpus is introduced, which is comprised of Arabic songs from 18 different countries. Further, there were two main experiments done on the corpus, one which involved dialect detection of the six main Arabic dialects, and one involving country-of-origin identification. The experiments used and compared classical and deep learning models, including the use of Multinomial Naïve Bayes, Logistic Regression and Support Vector Machine for the classical models, and various different neural network models, using two different continuous bag of words word embedding models. When comparing the results with all six dialects were placed against each other, the classical models generally outperformed the word embedding models, with the highest test accuracy of 72.6% achieved using the multinomial Naïve Bayes algorithm, whereas the word embedding models achieved results ranging in the ~45-65% range. However, the results were much closer when tested on fewer dialects, particularly when set as a binary classification problem, the word embedding model using a convolutional neural network outperformed the naïve Bayes model with 93.0% accuracy, compared to 92.6%.

Even more closely related to this project is a paper by Malmasi and Zampieri (2017), which was a submission for a German Dialect Identification competition task at the VarDial Evaluation Campaign. This task was based on interview transcripts consisting of four different dialects all originating from Switzerland, for their entry, they created three different systems; a plurality ensemble, a mean probability ensemble and a meta-classifier trained on character and word N-grams, using Support Vector Machines as their classifier, and Random Forest as their meta classification algorithm. They found that their meta classifier, using character n-grams, particularly character 3, 4, and 5-grams greatly performed word features, achieving up to 85.13% accuracy with character 4-grams. The team ranked first in the competition, with a weighted F1 score of 0.662. They also found that the four dialects were not equally difficult to be identified, particularly the dialect from Lucerne which scored only a 34.4% accuracy, only slightly higher than their baseline of 25.8%. Conversely, the Zurich dialect obtained a high 91.8% accuracy, and was also the one which created the most confusion between the other three dialects, suggesting that the algorithm tried to label most of its predictions to Zurich to maximise its performance. Their maximum F1 score of 0.662, although not as high as some in the previous papers on language detection, is quite impressive considering the very close proximity of the locations from which the tested dialects originated, it would have been interesting to see how these the proximity between these cities affects the similarity of these dialects.

I have not found there to be any research thus far on differentiating between German dialects spoken on the global scale, comparable to the scope in the work done by Goncalves and Sanchez (2015) with Spanish. A likely reason for this is because of the lack of data, and general low level of knowledge about the different dialects of German spoken around the world. This stems from the low population of native German speakers outside of the usual German speaking countries, and the fact that informal and colloquial speech is rarely

documented in a way which can be analysed, this gives rise to the importance of the dataset used for the project.

3. Data Collection

3.1. Dataset introduction

To create a supervised learning language classification program, a dataset of all our wanted dialects is needed to train the model to recognize features of each dialect, this means that the dialects which we can train and test is limited by the dataset.

The dataset which will be used for the project comes from the AGD (Archiv für Gesprochenes Deutsch), or Archive for Spoken German, which is an archive managed by the IDS (Institut für Deutsche Sprache), or Institute for the German language. The AGD contains a collection of 46 corpora of spoken German - many of them originating from different locations and times.

39 of the corpora from the AGD can be accessed for free through the DGD (Datenbank für Gesprochenes Deutsch, n.d.) once an account is made and approved. I have selected 12 different corpora which are suitable for the project.

Due to the terms laid out by the DGD, the use of their data is granted only for non-commercial use, the data cannot be passed on to third parties, as such I am not able to include the data used for this project.

The selection criteria used to decide which corpora were suitable for the project were based upon:

- The location from which corpus data was taken
- The format of corpus data (E.g., Availability of transcript, content of corpus)
- The size of corpus data

3.2. Selected Corpora

Australian German (Australiendeutsch – AD):

This corpus was created as part of a project in Monash University, it comprises 64 hours of interview and photo descriptions recorded between 1966 to 1973 by over 300 elderly people whose families have lived in Australia for three generations.

Hamburg Maptask Corpus (HaMoTiC – HMOT)

This corpus was created for a project about multilingualism at the University of Hamburg between October 2009 and September 2010. They conducted map tasks with 25 German learners with different levels of German proficiency, with the first language of the learners being represented with a wide variety – e.g. French, Polish, Farsi and some non Indo-European languages such as Turkish and Chinese. The speakers were between 17 and 40 years old with a higher level of education.

Flight and emigration to Great Britain (Flucht und Emigration nach Grossbritannien – FEGB):

This corpus was created during a research stay in Great Britain in 2017 by Eva-Maria Thüne from the University of Bologna. It contains interviews about the language and cultural identity of mostly Jewish migrants 70 to 80 after they immigrated to Great Britain from Nazi Germany, Austria and former Czechoslovakia in the 1930s. Most of the interviewees spoke German with a varying extent of English influence.

Berlin Turning Corpus (Berliner Wendekorpus – BW):

This corpus was created as part of a project at the institute for German and Dutch Philology at the Free University of Berlin. It constitutes multiple interviews of men and women between the ages of 19 and 55 who lived in both East and West Berlin, discussing the individual, social and economic aspects of life, taken between 1992 and 1996, following the fall of the Berlin wall in 1989.

German in Namibia (Deutsch in Namibia – DNAM)

This corpus was produced as part of a research project about German in Namibia funded by the German Research Foundation, which documents the language use and attitudes of the German-speaking minority in Namibia. The recordings consisted of interviews, free conversations and "language situations" simulating a formal or informal communication situation, taken throughout Namibia in partly German-speaking schools, farms, private buildings, and public spaces throughout 2017, the participants were people aged between 14 to 18, and 26 to 75 years.

German Dialects: GDR (Deutsche Mundarten: DDR – DR)

This corpus was created by the institute for German Language and Literature of the Academy of Sciences of the GDR. It contains a range of recordings from narrations, conversations to standard texts all originating from the GDR between 1960 and 1968.

German of Turkey Returnees (Deutsch von Türkeirückkehrern – DTRK)

This corpus was collected as part of a project at the department of German language and literature at Marmara University in Istanbul in 2011. The data consists mainly of autobiographical narrative interviews about the migration experiences of German language and literature students at Marmara University, who were born in Germany or came to Germany as children and grew up there and went through the German education system partially or completely.

Escape Stories from East Prussia (Fluchtgeschichten aus Ostpreussen – FGOP)

This corpus was collected as part of a project at the University of Turin. It consists of narrative autobiographical interviews with now elderly survivors from East Prussia, who recount events of the last months of World War II, taken between 2015 to 2020.

Emigrant German in Israel (Emigrantdeutsch in Israel – IS)

This corpus is one of three compiled as part of a project by the German Research Foundation on the language and cultural identity of German speaking emigrants 50-60 years after their immigration to Palestine/Israel. The corpus consists mainly of autobiographical interviews compiled between 1989 to 1994 in the private homes of interviewees. The interviews had some consistent themes, such as childhood and youth in Germany, experiences of antisemitism, and emigration.

Emigrant German in Israel: Viennese in Jerusalem (Emigrantdeutsch in Israel: Wiener in Jerusalem – ISW)

This corpus was formed as the first expansion to the original Emigrant German in Israel project. Most of the data collected for this corpus was collected in 1998, consisting of autobiographical interviews of Elderly Jewish men and women aged 69 to 90 who were born in Austria, mostly in Vienna, the majority of whom were rescued by Youth Aliyah Child Rescue, meaning they moved without their parents.

German Dialects: Former German eastern territories (Deutsche Mundarten: ehemalige deutsche Ostgebiete – OS)

This corpus was created by the German Language Archive (DSAv) in cooperation with the Research institute for the German language as part of a project documenting dialects of the closed German language area in the Former eastern territories of Germany, also including German speaking islands in Eastern and Southeastern Europe. The corpus compromises various speech events of 987 elderly migrants from the former German eastern territories, such as conversations and standard phrases (Days of the week, numbers etc.) recorded between the years of 1962 and 1965.

Russian-German Dialects (Russlanddeutsche Dialekte – RUDI)

This corpus was created as part of several projects at the universities of Tomsk and Omsk in Siberia, between the years of 1959 and 1989. It contains authentic speech recordings from The German language islands in the eastern part of the former Soviet Union, including a range of narratives, interviews and memories of the Russian-Germans, reflecting many faces of their life and culture.

3.3. Dataset Details

Corpus:	Number of sentences	Total Word Count:	Unique Word Count:
Australian German	2879	53375	4995
Hamburg	3604	30328	2648
Great Britain	5852	107614	9544
Berlin	4038	176777	15825
Namibia	2979	35854	4326
GDR	4213	71292	7403
Turkey	8503	93962	7595
Prussia	6711	82268	5806
Israel – Standard	2877	136617	11319
Israel – Viennese	2606	79415	7329
Eastern Territories	4464	236610	12776
Russian	2201	99015	11124

Table 3.1: Sentence and word count by corpus, after pre-processing

From this set of corpora used, the data will be randomly split into a training and testing set. The training set is used to train the program to learn the features of each different dialect, thus creating a model which should be able to accurately classify an unseen piece of text based on the data on which it was trained.

The testing set will be used on the model as the unseen data which has not been used to train it, it will predict which label most accurately fits each given sentence/text from the testing set.

The training and testing split can affect the overall accuracy of the model, generally, the more data which is used for training, the more accurate the model will be. However, if we do not have enough testing data, then the variance of our performance statistic may be too high,

depending on the absolute number of cases which are available - leading to a suboptimal or inaccurate result of the model.

After some testing, I have decided to use a split which consisted of 80% of the data being used for training, and the remaining 20% for testing, as this seemed to give me the best results for which I did not have to worry about performance variance, as compared to e.g., a 90% training, 10% testing split, especially since the absolute amount of data in the corpora is not exceedingly large.

4. Method

4.1. Programming Language and Libraries

The chosen programming language for this project will be Python 3.8 (Van Rossum and Drake, 2009), this is due to its flexibility for machine learning as well as the large amount of well documented, available libraries relating to NLP.

One of the main libraries I will be using is Scikit-learn (Pedregosa et al, 2011) (Abbreviated to sklearn), this is a library built on NumPy, SciPy and matplotlib, with a wide variety of uses in machine learning. It contains many useful classification and regression algorithms, as well as other utilities for pre-processing, splitting data into training/testing sets, and a range of functionality for metrics, and a pipeline transformer, all of which will be useful for the model creation and data analysis section of this project.

Another library that will be used for this project is Gensim (Rehurek and Sojka, 2011). This is an open-source library designed for processing texts using unsupervised machine learning algorithms, such as Word2Vec, which will be used in this project.

Other external libraries that will be used include:

- Pandas (McKinney, 2010) for its data manipulation utilities
- NLTK (Bird et al, 2009) for its corpus of stop words and other NLP utilities
- Numpy (Harris et al, 2020) for its mathematical functions with vectors
- Matplotlib (Hunter, 2007) for its graphing functions

4.2. Data pre-processing

Data pre-processing, or in the case of NLP, text cleaning, is the process of preparing raw text for Natural Language Processing so that machines can more effectively process human language (García et al, 2015).

Many of these techniques work to reduce the noisiness of our data, noise in data can be defined as additional meaningless information, which adds ambiguity while training the model.

This process is done through multiple stages:

Punctuation Removal – The removal of punctuation (e.g. !, ?, ‘,.) is done to reduce the noise of the data. Within my program, this is done by creating a dictionary using Python’s ‘string.punctuation’ constant, replacing all instances of punctuation with a null value, thus removing it from the text.

