Detecting and analyzing code-switching behavior on a Moroccan-Dutch corpus using transformer architectures

Tom Kalkman (6209890)
Daily advisor: Anna Wegmann (a.m.wegmann@uu.nl)
First advisor: dr. Marijn Schraagen (m.p.schraagen@uu.nl)
Second advisor: dr. Dong Nguyen (d.p.nguyen@uu.nl)
Master Artificial Intelligence, Utrecht University

December 20, 2023

Abstract

The use of linguistic influences from various languages, or the fluid alternation between languages, is known as code-switching. This phenomenon is particularly prevalent in areas shaped by diverse cultural influences, such as young urban communities in the Netherlands. Tools for Natural Language Processing (NLP) have seen an increase in use and performance quality over the last decade, but are typically not trained to work with multilingual, urban youth speech styles. In this thesis, I train models with different levels of complexity with the objective to recognize code-switching. For this objective I use the Moroccorp: an unlabeled Moroccan-Dutch corpus that consists of chat conversations between Moroccan-Dutch internet forum users. I annotate this dataset on word-level, with labels that describe the languages and linguistic varieties that are present, as well as labels for language independent utterances. Two deep learning models are fine-tuned on the annotated dataset. For this, I use two pretrained transformer models and compare their performance to a multinomial logistic regression baseline model on language identification. I use the Dutch model RobBERT and Multilingual BERT, that is trained on over a hundred languages, including Dutch, English and Arabic, which are all present in the Moroccan-Dutch corpus I use. If informal code-switched texts can be processed as well as more formal, monolingual texts by NLP tools like RobBERT, this could reduce performance bias for non-standard Dutch in technological appliances like voice activated tools or hate speech detectors on social media. The annotated subset of the Moroccorp contains about 10% of code-switched sentences, with a Code-mixing index (10,92) that is similar to other datasets used for language identification. The best hyperparameter configurations for both models reach a higher F1-score (F1 = 0,83) than the logistic regression baseline model (F1 = 0,53) on the token classification task. Although differences in F1-score between the two transformers proved to be insignificant (p=0,07), M-BERT shows higher precision, whereas RobBERT shows higher recall and accuracy. Recognizing code-switching and the Moroccan ethnolect simultaneously is shown to be complex for the models. Higher performance may be achieved by focusing on the two tasks individually and by using a more balanced dataset.

Contents

1 Introduction

This chapter introduces many different concepts, which are all tied together by my thesis, as I focus on the recognition and classification of multiple languages in one conversation or sentence, some of which are under-resourced or even difficult to define as a single well-established language. For this classification I use machine learning models for Natural Language Processing (NLP). In Section 1.1 I introduce the phenomenon of code-switching. Then, I introduce the NLP task of language identification in Section 1.2. The linguistic concept of a multi-ethnolect is explored in Section 1.3. Then, I dive into the relevance of working with under-resourced languages in Section 1.4.

1.1 Code-switching

Many people in the world are bilingual. As the group of bilinguals is large and diverse, there is a range of individual differences with regard to their language use, proficiency and alternating between languages [Genesee 2016]. Some bilinguals use each language they speak roughly equally often, and some may be more proficient in one over the other. Some switch languages frequently within one conversation, whereas others use each language only in separate contexts, switching rarely [Mann and de Bruin 2021]. For example, Moroccan immigrants in the Netherlands may speak a Berber dialect with close family members, but speak only Dutch at work [Schmeets and Cornips 2021]. However, in informal situations, like on social media, speakers may alternate more, choosing their language-use based on their communicative goals [Bali et al. 2014].

The phenomenon of using at least two languages in the same utterance, sentence or conversation is referred to as *code-switching*. See (1) for an example of code-switching from English to Spanish from Mellado and Lignos [2022]:

(1) I got it, but *prefiero usar mi Dell para cosas sencillas*.
I got it, but I'd rather use my Dell for simple things.

In (1), the switch happens within the same sentence, but we also speak of code-switching when the switch happens between sentences. There are several ways to categorize different types of code-switching. For example, the location of the switch, or the reason of its occurrence. I will introduce these in Section 2.1.2.

On social media, code-switching occurs frequently: over 10% of the X (formerly Twitter) users tweet in more than one language, but not necessarily within one tweet [Hale 2014], and Rijhwani et al. [2017] estimate that about 3.5% of tweets are code-switched, using more than one language in a single tweet. Although there has been research in the field of NLP on models that are focused on code-switching, like James et al. [2022], Samih et al. [2016], monolingual models are much more common and there is still a lot to explore, especially in low-resource language varieties. Although code-switching is a general phenomenon, I mainly focus on code-switching between Dutch and Moroccan-Dutch in my thesis.

1.2 Language Identification

As many NLP models are processed and analyzed in a language-specific way, it is imperative to identify the language in which a particular segment is written in order to handle code-switched data [Aguilar et al. 2020]. Language Detection, or Language Identification (LID), is the process of identifying the language of a given text or document automatically. In the context of code-switching, the language identification takes places more locally, obtaining language labels on the sentence-level or token- or even subword-level [Mager et al. 2019], depending on the type of code-switching that is present. Inversely, labeling each word with labels that identify the language in which it is written is also a way to detect code-switching, by detecting a change in labels. There are other methods to detect code-switching, like lexical-based approaches, such as a dictionary lookup [Nguyen and Doğruöz 2013] or syntax-based approaches, such as using Part-of-speech tags to quantify the complexity of code-switching [Kodali et al. 2022].