Lowercasing the data – This is done for simplicity (reducing noise) and to avoid any errors while performing case-sensitive NLP tasks, such as removing stop words (See below). This task is fulfilled by the use of Python’s .lower() function.

Tokenization – This is the process of splitting whole sentences into smaller units (e.g. words, characters, or subwords), which are known as tokens. This allows computers to process natural language, for example, by creating a vocabulary, which is a set of unique tokens in the corpus (Webster and Kit, 1992). For my program, this is done using sklearn’s vectorize.

Removal of stop words – Stop words are words which are used in natural language however have little to no contribution to the meaning of the text, for example in English words such as I, me, they, themselves, etc. Thus removing these will contribute to reducing the noise in our dataset. In my program, I use NLTK’s German “Stopwords corpus”, since it is one of the most complete corpora on German stop words, nevertheless, an issue to consider is that this corpus

is limited to Standard German stop words, meaning that any colloquial language from the various dialects in my dataset may not be represented in the corpus, and will remain in the data after the pre-processing is complete.

Regular Expressions – A Regular Expression is a sequence of specific characters, or types of characters which specifies a pattern in text. Due to the nature of our dataset being in the form of a transcript, it contains the names of the speaker before each line. Keeping these in would lead to a large amount of overfitting, so a regular expression is used to remove them. In addition, any data case which was repeated in one of the classes has been removed, so that the test cases would not include any cases which the model has already seen in the training phase.

4.3. Imbalanced Data

Due to the different sizes of the corpora, many of the dialects are more or less represented in the overall dataset. An imbalanced dataset can cause bias in the predictive modelling since the model would be better trained to detect one dialect over another (Sun et al, 2009).

Usually to correct an imbalanced dataset, undersampling or oversampling is used.

4.3.1. Undersampling

Undersampling is the process of reducing the number of cases from the majority class(es) to a comparable level of the minority classes, thus making it balanced:

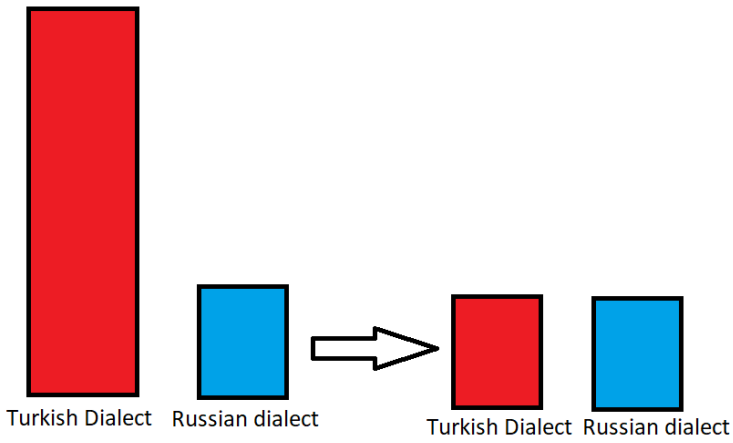


Figure 4.1: Undersampling visualisation

However, there are some problems with undersampling, mainly the loss of useful data which comes from simply removing so many of the data cases from the majority class.

4.3.2. Oversampling

The opposite technique of undersampling is oversampling, this is where the data from the minority classes is artificially increased:

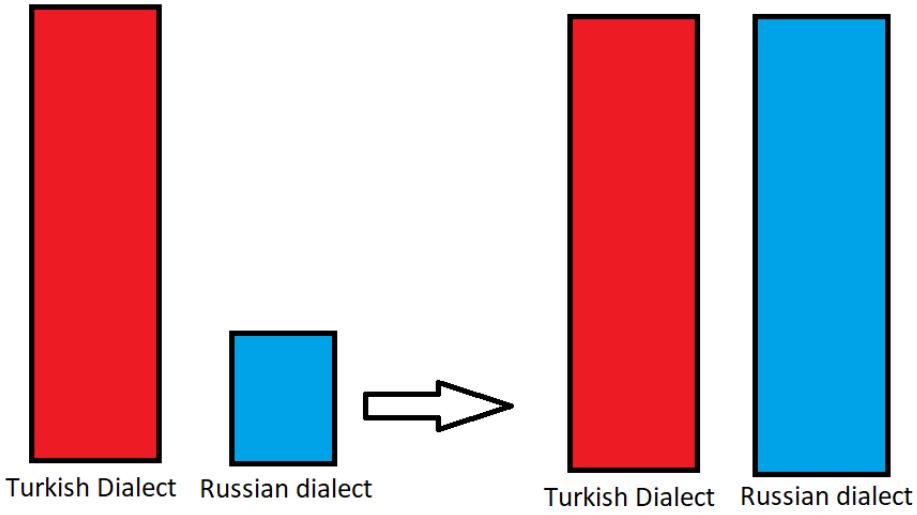


Figure 4.2: Oversampling visualisation

This can be done through different methods, such as randomly duplicating data points from the dataset into the training set, however, this does not actually result in any new information for the model.

Another technique used for oversampling is called SMOTE (Synthetic Minority Oversampling Technique). This involves creating new synthesised data points from the existing data, it uses the k-nearest neighbour algorithm to select a random nearest neighbour and then create a synthetic instance in the feature space (Chawla et al, 2002).

However, oversampling also has its issues, such as increasing the possibility of overfitting, by making exact or similar copies of existing data, in addition to increasing the computation time caused by the replication of existing samples or the creation of synthetic ones.

For my model, I have used a combination of undersampling and oversampling using SMOTE, in order to minimise the disadvantages caused by the two respective techniques. The degree to which either method is used depends on the corpora in the test groups.

4.4. Model Training

As part of the investigation, I have decided to create both a classical TF-IDF machine learning model, and one using a more complex word embedding method.

4.4.1. Classical Model

Term Frequency – Inverse Document Frequency

My first model uses Term Frequency – Inverse Document Frequency (TF-IDF) vectorisation. This works by using the Term frequency, which is a measure of how frequently a term t appears in a document d:

$$Term\ Frequency = \frac{Number\ of\ times\ term\ t\ appears\ in\ document\ d}{Number\ of\ terms\ in\ document\ d}$$

The term frequency is then multiplied by the Inverse document frequency:

$$Inverse\ Document\ Frequency = \ln \frac{Number\ of\ documents}{Number\ of\ documents\ with\ term\ t}$$

Combining these gives us the TF-IDF:

$$TF - IDF = Term\ Frequency * Inverse\ Document\ Frequency$$

In effect, this gives each word a value of how important it is to every document in each corpus. Due to the formulae above, common words in every corpus, such as connectives like "and" or "but" will be given a low TF-IDF score, whereas words which are rare in all documents, but common in a single document will be given a high TF-IDF score.

Word and character grams

I have decided to experiment with the efficacy of the models when using word, character and n-gram tokenisation.

In the most basic sense, an N-gram is a sequence of N words, for example in the sentence "The door is red", a 1-gram (unigram) system would split the sentence into each word, whereas a 2-gram (bigram) system would output: "The door", "door is", and "is red". These tokens are then what is represented by a term in the TF-IDF statistic.

This concept can also be applied to characters, so instead of separating each document by its words, we separate it by characters, or groups of characters. This generally leads to a more efficacious model; however, it also uses more computing power since the overall number of terms will be much greater.

For my model, I have decided to test a simple word unigram, as well as character n-grams ranging from 1 to 4 to see which n-gram values perform the best.

4.4.2. Word Embedding Model

Word embedding refers to the numerical representation of words used for NLP. Usually, they are mapped to vectors which encode their meaning in a way that similar words will be closer in the vector space, for example, the words "cat" and dog" would be close in the vector space, relative to something such as "cat" and "Earth". This meaning is extracted through various features, including surrounding words, meaning that word embedding can take into consideration the context in which a word is placed.

There are many different techniques used which can convert words into their numerical representation, such as Word2vec, fastText, and GloVe.

Word2Vec

The chosen word embedding technique used in this project is Word2vec (Mikolov et al, 2013). This is a technique created by a team at Google in 2013, which utilises a neural network architecture.

Word2Vec works by encapsulating the whole body of corpora in a vector space, with each word within the text being mapped to a vector of a fixed dimensionality, with each vector having an orientation meaning that two vectors can be compared by calculating their cosine similarity, this is the cosine of the angle between the two vectors. This means that the similarity between them is dependent only on the angle between them, and not their overall magnitudes.

The word embeddings are produced using a neural network with one hidden layer, with its weights being adjusted in the training process. These hidden weights are then taken and used as the word embeddings outputted by Word2Vec. Word2vec consists of two methods, Skip-Gram and Continuous Bag of Words (CBOW).

Once the words have been converted into vectors, an average of them is found for each instance of text, giving us a vector corresponding to each instance, which will be used as a representation of an instance which will be fed through a classification algorithm to train and test.

Skip-Gram

In the Skip-Gram model, the representation of a given word is used to predict the context words, meaning the words which surround the word, taking a word w(t) as its input, and outputting the words most likely to be surrounding it, e.g. w(t-1), w(t+1) etc.:

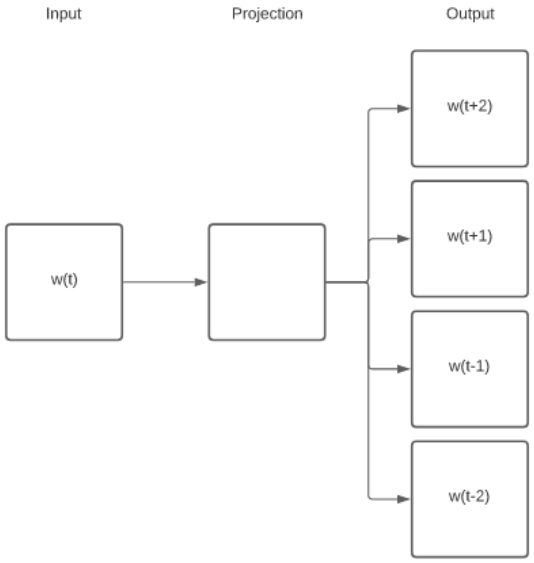


Figure 4.3: Skip-Gram model

Continuous Bag of words

The Continuous bag of words (CBOW) model works the opposite way to the skip-gram model, the words surrounding the target words are used to predict the target word, so, the input would be e.g. w(t-1), w(t+1) etc. and the output would be the predicted w(t):

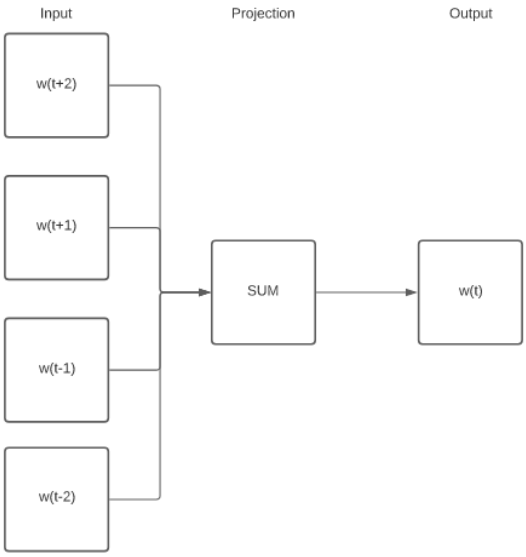


Figure 4.4: CBOW model 1

After some testing, I decided to use the Skip-Gram model, since it seemed to produce better results in most cases.