Compared to more complex NLP tasks, such as Question Answering or Natural Language inference, LID is a relatively easy task. Some scholars believe it to be an almost solved problem, especially outside the context of code-switching, at the document-level [McNamee 2005]. However, there are still many factors that make LID a challenging task in short and informal texts, such as chat utterances, that are characterized by misspellings, phonetic typing, word play or uncommon abbreviations [Das and Gambäck 2014, Thara and Poornachandran 2021]. Short texts may not contain enough linguistic features to determine the language, especially in the presence of grammatical errors, obscuring important linguistic patterns. On the other hand, it is possible that the to be identified language is relatively rare, ill-documented or not expected by a classifier and therefore difficult to detect [Adebara et al. 2022]. Other challenges of LID will be addressed in Section 2.3.1.

1.3 Multi-ethnolects

A type of language variety that is characterized by a combination of code-switching and the mentioned challenges of LID, like phonetic typing, grammatical errors and insufficient documentation is the one categorized as a multi-ethnolect [Kossmann 2019, Nortier and Dorleijn 2013]. This is an umbrella term for dialects and speech styles that are mainly used by young people in multi-ethnic urban communities that are rich in cultures and languages, caused by language contact [Quist 2008]. *Language contact* is the phenomenon where two groups of people who speak different languages meet and influence each other's language use [Bell 2013, p.47]. Migration is a prevalent source of language contact, but can have different outcomes regarding impact on language use [Auer 2020].

In urban areas of the Netherlands, the multi-ethnolect that is used is called *Straattaal* (lit.: street language), which is often used by second, third or even fourth generation immigrants with a background in various parts of the world [Appel 1999, Nortier and Dorleijn 2013]. Straattaal was first identified and researched in 1999 and is based in Dutch, although its lexicon draws from multiple languages, including the Surinamese language Sranan Tongo and English [Appel 1999, Nortier and Dorleijn 2013]. See (2) for an example of a sentence in Straattaal from [Kossmann 2019]. Words in italic are non-Dutch insertions, those not in italic are standard Dutch. 'Agga' is simply a distorted version of 'Den Haag', a city in the Netherlands, whereas 'osso' means 'house' in Sranan Tongo.

(2)	Kom	we	gaan	vandaag	naar	*Agga*	of	gaan	we	gelijk	naar	*osso*?
	Come	we	go	today	to	The Hague	or	go	we	straight	to	home?

I will elaborate on the characteristics and terminology of these language varieties in Section 2.2.

1.4 Under-resourced language variants in NLP research

Deep Learning language models have seen a large increase in performance on all common Natural Language Processing (NLP) tasks over the past few years. In November 2023, 22 different deep learning models have surpassed human baselines on the GLUE benchmark[1], which evaluates performance on various NLP tasks [Wang et al. 2019]. Very recently, Chat-GPT, an application of the Large Language Model GPT-3[2] [Brown et al. 2020], received a lot of praise and attention from the general public for being able to generate text about any subject that can be indistinguishable from text written by a human [Dale 2021].

In order to achieve this level of proficiency, Large Language Models (LLMs) are frequently pre-trained on corpora consisting of vast amounts of text, often exceeding 1 billion words, to provide them with statistical patterns of natural language, to predict the most likely word to follow a given input. Once pre-trained, these models may be fine-tuned for specific tasks such as Sentiment Analysis, Machine Translation, or Text Generation through additional training, typically on tasks distinct from those employed in pre-training. Fine-tuning requires a smaller amount of data and less time compared to the pre-training process.

A shortcoming of popular pre-trained language models, like BERT [Devlin et al. 2019] and GPT-3 [Brown et al. 2020] is that they draw mostly from *high-resource languages*, languages for which many sources are available. Moreover, NLP research tends to focus on a

[1] https://gluebenchmark.com/leaderboard/

[2] At the time of writing, Chat-GPT uses GPT-3, but may in the future use newer and more advanced models, like GPT-4, which is at the moment only available for premium members

handful of Indo-European languages, leaving behind other frequently used languages [Hovy and Spruit 2016]. According to a taxonomy of languages in NLP [Joshi et al. 2020], about 88% of the 6,500 languages taken into consideration are currently without representation, due to the absence of (un)labeled datasets, despite being used by over 1.2 billion speakers. However, awareness for this problem has been spread widely over the last few years, leading to an increase in research on linguistic diversity in NLP [Blasi et al. 2021, Levow et al. 2021]. Language diversity was even a special theme at the Association of Computational Linguistics (ACL) 2022[3], to encourage research in low-resource languages, as well as endangered languages to be sustained.

Interestingly, data that is available on social media for under-resourced languages, are often code-mixed [Thara and Poornachandran 2021]. As stated earlier, language models are often monolingual, trained on corpora written in a single language. Multi-language models exist, but are usually trained on monolingual documents taken from various high-resource languages, like Multilingual BERT (M-BERT) [Pires et al. 2019]. Usually single-language models perform better on downstream NLP tasks in that specific language [Martin et al. 2020, De Vries et al. 2019].

1.4.1 Problems of under-researched language variants

While models trained on high-resource languages, like the over-represented English, perform well on those languages, they may encounter problems when the to be processed text data is written in other varieties of the same language, like a dialect, as most English language models are almost exclusively trained and evaluated on Standard British English or Standard American English [Blodgett et al. 2016, Ziems et al. 2022]. For instance, African American Vernacular English is more likely to be misclassified as Non-English [Blodgett and O'Connor 2017] or as Hate Speech [Rios 2020] and is more difficult to identify in the first place [Jurgens et al. 2017]. The reasons behind this are complex and not entirely explainable from underrepresentation alone, but increasing inclusivity of dialects and language varieties is a step towards a reduction in inequality of NLP model performance for these dialects compared to their standard counterparts [Groenwold et al. 2020].

In the field of psychology, research tends to be carried out on samples drawn entirely from Western, Educated, Industrialized, Rich and Democratic (WEIRD) societies [Henrich et al. 2010], creating results that only apply to that demographic group. Similarly, if NLP models are only trained on language used by one demographic group, they will naturally perform worse on data falling out of the scope of the training data. Depending on the training data, a model would perform worse on certain demographic groups that are not represented in the training data at hand, like young people [Hovy and Søgaard 2015] or ethnic minorities [Jørgensen et al. 2015]. This can lead to exclusion or demographic misrepresentation, especially once

[3] https://www.2022.aclweb.org/post/acl-2022-theme-track-language-diversity-from-low-resource-to-endangered-languages

a model is applied in consumer facing technology, such as Google Translate, or a chatbot helpdesk. This reinforces demographic differences by making technology less useful for already disadvantaged groups [Hovy and Spruit 2016].

One way to mitigate the issue of linguistic bias, is to train models on balanced datasets that are diverse and representative of a broad demographic and reflects reality as well as possible [Dixon et al. 2018]. An approach to realize such a diverse dataset is to collect several datasets from multiple sources, including different demographic groups and geographical locations. Realizing such a diverse dataset would be a task too big for the scope of this thesis. Instead I will be focusing on one isolated low-resource language variety, similar to Papalexakis et al. [2014], who predict code-switching in a large online discussion forum for the Turkish-Dutch immigrant community in the Netherlands.

1.5 Research Goals and Chapter Overview

Both code-switching and the use of multi-ethnolects are phenomena that occur in modern society, specifically in immigrant communities in the Netherlands. Multilingual language varieties, especially multi-ethnolects such as Straattaal, are underrepresented in NLP research. One of the first steps needed to take in multilingual NLP is Language Identification (LID).

In Chapter 2, I have done a literature review on the phenomenon of code-switching, the Moroccan-Dutch ethnolect as well as on other works of language detection datasets and report on methods to measure code-switching. I conducted analysis on the annotated dataset using these methods, to examine if code-switching occurred in the Moroccorp, and how much.

I present the following two research deliverables:

I present an **annotated dataset**[4], suitable for a word-level language identification task that focuses on the identification of code-switching as well as the use of the Moroccan-Dutch ethnolect in the form of non-standard Dutch, like spelling variations and grammatical errors and the usage of wrong words. The words are tagged with labels that do not exclusively represent languages, but also include language varieties, like the Moroccan-Dutch ethnolect. I use the Moroccorp, a dataset composed by Ruette and Van de Velde [2013]. The corpus consists of informal chat messages in which the conversations are predominantly written in Dutch, but also contain messages with elements of other languages, like Arabic and English. I annotate the data on word-level. Each word receives a label that identifies the language or language variety in which it is written, or which category of language independent tokens it is. My approach for the annotation of the dataset is discussed in Chapter 3. The analysis to measure the amount of code-switching in the Moroccorp is found there as well.

Moreover, I have fine-tuned **two deep learning language models** that can perform a custom Language Identification task on the annotated dataset in the form of Multiclass Token Classification, as well as a baseline model. I fine-tune two pre-trained BERT-models on

[4] Dataset can be found here: https://huggingface.co/datasets/Tommert25/extradata0908/tree/main

Language Identification: RobBERT[5] [Delobelle et al. 2020], a Dutch language model, and Multilingual BERT (M-BERT)[6] [Pires et al. 2019]. I fine-tune the models on the annotated Moroccorp. I evaluate the models on accuracy, precision and F1-score. I searched for the highest scoring model on a validation set by performing hyperparameter tuning. Then I run the best performing model for five different random seeds, and perform error analysis on the highest scoring model. I compare their performance to a logistic regression baseline model.

I found that transformer models do relatively well on the task, better than logistic regression (F1-score = 0,528), but between RobBERT (F1-score = 0,817) and M-BERT (F1-score = 0,812) remains little difference (0,01 on macro-average F1-score).

In Chapter 4, I discuss the fine-tuning of the two transformer models, as well as training the baseline logistic regression model. After hyperparameter tuning and choosing the best models for each architecture, I analyze what the models can do in Chapter 5. Finally, limitations of my research as well as ideas for future work are discussed in this chapter.