4.5. Classification Algorithms

Once we have the TF-IDF statistic or the word embeddings for our corpora, the next step was to run machine learning classifiers through the training set, training the models to the labels and then using this to predict labels for the testing set.

I have decided test multiple classification algorithms to see which performs the best at differentiating between dialects and create the best possible model. The chosen algorithms are Multinomial Naïve Bayes, SVM, and Random Forest.

4.5.1. Naïve Bayes

The Naïve Bayes classifier is a simple probabilistic classifier which is built on the Bayes theorem (Rish, 2001), this is a simple but foundational theorem in probability, which states the probability of an event, based on prior information of conditions related to the said event. It can be represented mathematically as:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Where A and B are events and $P(B) \neq 0$

At a high level in the context of this project, the Naïve Bayes classifier works by calculating the probability of a given text instance being each dialect that is being tested, and then outputs the dialect which has the highest probability. This works by using the TF-IDF weights to find the most likely dialect which a given test text instance belongs to.

The implementation I am using is the Multinomial Naïve Bayes algorithm from the sklearn library, which works best for classification with discrete features, such as word counts, or in this case the TF-IDF feature counts.

Naïve Bayes is called 'Naïve' because it assumes that features are completely independent of one another, which is very rarely true, particularly not in the case of dialect detection. For this reason, I will not be using Naïve Bayes to test the word embedding model.

A character unigram Naïve bayes model will also serve as one of my baseline measurements to compare my other models against, since naïve bayes is a very fast and elementary algorithm within the classification algorithm space.

Support Vector Machine (SVM)

Support Vector machines are a set of supervised learning algorithms with generally fast speed and good performance (Cortes and Vapnik, 1995). SVM works by plotting each data instance is plotted in an n-dimensional space, n depending on the number of features. With the magnitude of each feature being placed on that dimension on the space, thus giving each data instance a certain coordinate.

Then, a hyper-plane is found which best differentiates the classes by maximising the distance between the two classes, this distance is known as the margin, the data points which lie on the margin are called support vectors. As the dialect detection program for this project is a multiclass classification problem, and SVM does not natively support multiclass classification, the problem is split into multiple binary classification problems. A visualisation of SVM is shown in figure 4.5:

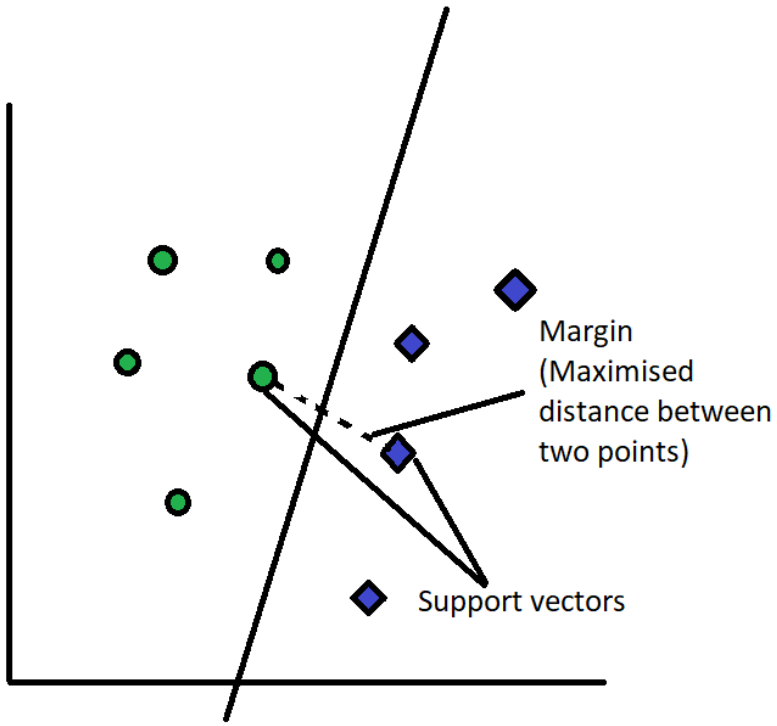


Figure 4.5: SVM visualisation

In addition to this, there is a range of kernel functions which determine how the classes are separated, including the efficiency and smoothness of the separation. Some examples of different kernel functions include linear, polynomial, and sigmoid.

When the data instances from the training set have all been mapped out onto the n-dimensional space and a hyperplane has been calculated, any data inputted into the model (i.e., from the testing set), will be classified based on which side of the hyperplane it falls on.

The implementation of SVM used for the dialect detection in this project will be sklearn's C-Support vector classification (sklearn.svm.SVC), For the sake of this project, I will use a linear kernel function, which is one of the simplest and most commonly used functions and using the default 1.0 as the regularisation parameter.

4.5.2. Random Forest Classification

Ensemble classification

Ensemble classifiers consist of a set of classifiers from whose decisions are combined to create a final classification verdict (Diettrich, 2000), the combination of multiple classifiers generally allows for a more accurate model with less bias and variance when compared to using a single classifier.

Random Forest is an ensemble machine learning classifier, which uses a combination of decision trees

Decision Trees

Decision trees are a supervised learning method used for classification which works by predicting values of a target variable by learning decision rules deduced from the data features (Rokach and Maimon, 2005).

A decision tree consists of two different types of nodes, decision nodes and leaf nodes, and they all start with a root (decision) node at the top of the tree. Decision nodes contain a condition which allows the data to be split down the tree, leaf nodes help decide the class of a data instance.

The training data set is broken down into increasingly smaller subsets which contain data instances of similar feature values, while the decision tree is developed. When developing the decision tree, a number of conditions are tried, and the resulting data split from the output of each condition is used to calculate the entropy of the state. Entropy is a measurement of the information contained in the state, where a lower entropy value results in higher certainty of a randomly picked data instance's class. For example, the entropy at the root node would be 1 (maximum value), and 0 at a pure node, whose data belongs to a single class.

$$Entropy = \sum - \ p_i \log (p_i)$$

Where p_i = probability of class i.

The decision tree calculates the entropy for every possible condition and chooses the one with the lowest entropy value, which signifies the maximum information gain from the condition.

When using the created decision tree to classify test data, each data point goes through the decision tree and the leaf node at which it ends up at is the prediction of the classifier.

Random Forest

As stated, the random forest classifier consists of an ensemble of trees with a low correlation (Ho, 1995), which operate as a committee to decide the final class of a data instance, this allows for less overall error since a large group of decision trees protects the classifier from being misled by an error which pertains only to an individual tree, assuming all of the trees do not make the same error. This means that the greater number of trees used in the random forest, the probability of correctly classifying a data instance increases.

Random forest works by creating multiple new smaller datasets from the original dataset, each new dataset being a part of the original, the datasets created are of equal size and can contain duplicates since the dataset is sampled with replacement, this process is known as bootstrapping. Since the form of the decision tree is extremely sensitive to the data which it is

trained on, small changes in the training datasets will create very different decision trees which will make up the random forest.

It is also important to note that every feature will not necessarily be used for training the trees, but a random subset of features will be selected for training each tree, meaning that each tree uses a different set of data, and uses different features to make its decision.

Once all the trees have a prediction, the results are aggregated, and the majority vote is taken as the final decision of the random forest.

The implementation of the random forest algorithm used for the project is the default sklearn ensemble.RandomForestClassifier algorithm which uses 100 decision trees in the forest.

4.5.3. Dummy Classifier

In addition to the classifiers mentioned above, I have also tested each group of corpora on a dummy classifier. This is a classifier which makes predictions that ignore input features and will be used as a baseline to compare against our other models to see how well they actually perform.

A "most frequent" strategy will be used for this classifier, which will always predict the most frequent label in the training set. For example, in a binary classification problem with "True" and "False" as the labels, if 90% of the training data was "True", then the classifier would always predict "True" and get a misleadingly good 90% accuracy value. Therefore, having this as a baseline will inform us about how good our models actually are, compared to, e.g., randomly choosing labels.

4.6. Data Analysis Tools

4.6.1. Precision

Precision is a performance metric used within machine learning, it is defined as the ratio between true positives predicted and all positives, for example, an 80% precision value for the Australian dialect would mean that of the times it predicts Australian, it is correct 80% of the time. Precision is calculated using the following formula:

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$$

Effectively, the average precision value of our model tells us the probability of our model making a correct prediction when it predicts a positive value.

4.6.2. Recall

Unlike precision, recall is a performance metric defined as the ratio of correctly classified true positives to total amount of positive data samples, for example, an 80% recall rate for the Australian dialect means that the model correctly predicted Australian when it came across it 80% of the time. It is calculated using the formula:

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives(FN)}$$

The recall is independent of how negative data samples are classified, so if the model classified all Australian samples as Australian, the recall would be 100%. For this reason, we use a combination of precision and recall.

Generally, there is a trade-off between precision and recall, meaning that at higher levels of precision, the recall suffers, and vice versa. This is useful for certain models which place a higher priority on one of these values, meaning that a model can be tuned to optimise one of these at the cost of the other. However, as these are equally important for this project, this will not be done. In addition, I will use the weighted average value from sklearn's metrics accuracy score, which takes into account each class's support value, or the number of true instances per label.

4.6.3. F1-score

The F1-score is a measure of the model's accuracy, calculated by using a combination of precision and recall of the model (Goutte and Gaussier, 2005):

$$F1\ Score = 2 \times \frac{Precision\ \times Recall}{Precision + Recall}$$

The F1-score is generally a more useful metric in classification problems when compared to the intuitive metric, accuracy:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ total\ predictions}$$

This is because accuracy does not take into consideration the distribution of the data, unlike the F1-score.

The F1-score allows us to create a model and optimise it, since it is indicative of both the precision and recall values, rather than having to juggle between them. For this reason, the weighted F1-score will be the main metric used to evaluate the overall efficacy of our models.

4.6.4. Confusion Matrix

A confusion matrix is another performance measurement/technique used within machine learning, it allows for more detailed understanding of a model by showing what the model is getting right and what errors it is making in classification (Ting, 2017). A simple confusion matrix for a binary classification problem is shown below:

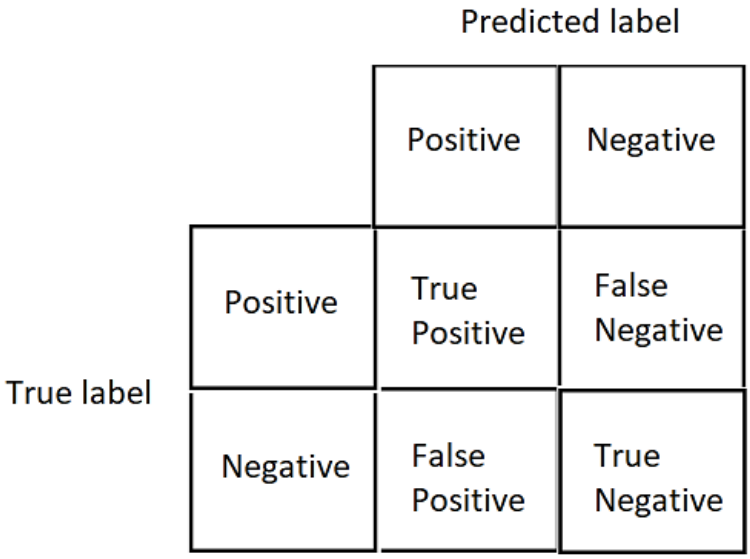


Figure 4.6: Confusion Matrix example

The confusion matrix is made up of two axes, reading along the predicted axis shows us how the model predicted each data point in the testing set, compared to the true axis, which shows the actual label of each data point, meaning that the values along the major diagonal show how many cases were correctly predicted for each class by the model, and the remaining squares show where the model was mistaken.