[5] The fine-tuned RobBERT model can be found here: https://huggingface.co/Tommert25/RobBERTBestModelOct13
[6] The fine-tuned M-BERT model can be found here: https://huggingface.co/Tommert25/MultiBERTBestModelOct13

2 Theoretical Framework and Related Work

My thesis touches upon diverse topics spanning various research fields, which have each been both researched individually, as well as in combination with each other. To provide context on each of the topics I introduced in the previous chapter, I conducted a literature review, which has resulted in the following sections: In Section 2.1, I discuss linguistic research on code-switching. In Section 2.2 I elaborate on what characterizes the Moroccan-Dutch ethnolect. Then, in Section 2.3 I discuss previous NLP research on multilingual datasets and language detection. A general explanation of Transformer models and specifically the models RobBERT and M-BERT follows in Section 2.4.

2.1 Code-switching from a linguistic perspective

Since the 1980s, code-switching - using multiple languages in an utterance, sentence or conversation - has been recognized as a part of multilingual language use and is researched mainly in the context of sociolinguistics [Das and Gambäck 2014]. The phenomenon of code-switching is often looked down upon, both from inside as well as outside of the communities that participate in it, and is sometimes considered a corrupted semi-language [Bell 2013, p.114]. A misconception of code-switching may be that a speaker resorts to using another language because they do not master the first one well enough. However, code-switching shows predictable structures, such as preservation of grammatical structure of each language involved and speakers who use code-switching are usually proficient bilinguals [Myers-Scotton 1997].

In a code-switched utterance or conversation, one language is usually dominant over the other. This dominant language is often referred to as the *matrix language* [Myers-Scotton 1995], which creates the skeleton for each sentence, predominantly using that language's grammatical structure and word order. The *embedded language* is Myers-Scotton's term for the other language, which is the less dominant language, often only used for content material, like vocabulary insertions.

(3)	‘Leo	si	ku-	*come*	na	*books*	zangu.’
	today	1s/neg	past/neg	come	with	books	my

‘Today I did not come with my books.’

In (3), Swahili is the matrix language, and English is the embedded language. The switches are made both from Swahili to English and back to Swahili, amounting to four code-switches in total. The grammatical structure is provided by Swahili, and is therefore the matrix language. The use of the terms matrix language and embedded language has been spread widely throughout code-switching research [Das and Gambäck 2014, Khanuja et al. 2020, Solorio et al. 2014], but has also received criticism. For instance, the matrix language and the embedded language can switch places from sentence to sentence. Although it is uncommon for texts that use code-switching to lack a clear matrix language, it is theoretically possible, especially if the text is short. In both examples (1) and (4), the sentence starts in English, but finishes in Spanish and Turkish, respectively. As neither one looks more dominant than the other, contextual clues are necessary to determine which language is the matrix language.

2.1.1 Types of code-switching

There are several ways to define categories of code-switching, based on *where* the switch occurs, *why* a switch happens and *how* it relates to other multilingual phenomena, like lexical borrowing, which I discuss in Section 2.1.2. For the location of the switch, linguists generally agree on three types of switches, originally identified by linguist Shana Poplack [Poplack 1980]. The first two are intersentential and intrasentential switching. Respectively, switches that happen at sentence boundaries like in (4), in which a switch happens from Dutch to Turkish [Nguyen and Doğruöz 2013], and switches that happen within a sentence, like in (5) [Aguilar et al. 2020]. Code-switching within the same sentence is also referred to as 'code-mixing'. A third type, shown in (6) [Esenjul 2022] is called tag switching, where the switch occurs in a sentence tag like *'you know?'* in English [Poplack 1980]. As a tag occurs usually within the same sentence, it can be argued that this is a subset of intrasentential switching. An overview of the categories is given in Figure 2.1.

(4) Mijn dag kan niet stuk. *Cok guzel bir haber aldim.*
My day cannot go wrong. I received good news

(5) *LREC será* hosted in Marseille.
LREC will be hosted in Marseille.

(6) Él es de Oaxaca y así los criaron a ellos, *if you know what I mean*.
He is from Oaxaca and that's how they were raised if you know what I mean.

(7) *oet*verkocht
sold out

Mcarthur [1992] identified a fourth type, intra-word switching, in which a change occurs within a word boundary (7) [Nguyen and Cornips 2016]. Some scholars use different definitions of inter- and intrasentential code-switching, including switches at a clause boundary within the same sentence as intersentential [Das and Gambäck 2014, Lynn and Scannell

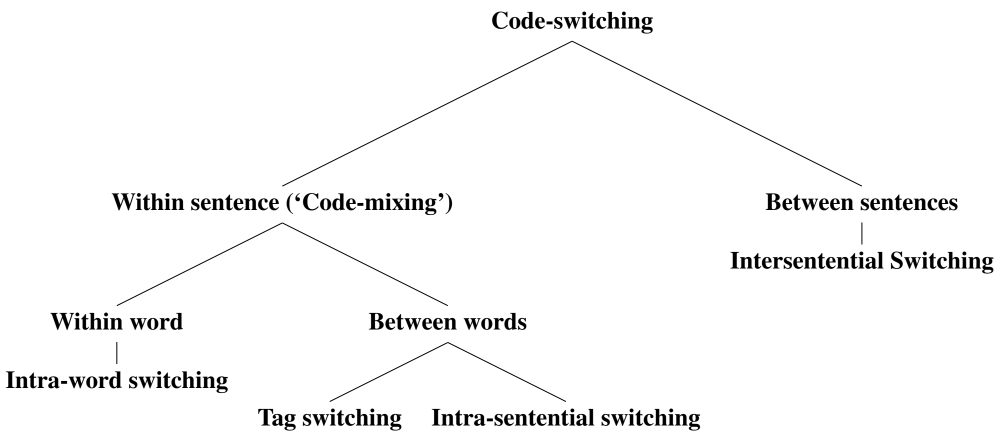


Figure 2.1 Schematic overview of types of code-switching based on location. On a high-level split into two types, with three subtypes of switching within a sentence.