This will allow us to see in what way our models get "confused" when they make predictions, meaning we can see which dialects get mistaken for one another frequently, and which ones do not, which will be a useful tool for data analysis

4.6.5. Time

I will also measure the elapsed time of each model on my group which contains all dialects, to get the 'worst case scenario' (With optimal settings, compared by F1-score), to see how time-efficient each model is, in comparison to its accuracy.

4.6.6. Word Heat Map

A heat map is a technique which allows us to represent our data in two dimensions, with the use of colour to represent the values. For this project, we will find the common vocabulary in each dialect, and map it on a heat map, this will allow us to see which dialects are most similar and dissimilar one another by the words which are used in each corpus.

5. Results

To get results from the model, I have decided to put the corpora into multiple groups based on the location from which they originated, and one group including all the dialects, this will provide me with sufficient results to analyse their similarities and to test the efficacy of the models when tested against a reasonable number of other dialects, from a relatively similar geographical area.

The groups were sorted as follows:

5.1. Group 1 – Dialects originating in, or close to Germany:

Berlin Turning Corpus

Hamburg Maptask Corpus

German Dialects: GDR

German Dialects: Former German eastern territories

	TF-IDF - Naïve Bayes	TF-IDF - SVM	TF-IDF – Random Forest	Word Embedding - SVM	Word Embedding – Random Forest
Word	0.82	0.82	0.78	0.72	**0.75**
Character 1-gram	0.60 (B2)	0.62	0.68	X	X
Character 2-gram	0.76	0.81	0.77	X	X
Character 3-gram	0.82	**0.86**	0.80	X	X
Character 4-gram	0.81	0.86	0.80	X	X

	Dummy Classifier – Most Frequent
Accuracy	26.9%
F1-Score	0.11 (B1)

Table 5.1: Group 1 Results

3-gram SVM	Precision	Recall	F1-Score
Berlin	0.86	0.89	0.87
East Germany	0.80	0.82	0.81
Eastern Territories	0.87	0.81	0.84
Hamburg	0.91	0.93	0.92
Weighted Average	0.86	0.86	0.86

Table 5.2: Group 1 Classical Results

Word Embedding – Random Forest	Precision	Recall	F1-Score
Berlin	0.82	0.82	0.82
East Germany	0.65	0.71	0.68
Eastern Territories	0.75	0.72	0.73
Hamburg	0.80	0.75	0.78
Weighted Average	0.75	0.75	0.75

Table 5.3: Group 1 w2v Results

5.2. Group 2 – Distant European dialects:

Flight and emigration to Great Britain

German of Turkey Returnees

Escape Stories from East Prussia

Russian-German Dialects

	TF-IDF - Naïve Bayes	TF-IDF - SVM	TF-IDF – Random Forest	Word Embedding - SVM	Word Embedding – Random Forest
Word	0.79	0.80	0.75	0.62	0.64
Character 1-gram	0.53 (B2)	0.60	0.62	X	X
Character 2-gram	0.71	0.77	0.74	X	X
Character 3-gram	0.80	**0.82**	0.77	X	X
Character 4-gram	0.81	0.82	0.78	X	X

	Dummy Classifier – Most Frequent
Accuracy	27.1%
F1-Score	0.12 (B1)

Table 5.4: Group 2 Results

3-gram SVM	Precision	Recall	F1-Score
Great Britain	0.79	0.78	0.78
Prussian	0.76	0.79	0.78
Russian	0.99	0.95	0.97
Turkish	0.81	0.80	0.80
Weighted Average	0.82	0.82	0.82

Table 5.5: Group 2 Classical Results

Word Embedding – Random Forest	Precision	Recall	F1-Score
Great Britain	0.61	0.55	0.58
Prussian	0.54	0.57	0.55
Russian	0.96	0.93	0.94
Turkish	0.62	0.64	0.63
Weighted Average	0.64	0.64	0.64

Table 5.6: Group 2 w2v Results

5.3. Group 3 – Non-European Dialects

Australian German

German in Namibia

Emigrant German in Israel

Emigrant German in Israel: Viennese in Jerusalem

	TF-IDF - Naïve Bayes	TF-IDF - SVM	TF-IDF – Random Forest	Word Embedding - SVM	Word Embedding – Random Forest
Word	0.73	0.72	0.64	0.49	0.54
Character 1-gram	0.48 (B2)	0.50	0.55	X	X
Character 2-gram	0.64	0.67	0.63	X	X
Character 3-gram	0.68	0.72	0.66	X	X
Character 4-gram	0.66	**0.74**	0.65	X	X

	Dummy Classifier – Most Frequent
Accuracy	24.9%
F1-Score	0.10 (B1)

Table 5.7 Group 3 Results

4-gram SVM	Precision	Recall	F1-Score
Australian	0.80	0.79	0.80
Namibian	0.80	0.89	0.85
Israel	0.71	0.68	0.69
Viennese in Israel	0.65	0.60	0.62
Weighted Average	0.74	0.74	0.74

Table 5.8: Group 3 Classical Results

Word Embedding – Random Forest	Precision	Recall	F1-Score
Australian	0.55	0.60	0.58
Namibian	0.65	0.79	0.72
Israel	0.50	0.47	0.48
Viennese in Israel	0.45	0.34	0.38
Weighted Average	0.54	0.55	0.54

Table 5.9: Group 3 w2v Results

5.4. Group 4 – All Dialects

	TF-IDF - Naïve Bayes	TF-IDF - SVM	TF-IDF – Random Forest	Word Embedding - SVM	Word Embedding – Random Forest
Word	0.67	0.66	0.57	0.48	0.48
Character 1-gram	0.36 (B2)	0.40	0.44	X	X
Character 2-gram	0.56	0.61	0.56	X	X
Character 3-gram	0.63	0.69	0.59	X	X
Character 4-gram	0.62	**0.70 (796.3 s)**	0.60	X	X

	Dummy Classifier – Most Frequent
Accuracy	10.0%
F1-Score	0.02 (B1)

Table 5.10: Group 4 Results

4-gram SVM	Precision	Recall	F1-Score
Australian	0.68	0.74	0.71
Berlin	0.83	0.76	0.79
Namibia	0.50	0.78	0.61
Britain	0.62	0.57	0.59
East Germany	0.73	0.67	0.70
Eastern Territories	0.84	0.79	0.81
Israel	0.53	0.53	0.53
Hamburg	0.91	0.82	0.87
Prussian	0.63	0.63	0.63
Russian	0.99	0.92	0.95
Turkish	0.76	0.68	0.72
Viennese in Israel	0.44	0.48	0.46
Weighted Average	0.71	0.69	0.70

Table 5.11: Group 4 Classical Results

4-gram SVM	Precision	Recall	F1-Score
Australian	0.48	0.33	0.39
Berlin	0.60	0.63	0.61
Namibia	0.43	0.50	0.46
Britain	0.34	0.38	0.36
East Germany	0.44	0.60	0.51
Eastern Territories	0.64	0.66	0.65
Israel	0.40	0.26	0.32
Hamburg	0.64	0.64	0.64
Prussian	0.32	0.38	0.35
Russian	0.93	0.90	0.91
Turkish	0.40	0.44	0.42
Viennese in Israel	0.32	0.11	0.16
Weighted Average	0.49	0.49	0.48

Table 5.12: Group 4 w2v Results

5.5. Times

Classifier:	Average Elapsed time: (seconds)
TF-IDF – Naïve Bayes	7.2
TF-IDF – SVM	544.4
TF-IDF – Random Forest	281.4
Word Embedding - SVM	751.4
Word Embedding – Random Forest	170.4

Table 5.13: Best performance times

5.6. Confusion Matrices

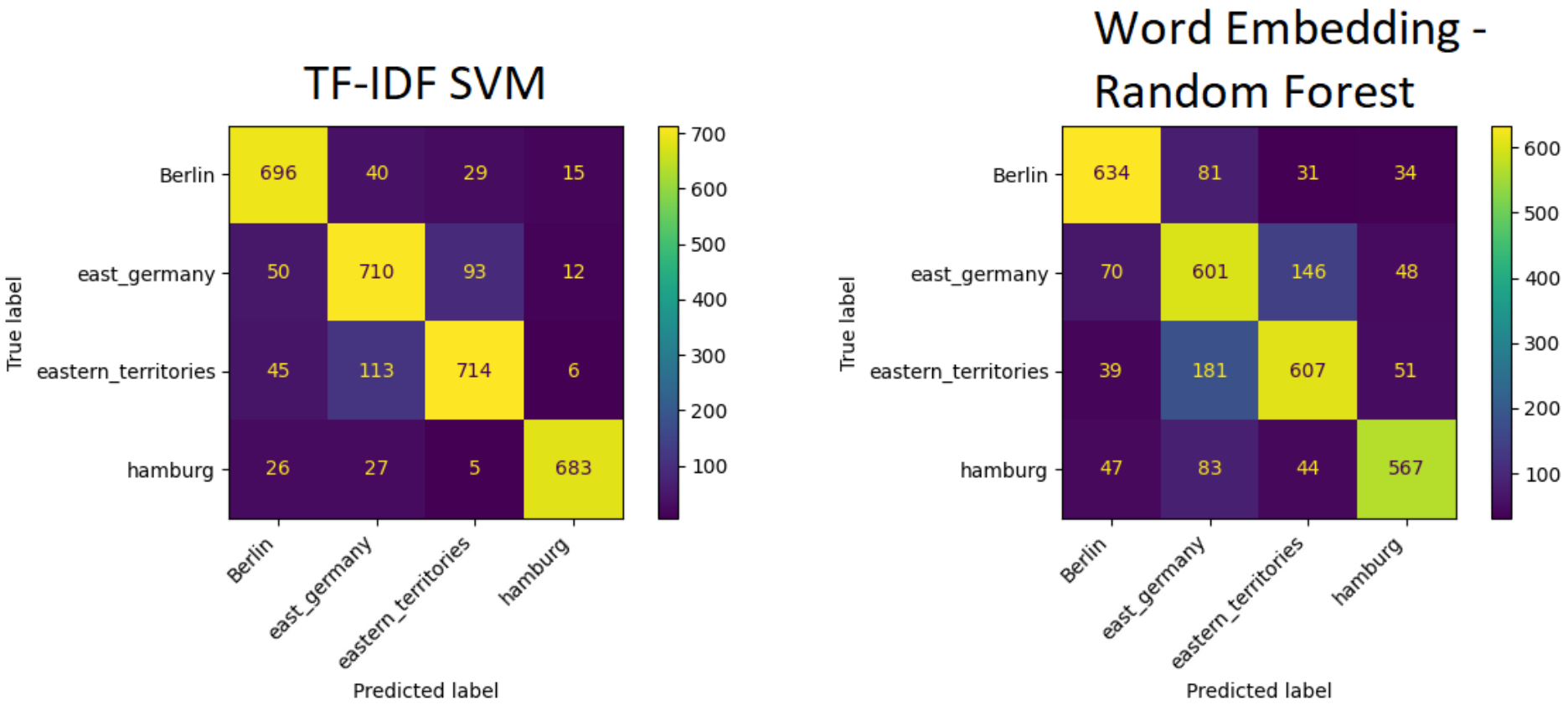


Figure 5.1: Group 1 Confusion Matrices

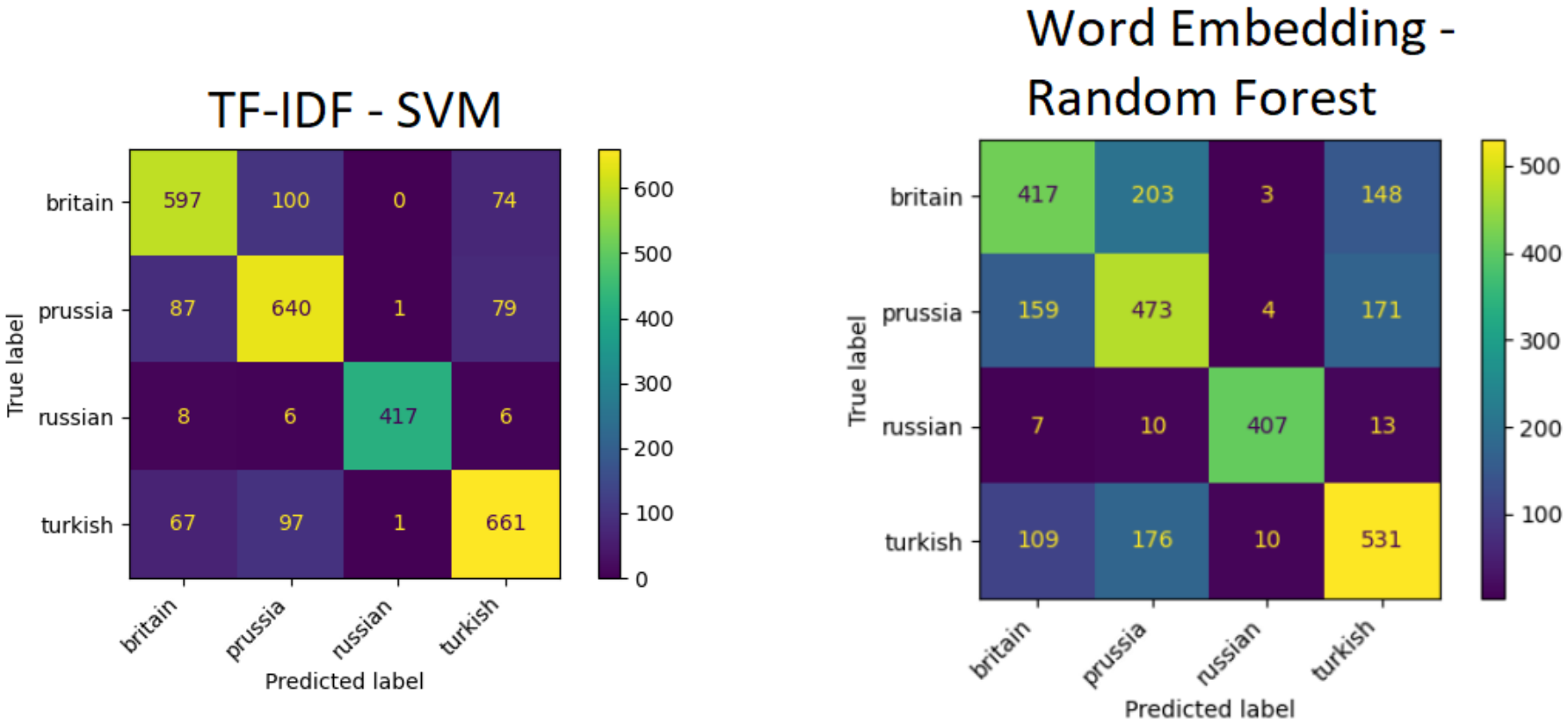


Figure 5.2: Group 2 Confusion Matrices

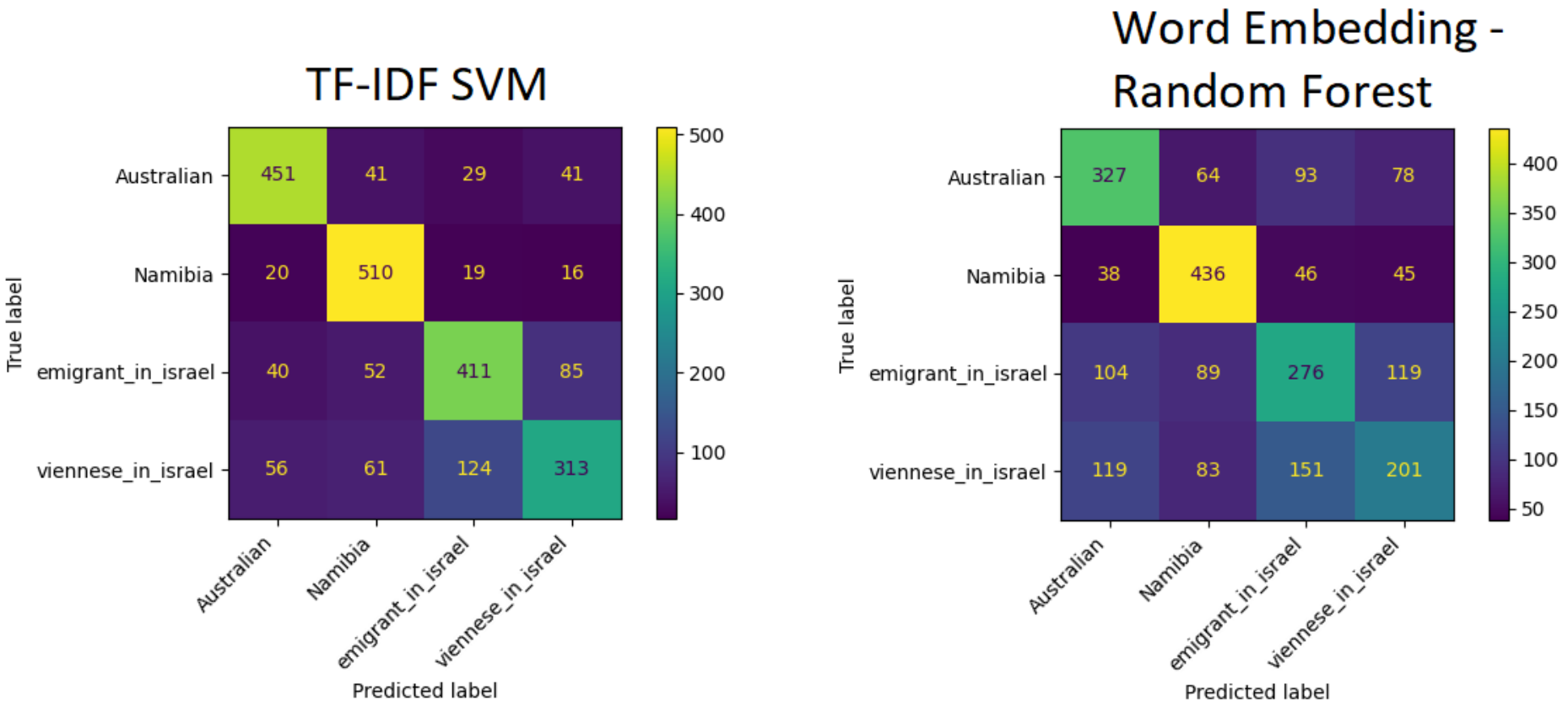


Figure 5.3: Group 3 Confusion Matrices

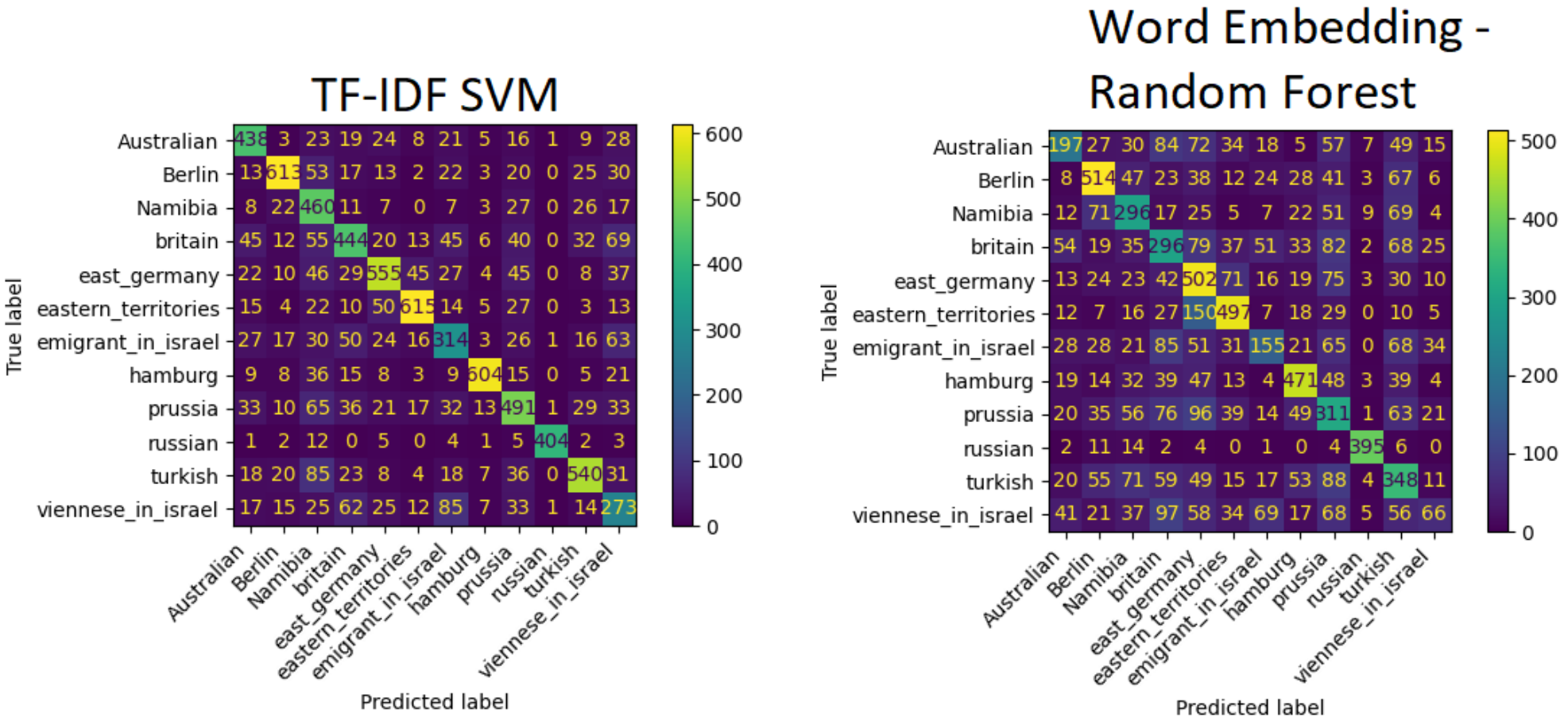


Figure 5.4: Group 4 Confusion Matrices

5.7. Vocabulary Heatmap

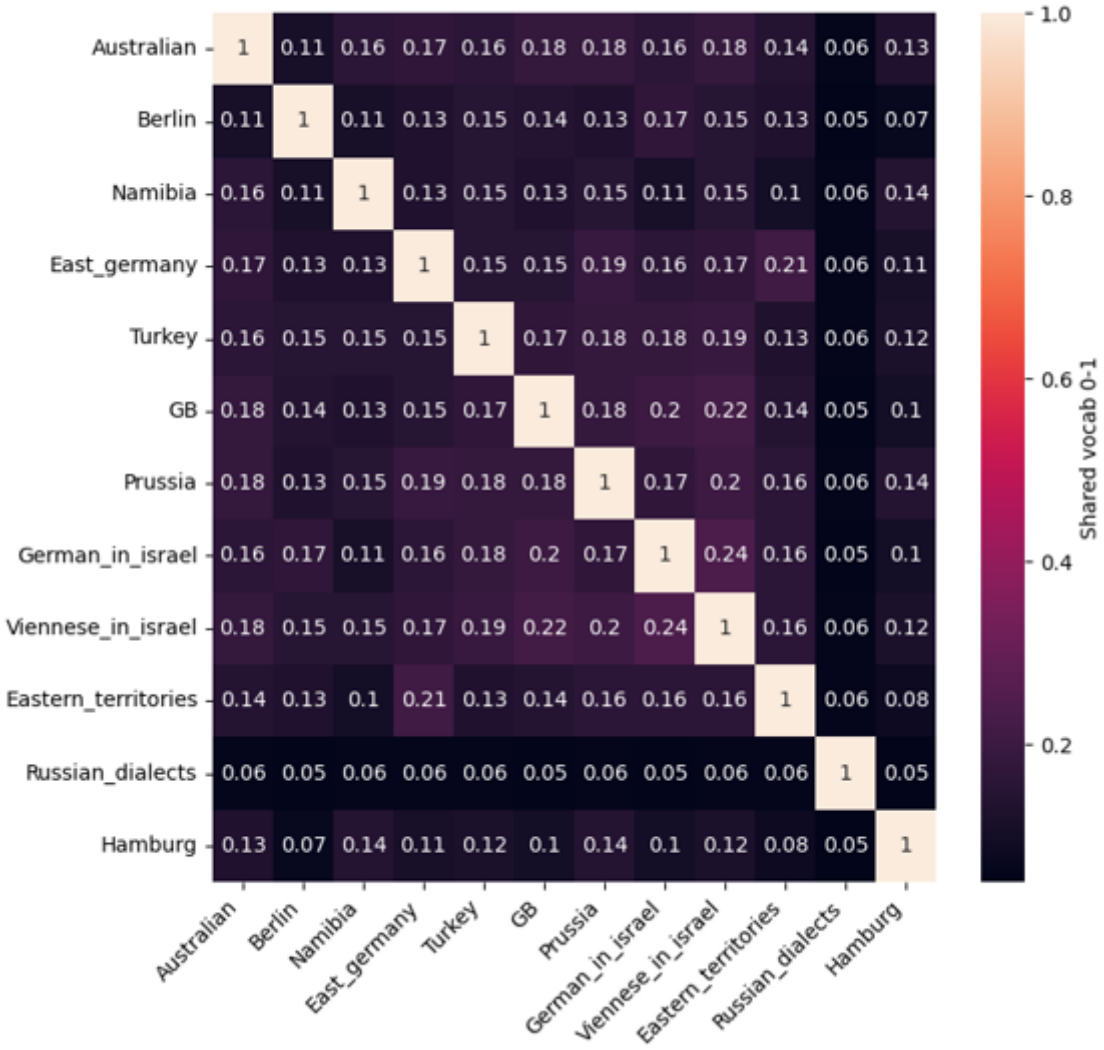


Figure 5.5: Shared Vocabulary Heatmap

6. Discussion

In this section, I will analyse and discuss the results gathered in section 4, in the context of both dialectal similarities/differences, and a comparison of the different models, I will then compare the results from this project to those in related research.

6.1. Dialectal Findings

6.1.1. Confusion Matrix analysis:

Considering group 1, which contains the dialects originating from modern or historical Germany, the large bulk of the inaccuracy stems from the model confusing the East German (GDR) class for the Eastern Territories class, and vice versa. This is largely expected, considering the close proximity of the locations, as well as the fact that both of the data from these two corpora originates from the 1960s, meaning that the language at the time is likely to be similar. To a slightly lesser extent, there is also some confusion between the Eastern territories/East Germany classes, and the Berlin class, a likely reason for this may be that the Berlin corpus contains data from both West and East Berlin, meaning that there is overlap between East Berlin, which was part of the GDR. This was further shown by the overall precision and recall, and by extension the F1-score of the East German class being consistently lower than the other classes in the group, this applied to both the classical and word embedding models, suggesting real similarities between the East German dialect and the other two mentioned dialects in this group.

With regards to group 2, the European group, there appears to be a much larger amount of confusion between most of the classes. Particularly, there was a lot of confusion between the Prussian and British dialects, at first this may seem unexpected, however, upon further inspection, it makes sense considering the context of both corpora, which mostly consisted of German speaking Jews who escaped from Germany and East Prussia after, or during World War II. The Turkish dialect also had some confusion between the British and Prussian dialects, however, nothing that was exceptional. More interestingly, there was extremely minimal confusion between the Russian dialect and the other dialects in the group, even using the word embedding model, which generally had much higher levels of confusion between the other three dialects, obtaining an F1-score of 0.97 and 0.94 on the classic and word embedding models respectively. This suggests that there are very stark linguistic differences between the Russian dialect and the others in the group. Upon inspecting the contents of the Russian corpus, it is clear that many of the words used in the corpus are very different in spelling and structure, with clear Russian influence, for example the use of the word ‘njet’, instead of the standard German ‘nein’, meaning ‘no’.

Group 3 was the non-European dialect group, which contains German spoken all around the world, as such, I expected there to be little confusion between the dialects which originate from very distant locations. The results generally confirmed this, considering the high level of confusion between the Viennese in Israel and German in Israel classes, and significantly lower levels of confusion between these and the other two classes. As a result, the precision and recall of the two Israeli dialects was quite low, compared to the average ~0.8 from the other two dialects. Naturally, this dragged the overall F1-score down to a slightly lower average than the other 4-dialect groups.

Testing the models on group 4 allowed me to get a broader picture of how similar each of the dialects are in the aggregate by observing the confusion between each one, instead of having them grouped into the smaller subsets based on location, although difficult to discuss every dialectal combination, there were some results which stood out. As such, many of the observed patterns were similar – The two Israeli classes produced a lot of confusion between

them, and the Russian dialect produced substantially less confusion when compared to the rest, meaning that the conclusion about this dialect in the Group 2 discussion can be extended to all the other dialects in the dataset.