2019]. Although including clause boundaries can make the distinction between intra- and intersentential code-switching ambiguous, most scholars agree on the sentence boundary for intersentential switching.

There are several theories on why a code-switch happens. According to research on Chinese-English code-mixing in highly bilingual societies in Hong Kong [Li 2000] and Macao [San 2009], the primary reasons for switching to English were found to be linguistic in nature, rather than social, for instance, the absence of a certain expression in Chinese and therefore needing to switch to English. However, this is inconsistent with studies on different language pairs [Bock 2013, Negrón 2009], which suggest that code-mixing is often used at the start of a message, or through simple insertions, mainly to indicate in-group membership. Examples of moments in a conversation in which code-switching is more likely to occur are: Introducing a quotation, which is usually an intrasentential switch, like in (8); picking out a specific addressee, which is often intersentential, as it is in (9); an interjection, which is almost always the same as a tag switch (6) [Gumperz 1977]. The following examples are from Gumperz [1977]:

(8) I went to Agra, *to maine apne bhaiko bola ki*, if you come to Delhi you must buy me some lunch.
I went to Agra, then I said to my brother that, if you come to Delhi, you must buy me some lunch.

(9) B (to A in Slovenian): je ki tako nasičen z jabolka pa je že čist' stum, je pa stran.
it is so overloaded with apples and the entire tree is bent already

B (in German, turning to C sitting apart): Regen werd, so ein Wind ist draußen
It will rain, it is so windy outside.

Gumperz also found that some switches are situational, when the association between a social situation and language causes the speaker to switch to a certain language, lasting as long as the situation does. For instance, researchers observed in Norway that the entry of outsiders caused a local group to switch from a local dialect, Ranamål, to standard, Bokmål [Gumperz 1977]. Changes in conversational topics, such as a switch from business to personal topics also triggered a switch from standard speech to dialect.

Also, it is not always clear how each segment of a word that carries meaning, or *morpheme*, should be classified. (10) is an example from Maharjan et al. [2015], in which the word 'snapchateame' could be considered code-switching within the word, as 'snapchat' is an English term, mixed with a Spanish verb conjugation 'chateame', which means 'chat me'. However, one could also argue that the verb 'snapchateame' is not code-switching at all, counting 'snapchat' as a single borrowed word, which was then conjugated in Spanish.

(10) Ayy que pepe *snapchateame* el arreglo
Ay, what a 'Pepe', snapchat me the arrangement

2.1.2 Code-switching compared to lexical borrowing

Code-switching overlaps quite a lot with *lexical borrowing*, incorporating a word or lexical unit from one language into another language [Haugen 1950], but it is not equivalent to code-switching. Most words that are borrowed from other languages, are well-established loanwords in the matrix language and are often adapted to fit the sound system and grammar of that language [Poplack and Dion 2012]. (11) is an example of a single lexical borrowing from Mellado and Lignos [2022]:

(11) Intentando comprar *online* uno de los nuevos discos duros que saco Samsung, pero qué lata tener que rellenar tanto formulario
Trying to buy one of the new hard disks that Samsung released *online*, but what a nuisance to have to fill in so many forms

In order to borrow a single lexical item from a second language while retaining the grammatical structure from the first one, one does not necessarily need knowledge of that other language. Typically, linguists have distinguished code-switching from borrowing, because for code-switching speakers must deliberately draw from their knowledge of both languages to produce a word or phrase from either language [Jose et al. 2020, Poplack and Dion 2012]. However, speakers do not always do this in the same way, and even relatively well-established code-switched languages like Spanglish (Code-switched Spanish and English) or Hinglish (code-switched Hindi and English) are very dynamic [Sitaram et al. 2019].

As there does not seem to be a consensus on a clear-cut definition of code-switching, it remains hard to concretely distinguish code-switching from lexical borrowing. However, Mellado and Lignos [2022] set guidelines as to what they consider a lexical borrowing, and therefore not a code-switch. They investigated Spanish-English code-switching, with the Spanish language as the matrix language, and elements of the English language embedded in the speech. According to them, the following types of words are considered borrowings and should not - by themselves - be categorized as code-switching:

- Technology words and words related to Twitter terminology[1], like *follower*, *blog*, *computer* or *online*.
- English words that are already registered in the official Spanish language dictionaries or have an entry in Spanish Wikipedia.
- Words that originate from the English language, but are used following Spanish grammatical structure, like an English adjective following a Spanish noun.

Whether the last type is borrowing or code-switching is defined by the usage of the grammatical structures of both languages involved. When Spanish grammar does not need to make way for English grammar, it is regarded as a lexical borrowing.