Interestingly, there was a quite a significant amount of confusion between the Namibian dialect and some of the others, particularly the Turkish dialect, and to a lesser extent the British, and Prussian dialects, however, this confusion was quite one sided, as the classical model seemed to predict many of the data points from these dialects as Namibian, but not the other way around. In addition, these observations were generally not replicated by the word embedding model, with the exception of the Turkish dialect. This makes it difficult to conclude whether there are actual linguistic similarities between these dialects, or if it is just the model trying to optimise its results. However, the prevalence of the confusion between Turkish and Namibian in both models, and the completely different contexts in which the data for each corpus was gathered does suggest that there may be some common linguistic features, which may have developed independently of one another, considering the geographical distance between the two countries.

6.1.2. Vocabulary Similarity Analysis:

Figure 5.5 shows the proportion of shared vocabulary between each of the classes, as expected, many of the same common dialects as mentioned in the previous section share a lot of vocabulary, such as the two Israeli dialects, as well as the East German and Eastern territories dialect, which both had more than 20% of the same vocabulary, interestingly, there seems to be a high vocabulary overlap of 22% between the Viennese in Israel and the British classes, and 20% with the German in Israel class, this similarity was not proportionally represented in the group 4 confusion matrix.

In addition to this, the table shows the very low level of common vocabulary between the Russian dialect and all the others, which confirms the conclusion laid out in the previous section. The same, albeit to a lesser extent applies to the Hamburg class, however, this is likely a result of the contents of its corpus being rather dissimilar to the others.

The rest of the classes generally shared a similar range of ~14-20% of their vocabulary.

In order to explore the dialectical similarity in comparison to the distance of its originating location, I created figure 6.1, which is a plot of the common vocabulary of each class, compared to their beeline distance in kilometres, for countries which do not exist anymore, (i.e., the GDR, Eastern territories, East Prussia), I used the capital cities of where they used to lie.

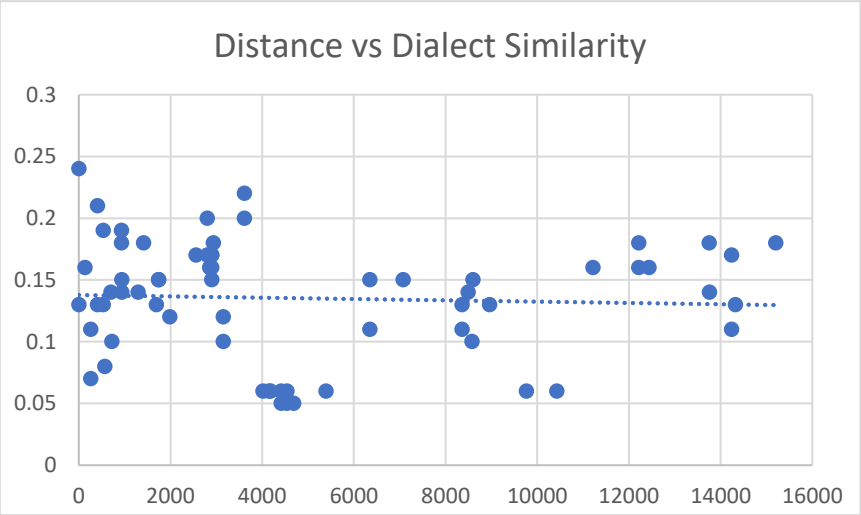


Figure 6.1: Distance v Dialect similarity – All dialects

As shown, there does not appear to be a correlation between the two variables in the aggregate scale, a strongly negative gradient of the best fit line would imply dialects originating from nearby locations are more similar to dialects from distant locations, however, the observed line of best fit has a very slightly negative gradient. There could be many possible contributing factors for this, firstly, consider Australia and Great Britain as an example, although these countries are extremely far away, they are both likely to have strong English influence, this is backed up by their relatively high vocabulary similarity value of 0.18, which skews the data. Secondly, many of the speakers in the corpora immigrated to faraway countries, even though they originate from nearby locations, which is likely to largely influence their dialect. Lastly, there are some outliers, particularly the dialects from Germany, which have quite low similarity, such as the Hamburg class, which probably stems from the content of this corpus being quite different from the others. Omitting the Hamburg class pairs leads to a greater negative trend:

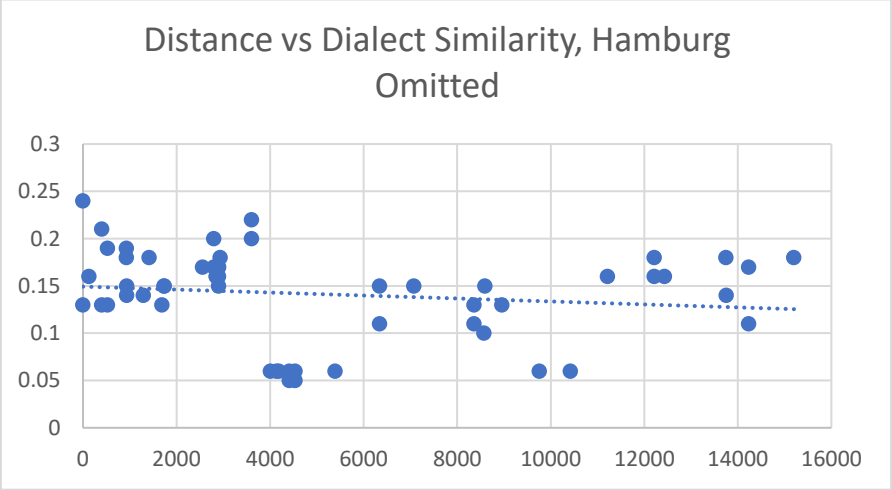


Figure 6.2: Distance v Dialect similarity – Hamburg Omitted

Further, there are some clear examples throughout which do suggest the similarity of dialects from nearby locations, such as the two Israeli dialects, and the Eastern territories & East Germany dialects, however there are also counter examples, such as the Russian dialect which does not seem to be similar to any others, regardless of distance. As such, without more data, it would be difficult to conclude that there is any significant relationship between the similarity of German dialects, and the location from which they originate, since there are too many exceptions which counter this trend all around the world.

6.2. Models

As seen throughout the tables 5.1, 5.4, 5.7, and 5.10, the TF-IDF SVM model seemed to produce the most accurate results, with the most accurate results stemming from either character 3-grams, in the case of group 1 and group 2, or 4-grams, in the case of group 3 and group 4. In almost all cases, the character unigram model performed the worst, with one exception in group 3, in which the Random Forest unigram model slightly outperformed both word embedding models.

When comparing the classical TF-IDF models to the word embedding models, the classical models very consistently did better than the word embedding models, with an average difference in F1-score of 0.18 across all four dialect groups, compared by the highest F1-score from each model type.

As expected, regardless of the differences between each model, every one of the models easily outperformed my first baseline of the most frequent class dummy classifier, with the smallest difference between any of the models and the classifier being 0.38 using the character unigram SVM model on all dialects. Similarly, they all outperformed my second baseline of the Naïve bayes character unigram model, however not by as large a margin, with the smallest difference between it and the models being just 0.01, against the word embedding SVM model in group 3.

Interestingly, the SVM algorithm always performed the best on the classical models, whereas the random forest algorithm outperformed SVM in almost every test when used on the word embeddings. Although, the difference between the SVM-Random Forest performance is much greater on the classical models compared to the word embedding models. This provides interesting insight into how effective the respective algorithms are at classifying the different features extracted from a TF-IDF matrix and a Word2Vec matrix, at least in the context of dialect detection as is done in this project.

A rather unexpected result is the very high accuracy of the models when tested on group 4, originally, I created this group largely just to see where the majority of dialect confusion lies, as such I expected a very poor result due to the large number of additional classes in this group, however, there was a minimal degradation of performance, disproportionate to the number of additional classes. Especially when comparing the word embedding models, the difference in performance between the SVM algorithm from group 3 and group 4 is only 0.01, although this is likely a matter of proportions, since group 3 contains two of the classes with the highest level of confusion between them. In addition to the inclusion of the low confusion classes (i.e., Hamburg and Russian), as discussed in section 6.1, which drag up the average of the overall F1-score.

Comparing the production times of each model on group 4, as shown in table 7.1, the SVM algorithm was by far the slowest on both the classical and the word embedding model, but significantly longer on word embeddings, followed by Random Forest, and finally Naïve

Bayes, which was much faster than the rest, below is a table comparing the times and F1-scores:

Model:	Time (Seconds):	F1-Score	F1-Score/Time (Normalised) (Higher is better)
TF-IDF Naïve Bayes	7.2	0.67	93
TF-IDF SVM	544.4	0.7	1.3
TF-IDF Random Forest	281.4	0.6	2.1
Word Embedding – SVM	751.4	0.48	0.64
Word Embedding - Random Forest	170.4	0.48	2.8

Figure 7.1: Time vs F1-Score 1

As shown above, the performance to training time ratio of the Naïve Bayes model is a whole magnitude better than the next best, which is the random forest word embedding model, and it even performed better than the TF-IDF random forest model. Naturally, this is only important in certain use cases where timeliness of model training is important since models can be saved and reused once trained. However, this would also scale to larger datasets, where using the slower models could be very expensive, in comparison to the marginal increase of accuracy.

6.3. Comparison to Related Work

As mentioned before, there is a very limited amount of literature on German dialect detection, particularly at the scale of this project, making comparison relatively difficult. The paper by Malmasi and Zampieri (2017) does this on five different dialects of Swiss German, achieving an F1 score of 0.66, which is reasonably lower than most of the classifiers used in this project. However, I would suggest that this disparity is likely a result of the dataset which the models are being trained on, since five dialects all within a small country like Switzerland are unlikely to have as much variability as dialects from all over the world.

In comparison to the model efficacy for dialect detection, the results from this project generally line up with similar research, such as the competition which the paper mentioned above took part in, the winning model with the highest accuracy was a meta-classifier using classical methods with different N-gram combinations, and SVM as their classifiers. Similarly, the paper by El-Haj (2020) showed the highest performance on multiclass dialect detection using classical classification methods, although the Naïve Bayes classifier slightly outperformed the SVM classifier in this case. As such, this project confirms a lot of previous literature of classical models generally outperforming word embedding models in multiclass dialect detection.

7. Conclusion

7.1. Project Objectives

7.1.1. Models

In this project, I set out to create a range of machine learning models to detect different dialects of German and comparing how well they perform based on different metrics, such as accuracy and timeliness.

To do this, I created a few classical models using a classical TF-IDF measure in combination with a range of different classification algorithms. To compare, I created a model which uses a word embedding technique, with similar classification algorithms. I found that the classical TF-IDF models generally performed better, with the best of each type obtaining a very good weighted F1-score of 0.7 on all dialects, compared to a weighted F1-score of 0.48 using the word embedding model. Regardless, both models performed far better than the 10% accuracy/0.02 F1-score of my "Most Frequent class" baseline, and better than my Naïve Bayes baseline which obtained an F1-score of 0.36. These results are encouraging for future work on the problem and suggest that there are recognisable features within these dialects which are useful for classification.

In addition to this, I found that the timeliness of the model was more dependent on the classification algorithm, rather than the feature extraction method, as such, the TF-IDF Naïve Bayes model had by far the best performance to training time ratio. Both SVM models performed quite badly in this aspect, however, did produce the most accurate results as discussed.

7.1.2. Dialect comparison

The other main objective of this project was to investigate the similarities of the dialects in my dataset, and to see how distance between dialect locations affects the similarity.

This was achieved using two methods, firstly, creating multiple groups of dialects to run my models on, and analysing the overall accuracy of these models, as well as the confusion matrices which they produced, working under the assumption that the models will create more confusion between more similar dialects, and less between dissimilar dialects. Secondly, I created a vocabulary heatmap which shows the proportion of common vocabulary in every possible dialect pair combination, which gives a good approximation, and quantitative measure of the general similarity of dialects.

The results from the models and confusion matrices were generally as expected, many of the dialects originating from nearby areas and times, such as the two dialects from Israel, which had more confusion between them than other dialects. However, there were also exceptions which led to some interesting findings, such as the minimal level of confusion between the Russian and other dialects, suggesting the Russian dialect is very distinct from the others.

The results from the vocabulary heat map confirmed most of the findings from the confusion matrices, as a more similar vocabulary seemed to correlate with a higher level of confusion between dialects. With the results from the heat map, I graphed out the similarity value of each dialect pair, against the distance of the location from which they originate. The resulting graph showed a very weak negative correlation between dialect similarity and distance, suggesting that there is a slight negative relationship between these two variables but not nearly as much as I expected, since there were many exceptions where nearby dialects were very dissimilar, and faraway dialects similar.

7.2. Reflection

One the main limitations of the project was the amount of data available from each corpus, meaning that I had to use resampling techniques to balance the data I had. There were some interesting corpora within the DGD database which I considered using, such as one corpus containing German spoken in Belgium, and one in Wisconsin, however on further inspection, they either did not contain enough data to create a reliable model or did not have available transcripts. In the same branch, the contents of some of the corpora which I did use were sometimes too narrow, for example, the Hamburg corpus mostly consisted of transcripts of people solving a map task and recording their use of language, as shown by the relatively low unique word count shown in Table 3.1. This makes it difficult to separate some of the narrow corpora from the subjects about which they are predominantly about, and the actual linguistic features of their dialects, since many of the corpora likely contained a lot of niche language which helped the models classify them, which may have led to some bias in my models' predictions. Perhaps if I were to redo this project, I would re-evaluate the corpora I use for dialect detection using the knowledge that I have in hindsight.

Despite these limitations, when considered as a whole, I think most of the corpora used were suitable for purpose of the project, many of them contained a lot of the same/similar themes which would allow the models to differentiate by dialectical differences. Furthermore, this is one of the only available datasets which contains regional spoken German, especially on such a scale which also contains data from very small German speaking communities abroad, in addition to the fact that German dialects have been observed to be declining since the 1950s as observed by Leopold (1959), which means that the likelihood of these dying dialects being recorded in the future is low. The only other papers which I found that work on German dialect identification used the Swiss German dataset which Malmasi and Zampieri (2017) used, however, the work done on this dataset is already quite vast, and it did not fit my criteria.

If I had more time, I would further experiment by creating different types of models, for example with the use of neural networks, as well as different forms of word embeddings, such as fastText and GloVe to see if they perform better than Word2Vec, as well as looking into further optimising my current models by further experimentation with the parameters, or by creating a custom ensemble of these models to improve their results.

7.3. Further Work

One way in which this project could be built upon is by repeating the experimentation done in this project with a larger and more varied dataset, such as one which contains more dialects, and addresses the issues mentioned in the previous section. One way this could be used for some interesting results is by the use of the Twitter methodology as is described in the paper by Goncalves and Sanchez (2015), however, this corpus of tweets would be unlikely to contain much data on the historical and niche dialects which the dataset used for this project does.

In addition to this, an investigation into how effective different types of models, such as the ones mentioned in section 7.2, would be at classifying these German dialects, as well as how they compare to the results from this project.

Moreover, it would be interesting to investigate the dialectical differences based not only on location, but the point in time from where the data originated and see if there are any interesting relationships between these two factors, however, this would likely require more data.

7.4. Closing Remarks

Overall, the results of this project have been very good. It has met and exceeded the original aims and objectives as set out in the project proposal. I have successfully created and compared multiple machine learning models which can differentiate between all different German dialects in my dataset with accuracies as high as 70%. In addition, I have made some interesting and useful findings regarding the dialects, and how they differ depending on their location.

Working on this project has given me an opportunity to develop a much deeper understanding and appreciation for Natural Language Processing and text classification. It helped me appreciate the difficulties associated with the field, as well as the amount of work and research that there is still to be done.

8. References

Bird, S., Klein, E. & Loper, E., (2009) Natural language processing with Python: analyzing text with the natural language toolkit, "Reilly Media, Inc."

Cavnar, William & Trenkle, John. (2001) N-Gram-Based Text Categorization. Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., (2002) SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357.

Cortes, C. & Vapnik, V., (1995) Support-vector networks. Machine learning, 20(3), pp.273–297.

Datenbank fur Gesprochenes Deutsch. (n.d.) Available at: http://dgd.ids-mannheim.de. Last Accessed: 19/03/2022

Dietterich, T.G., (2000) Ensemble methods in machine learning. In International workshop on multiple classifier systems (pp. 1-15). Springer, Berlin, Heidelberg.