2.2 The Moroccan-Dutch Ethnolect

This sections discusses the Moroccan-Dutch ethnolect. First, I provide context on the origin of the prevalence of Moroccans in the Netherlands in Section 2.2.1. Then, in Section 2.2.2 I define which terms I'll be using throughout the thesis for the different linguistic varieties I discuss. Section 2.2.3 covers what the origins are of the Moroccan-Dutch ethnolect, and I discuss what characterizes it as well as its differences from Straattaal in Section 2.2.4. Lastly, I introduce the dataset I use for annotation and language identification in Section 2.2.5.

2.2.1 Recent history of migration and language contact in the Netherlands

As briefly discussed in Chapter 1, migration influences the development of new language forms. In recent history, migration started to increase in the Netherlands after the wars of the 20th century. A prevalent reason for migration to the Netherlands was decolonization: after World War II, many people from the former Dutch colony Indonesia migrated to the Netherlands. In the years right before and after 1975, when Suriname gained their independence from the Netherlands, a mass migration took place as well [Schumacher 1987]. Along with the Antillians, which are still a part of the Kingdom of the Netherlands today, the Surinamese are the most recent examples of migration from former Dutch colonies and special municipalities as well as the most influential. In contrast to Indonesian Dutch, which

[1] Mellado and Lignos [2022] specifically work with data from X (formerly Twitter), which is why these are singled out. This could be replaced with any foreign domain-specific terminology, depending on the context.

Country of Origin	First Generation	Second Generation	Share of General Population
Türkiye	215.073	228.619	2,49%
Morocco	175.207	249.706	2,38%
Suriname	176.213	185.206	2,02%
Indonesia	89.917	256.447	1,94%
Germany	110.806	228.539	1,90%
Poland	182.613	50.045	1,31%

Table 2.1 Top 6 origins of citizens with migration background [CBS 2023]

is used less over time [De Vries 2005], the Curaçao and Suriname ethnolects of Dutch are commonly used, and their influence is reflected in the linguistic variety *Straattaal* (street language) [Kossmann 2019].

Later, reasons for migration were usually non-colonial and not unique to the Netherlands. The 1960s and 1970s faced a shortage of skilled workers, especially in food, mining, steel and textile sectors, as a consequence of a surge in education and prosperity in all of North-Western Europe. These workers migrated from Italy, Spain, Morocco and Türkiye, stimulated by the Dutch government to work in the Netherlands, since they could do the same work as the Dutch, but cheaper [Bardaï 2003]. After the first oil crisis, this stream of immigration ended, and most workers, especially those from Spain and Italy, returned to their countries of origin. The same applies to Turkish and Moroccan migrants. Respectively 85% and 70% of the migrants from these countries were no longer living in the Netherlands by 2003 [Bonjour 2009]. Others invited their families to the Netherlands, to continue building their lives there.

As depicted in Table 2.1, the Turkish community is currently the largest group of citizens in the Netherlands with a migration background, and also has the most first generation migrants. Second place is Morocco, showing a slight increase in second generation from Türkiye. The largest group of second generation migrants are Indonesians, but the group of first generation migrants is in their case significantly lower than the other groups.

Between the two largest groups, the Turkish and Moroccan communities show some differences in language use. Nortier and Dorleijn [2008] found that people from Türkiye alternate between speaking Turkish and Dutch with their peers, depending on the majority language in the group. However, this is different for Moroccans in the Netherlands. Although Moroccans alternate as well, Dutch Moroccans usually communicate in Dutch. Specifically, a language variety that relates to Straattaal and is typically called the *Moroccan-Dutch ethnolect*.

2.2.2 Terminology of multilingual language varieties

A few decades after the first generation migration wave, the second generation of the immigrated ethnic groups have created *Straattaal*. Although there have been attempts to list lexical items that are commonly used in Straattaal [Appel 1999], a clear-cut definition has not arisen. Kossmann [2019] even states that the term can be interpreted as its literal translation 'street language': 'the type of speech that you would use on the streets (i.e. outside home, classroom or work)'. As stated in Chapter 1, Straattaal is heavily influenced by Sranan Tongo, a Surinamese language, but also draws from Papiamentu, Turkish, Berber, vernacular Dutch and English [Appel 1999, Kossmann 2019].

An example from Kossmann [2019]:

(12) Hind: Ey *faka* dan met die *torrie* op *scorro*?
Hind: Ey hi$_{[\text{Sranan}]}$ then with that story$_{[\text{Sranan}]}$ at school$_{[\text{Sranan}]}$?
Hind: Ey what's up then with that story at school?

(13) Was er *fittie* tussen Mo en Apps, *wollah* gruwelijk *eh mattie*.
Was there fight$_{[\text{Sranan}]}$ between Mo and Apps, by God!$_{[\text{Arabic}]}$, gruesome, mate$_{[\text{Sranan}]}$.
Was there a fight between Mo and Apps, by God!, gruesome, mate.'

As shown in both (12) and (13), Straattaal is characterized by an informal Dutch sentence structure, thus acting as the matrix language, alternated by insertions of single words from various other languages. This is different from code-switching, as there is not one 'embedded language', but rather an embedded lexicon. Straattaal has little function words nor its own grammatical structure, which prevents it from acting as the matrix language. It is therefore rarely classified as a language, but rather a linguistic variety.

Several terms have been created in order to describe linguistic varieties like Straattaal, such as:

Urban Youth Speech Styles (UYSS) [Dorleijn et al. 2015].
Dorleijn et al. [2020] define UYSS as: 'Linguistic practices involving the use of linguistic material from different languages by young people in a multilingual urban environment, with a performative character, and which the speaker can control.'

Ethnolect: Originally non-indigenous intermediate varieties of the dominant language in an area, are referred to as 'ethnolects' [Hinskens 2011]. The word ethnolect consists of 'ethno', meaning with regard to ethnicity, and 'lect', implying a set of linguistic features that separates it from standard language or other 'lects' [Quist 2008]. The term ethnolect has been criticized for both parts: The term 'ethno' is imprecise and directs attention to just one aspect of a socially complex phenomenon, and is deemed unsuitable for migrants in Europe

[Kossmann 2019] and 'lect' is said to be inappropriate for describing a style, stylization or variety [Nortier and Dorleijn 2013].

Multi-Ethnolect: A multi-ethnolect is different from an ethnolect, as it arises from the dynamics of multi-heritage groups, instead of a single heritage [Cheshire et al. 2011]. Multi-ethnolects typically emerge in particular among adolescents in the context of ethnically diverse urban regions, and its users can have any ethnic background, including the dominant, mainstream background, like ethnically Dutch youths using Straattaal [Nortier and Dorleijn 2013].

Unlike regional dialects, such as *Limburgs*, multi-ethnolects are not necessarily bound to specific ethnicities or geographic areas. Multi-ethnolects come in many forms, depending on the dominant language in the area as well as on the present multi-heritage groups. For example, in Amsterdam, the Netherlands, a multi-ethnolect is manifested as *Straattaal*, but in Oslo, Norway, a multi-ethnolect takes the form of *Kebabnorsk* [Nortier and Dorleijn 2013]. Speakers of multi-ethnolects in Western Europe often have a background from outside Western Europe, like Morocco, Suriname or Indonesia. However, a multi-ethnolect emerges anywhere with multi-ethnic prevalence, and is therefore not exclusive to Western Europe, but also occurs in African, American and Asian cities [Svendsen 2015].

Communicating in an ethnolect that takes words from their own background, provides a way for immigrants to construct and present a sense of identity [Svendsen 2015]. To explain why (multi-)ethnolect speakers are so involved in identity construction and presentation, Nortier [2016] names it an 'unavoidable consequence' of living in the large multi-ethnic urban area in a landscape with a plethora of linguistic possibilities, without any dominant tendencies. However, borrowing lexical items from Berber or Arabic can also be a way of keying a message to be ironic, and not meant seriously, especially when used in an online, informal environment, like in (14) [Kossmann 2017]:

(14) *Oepppaaaaaaaaaaaaaaaaaaaa* *wajoow* me moeder wilt me naar gekken
Up! *wow (wayyaw)* my mother wants me to mad
tehuis sturen *wollah* ik zit *bhel* *shi* zombi achter de laptop
house send *by god! (wollah)* I sit *like (bḥal) some (ši)* zombie behind the laptop
om te kijken of je al een vervolg had geplaatst!!
to be looking whether you already a sequel had posted.
Up! my mother wants to send me to the mad house, by god I sit like some kind of zombie behind the laptop looking if you already posted a sequel!!

To prevent further confusion by alternating between terms, I will refer to Straattaal as a multi-ethnolect, and to Moroccan-specific linguistic varieties of Dutch as the Moroccan-Dutch ethnolect.

2.2.3 Moroccan Languages and the Moroccan-Dutch ethnolect

The two most common languages in Morocco stem from different language families. Moroccan Arabic is a Semitic language, whereas the Berber languages are categorized as a Hamito-Kushitic language or as a category of its own within the Afro-Asiatic language families. Both languages influenced each other due to their longtime parallel existence in Morocco, but not to the extent that they became mutually intelligible [Dorleijn and Nortier 2009].

In the Netherlands, the majority of the Moroccan community is a Berber-speaker, as most immigrants originate from a region in the north of Morocco called the Rif, where *Tarifiyt Berber* is the dominant language [Kossmann 2017]. Generally, Berber-speakers in Morocco are to some degree familiar with Moroccan Arabic, as it is considered the *lingua franca*, or 'bridge language' in Morocco. In the Netherlands on the other hand, fluency in Arabic is not common among Moroccans [El Aissati 1997]. Berber is not taught in schools in the Netherlands, and does not have one standard written form, so the second and third generation of Moroccans in the Netherlands have higher proficiency in Dutch. Another complication is the unequal status of Berber and Arabic, which reflects a diglossic situation. This diglossia signifies the presence of distinct but interconnected language forms, where one form is typically used locally and informally, and another is more prestigious and universal [Ferguson 1959]. For example, varieties of Colloquial Arabic and Classical Arabic [Bell 2013, p.109].

Although the Berber language is not a different version of the Arabic language, the social part of the diglossia phenomenon is still present, as the Berber language is stigmatized in Morocco and an Arabic speaker may be offended when addressed in Berber [Dorleijn and Nortier 2009]. To mitigate such problems, when Moroccan-Dutch speakers are unsure whether their interlocutor speaks Arabic or Berber, nor which dialect of either language, they speak Dutch instead, as it is the most neutral choice [Dorleijn and Nortier 2009, El Aissati 2002].