El-Haj, M. (2020) Habibi-a multi dialect multi national Arabic song lyrics corpus.

Elworthy, D. (1999) Language identification with confidence limits. arXiv preprint cs/9907010.

García, S., Luengo, J. and Herrera, F., (2015) Data preprocessing in data mining (Vol. 72, pp. 59-139). Cham, Switzerland: Springer International Publishing.

Gonçalves, Bruno & Sanchez, David. (2015) Learning Spanish dialects through Twitter. 14.

Goutte, C. and Gaussier, E., (2005) March. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In European conference on information retrieval (pp. 345-359). Springer, Berlin, Heidelberg.

Gross, B.P., (2009) The ‘High’ and ‘Low’ of the German Dialect. Hebron Herald, 21st Jan. p. 9.

Harris, C.R., Millman, K.J., Van Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J. and Kern, R., (2020) Array programming with NumPy. Nature, 585(7825), pp.357-362.

Ho, T.K., (1995) Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition. pp. 278–282.

Hunter, J.D., (2007) Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(03), pp.90-95.

Leopold, W.F., (1959) The decline of German dialects. Word, 15(1), pp.130-153.

Liddy, E.D., (2001) Natural language processing.

Malmasi, Shervin & Zampieri, Marcos. (2017) German Dialect Identification in Interview Transcriptions. 10.18653/v1/W17-1220.

McKinney, W., (2010) June. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, No. 1, pp. 51-56).

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., (2011) Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, pp.2825-2830.

Rehurek, R. & Sojka, P., (2011) Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 3(2).

Rish, I., (2001) August. An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).

Rokach, L. and Maimon, O., (2005) Decision trees. In Data mining and knowledge discovery handbook (pp. 165-192). Springer, Boston, MA.

Chan, S., Jahromi, M.H., Benetti, B., Lakhani, A. and Fyshe, A., (2017) September. Ensemble methods for native language identification. In Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications (pp. 217-223). Sun, Y., Wong, A.K. and Kamel, M.S., (2009) Classification of imbalanced data: A review. International journal of pattern recognition and artificial intelligence, 23(04), pp.687-719.

Ting K.M. (2017) Confusion Matrix. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7687-1_50

United States Holocaust Memorial Museum. (n.d.) Refugees. Holocaust Encyclopedia. Available at: https://encyclopedia.ushmm.org/content/en/article/refugees Last Accessed: 19/03/2022

Van Rossum, G. & Drake, F.L., (2009) Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.

Webster, J.J. and Kit, C., (1992) Tokenization as the initial phase in NLP. In COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics.

Appendix A. Project Proposal

Automatic Detection of German Dialects Around the World

Abstract

The goal of this project is to create a program which will be able to recognise different dialects of German when given an unseen text using Natural Language Processing and with this, investigate similarities between German dialects. This project will involve using a training model to identify different dialects of speech in the German language using corpora provided in the AGD (Archiv für Gesprochenes Deutsch/Archive for Spoken German).

1. Introduction

Natural Language Processing can be defined as a theoretically motivated range of computational techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications (Liddy, 2001). It is a powerful tool within artificial intelligence with many uses, such as sentiment analysis, machine translation, and for this project, text classification.

German is a language which has hundreds of different dialects (Gross, 2009), spoken not only in Germany and neighbouring countries, but also less commonly, across multiple continents largely as a result of historical immigration and asylum seeking of German speaking citizens abroad (United States Holocaust Memorial Museum, n.d.).

An investigation into how these less common dialects of German compare to standard German spoken in central Europe and other lesser-known dialects from other parts of the world may prove useful in linguistic research, in addition to this, the project could have uses in location detection, based on the regional dialect which a user may input.

One of the difficulties in dialect detection, unlike language detection, is the relative scarcity of available datasets with distinct dialects of a language which must be used for training the model. A reason for this in the case of German, is that most formal communication is done in standard high German, meaning that most of data available of regional dialects in written form will come from transcribed speech, as is the case in the dataset used for this project – in which spoken German from different parts of the world is transcribed. (See 3.2)

2. Background

Within Natural Language Processing, the bulk of the work related to linguistic text classification is based on distinguishing between languages, some ways which this has been done is using character n-grams (Cavnar and Trenkle, 2001), ensemble methods (Chan et al, 2017) and statistical models (Elworthy, 1999), all of which have been highly effective methods of accurately identifying languages.

A paper which tackles the problem with German dialects is one by Ciobanu et al (2018), In this paper, they use an ensemble of classifiers, using a majority rules method, meaning the final ensemble would predict a certain dialect if most of the classifiers agreed on it. Their model managed to obtain an F1 score (Wood, n.d.) of 0.6203, greatly outperforming their random baseline of 0.2521.

However, in this paper they used a dataset limited to only four different dialects, all of which originated from different regions in Switzerland, although the close proximity of each dialect region brings a level of difficulty to developing an accurate model, it lacks the geographical range and potential for analysis of dialectical differences which my project has, including from regions of lesser known, minority German speakers, such as Namibia (See 3.2).

3. The Proposed Project

3.1 Aims and Objectives

The main aim of this project is to investigate how similar different dialects of the German language are to one another. To get sufficient information about similar or different these dialects are, the following objectives will be involved:

- Building of the model – A model will have to be created which can accurately classify pieces of texts into their respective dialect. This model should be a better predictor than a baseline of the Naïve Bayes Classifier.
- Compare similarities between dialects – Dialects will be grouped (See 3.2) and compared against each other, the program will say how similar dialects are to one another
- Analyse results to find similarities between dialects – Similarity results will be analysed and evaluated in the final write-up.

3.2 Methodology

A dialect detection model will be built using Python. The dataset that the model will test is the Archive for Spoken German (AGD), which contains multiple corpora of transcribed German spoken in distinct parts of the world. The following corpora picked out from the AGD will be used:

- Australian German (Australiendeutch – AD)
- Belgian TV Debates (Belgische TV-Debatten – BETV)
- Berlin Turning Corpus – (Berliner Wendekorpus – BW)
- German in Namibia – (Deutsch in Namibia – DNAM)
- German Dialects: DDR (Deutsche Mundarten: DDR - DR)
- German of Turkey Returnees (Deutsch von Türkeirückkehrern – DTRK)
- Flight and emigration to Great Britain (Flucht und Emigration nach Großbritannien – FEGB)
- Escape Stories from East Prussia (Fluchtgeschichten aus Ostpreußen – FGOP)

- Spoken German in South Africa (Gesprochenes Deutsch im Südlichen Afrika – GDSA)
- Emigrant German in Israel (Emigrantendeutsch in Israel – IS)
- Emigrant German in Israel: Viennese in Jerusalem
- Second generation of German-speaking migrants in Israel (Zweite Generation deutschsprachiger Migranten in Israel – ISZ)
- German Dialects: Former German eastern territories (Deutsche Mundarten: ehemalige deutsche Ostgebiete – OS)
- Russian-German Dialects (Russlanddeutsche Dialekte – RUDI)

These corpora cover a wide range of geographical locations, so they will be grouped for testing, for example, by the corpora within Europe vs outside of Europe, or the dialects from Germany vs the rest of the corpora sourced from Europe.

To ensure the validity of the model, it will be compared to a baseline accuracy of the naïve bayes classifier to show the effectiveness of the model at classifying the texts into their dialects. In addition to this, the model will be tested against another completely unrelated dataset (e.g., a dataset of Spanish dialects) to ensure the accuracy of the model is maintained throughout different texts of a similar nature.

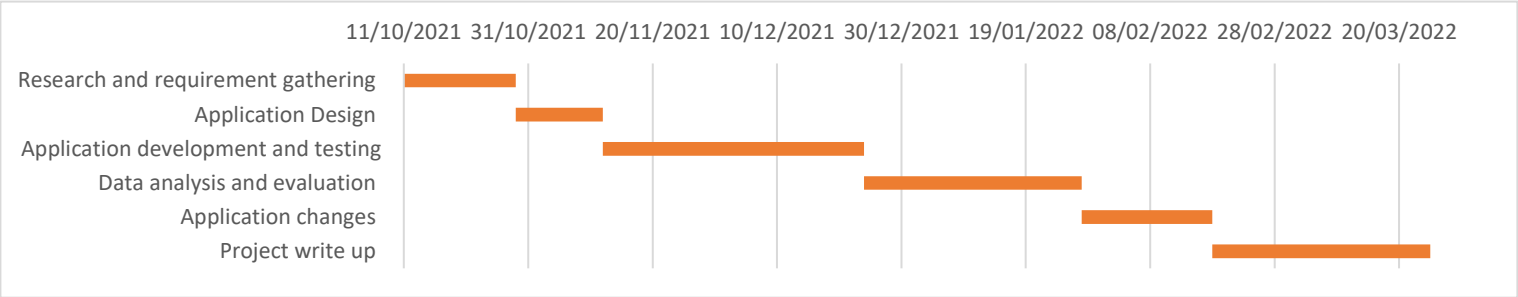
I will analyse and evaluate the data provided by the model to see if there are any interesting or useful findings

4. Programme of work

This project will begin October 2021, and will be running up until late March 2022, it will be split up into the following parts:

- Research and requirement gathering – This involves all the preliminary research with regards to the methodology, including requirements engineering, basic technical research, and finding and getting access to a suitable dataset for the training model, as well as writing of the project proposal. This will take three weeks.
- Application Design – This stage will involve figuring out the details of how the model will be built and designing the software. This will take two weeks.
- Application development – This stage will involve the development of the model following the application design, this includes debugging and testing of the model on our dataset and another. This will take around six weeks.
- Data analysis and evaluation – This stage will involve tabulating my results, and analysing them to find meaningful similarities between dialects, as well as seeing if anything in the model needs to be changed. This will take five weeks.
- Application changes – This stage will involve making any necessary changes to the model, depending on the analysed results from the previous stage. This will take three weeks.
- Project write up – This stage will involve any analysis from the updated model, and documenting all my findings, including all the analysed data and my evaluation of the project as a whole. This will take five weeks.

The schedule of the project is shown by the stages in the Gantt chart below:



5. References

Cavnar, William & Trenkle, John. (2001) N-Gram-Based Text Categorization. Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval.

Chan, S., Jahromi, M.H., Benetti, B., Lakhani, A. and Fyshe, A., (2017), September. Ensemble methods for native language identification. In Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications (pp. 217-223).

Ciobanu, A.M., Malmasi, S. and Dinu, L.P., 2018, August. German dialect identification using classifier ensembles. In Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018) (pp. 288-294).

Datenbank fur Gesprochenes Deutsch. (n.d.) Available at: http://dgd.ids-mannheim.de. Last Accessed: 19/10/2021

Elworthy, D. (1999) Language identification with confidence limits. arXiv preprint cs/9907010.

F-Score Explanation (Thomas Wood, DeepAI): https://deepai.org/machine-learning-glossary-and-terms/f-score. Accessed 25th October 2021.

Gonçalves, Bruno & Sanchez, David. (2015) Learning Spanish dialects through Twitter. 14.

Gross, B.P., (2009) The 'High' and 'Low' of the German Dialect. Hebron Herald, 21st Jan. p. 9.

Liddy, E.D., (2001) Natural language processing.

United States Holocaust Memorial Museum. (n.d.) Refugees. Holocaust Encyclopedia. Available at: https://encyclopedia.ushmm.org/content/en/article/refugees Last Accessed: 20/10/2021