Dutch Moroccans speak their own ethnolect, which is among other factors characterized by occasionally inserting words from Berber and Arabic. These insertions are often function words such as indefinite articles, like in (15) or question markers, see (16) [Kossmann 2017].

(15)

En	*wa7id*	meisje	doet	*fh7al*	*shie*	lesbie	&			
And	*one (waḥid)*$_{\text{[Arabic]}}$	girl	acts	*like (fḥal)*$_{\text{[Arabic]}}$	*some (ši)*$_{\text{[Arabic]}}$	lesbian	and			
knuffelde	*wa7id*	vriendin	van	mij	en	die	vond	dat	kapot	eng.
hugged	*a (waḥid)*$_{\text{[Arabic]}}$	friend	of	mine	and	she	found	that	terribly	scary.

(16)

wesh	ga je	nog	verder	of niet?
question marker$_{\text{[Arabic]}}$	are you going even further or not?			

Although the inserted function words often exist in both Berber and Arabic, usually only one of the words is used in the Moroccan-Dutch ethnolect. For instance the Berber equivalent 'ma' of the Arabic question marker 'wesh', is almost not used [Kossmann 2017]. Also, although a speaker may use Berber when speaking full sentences in the heritage language, insertions while speaking Dutch will still come from both languages, often in the same sentence, like in (17).

(17) en als ik de slappelach heb dan lach ik *bhal* *izjen*
and when I have a fit of laughter then I laugh *like*$_{\text{[Arabic]}}$ *some kind of*$_{\text{[Berber]}}$
spongebob.
Spongebob.

The Moroccan-Dutch ethnolect, or Moroccan Flavored Dutch (MFD) as Nortier and Dorleijn [2008] call it, is recognizable mostly phonetically and include characteristics such as a strongly voiced /z/, or the use of /sh/ instead of /s/ in consonant combinations, like in the word ‘school’ [Nortier 2018]. On the internet, identity messages can only be conveyed through written text, which is therefore rich in terms of identity markers [Nortier 2018]. For instance, the ethnolect is shown by spelling words phonetically. Here are two examples from [Nortier 2016]:

(18) ben nu op shgool en we miss you a lot!!
(I) am now at school and we miss you a lot!!
(I) am now at school and we miss you a lot!!

(19) als je voor shlet aangezien wilt worden
if you for slut considered want to be
if you want to be considered a slut

Regularly, the Dutch words are spelled ‘school’ and ‘slet’, without the ‘h’, unlike in (18) and (19).

This overlaps with another characteristic of multi-ethnolects: *rebellion spelling*, in which users deliberately spell words incorrectly for expressive purposes [Sebba 2003, Shaw 2008]. However, misspellings in the form of letter repetition are also a characteristic of computer mediated communication in general, showing creativity and excitement [Darics 2013].

For the insertions from neither of the two Moroccan languages there is a commonly known orthography, resulting in creative freedom when writing sounds that lack a simple Dutch equivalent. For example, Kossmann [2017] found 57 different spellings of ‘wahed’ (one) on a Moroccan-Dutch internet forum.

2.2.4 Moroccan-Dutch ethnolect and Straattaal compared

Interestingly, the ethnolect that the Moroccans created, has come to be popular with people from different sociocultural backgrounds: white, native Dutch youths also pick up on this way of speaking, as do other Mediterranean and/or Muslim ethnic minorities in the Netherlands that do not identify as Moroccan [Nortier and Dorleijn 2008]. Other ethnic minorities in the

Netherlands, like people from a Surinamese, Antillian or African background are not typical users of the Moroccan-Dutch ethnolect, while they do engage in Straattaal [Cornips and de Rooij 2003].

Kossmann [2019] performed analysis on a Moroccan Dutch chat forum named chaima.nl, which is used as a space for young Moroccan girls, characterized by its all-pink lay-out. According to Kossmann [2017], the demographic that uses the forum are Moroccan-Dutch girls between 12-22 years old, who live in a big city in the Netherlands and 'almost invariably present themselves as sensible, decent girls that attach great importance to being a good Muslim.' Kossmann attempted to create categories for Straattaal and the Moroccan-Dutch ethnolect, to improve delimiting the two styles. They found that most insertions from Moroccan languages are function words and only a few content words from Berber and Arabic would appear regularly in the chats, but all of these have a strong emotional connotative association, e.g. words for 'Morocco', 'Berber', and derogatory term for 'police'.

What sets the usage of the Straattaal lexicon apart from the Moroccan lexical elements, is the choice of which nouns and adjectives are taken from Sranan in comparison to Moroccan. The Sranan lexicon provides quite mundane and concrete nouns in Straattaal, e.g. 'oso'(house), 'pata' (shoe). This stands in contrast to the Moroccan Lexical elements used in Straattaal, which have strong cultural and expressive associations, like 'rwina' (chaos), 'zamel'(homosexual), or 'lmegrib' (Morocco) [Kossmann 2019].

Moroccans frequently use Sranan words as well as Moroccan words in informal conversations held in Straattaal, whereas Surinamese speakers do not mix Moroccan elements into their speech as much. An example of a Moroccan speaker from Nortier and Dorleijn [2013]:

(20)	tfoee	jullie	hebben (...)	nog	nooit	in jilla	gezeten	he?
	shit$_{\text{[Berber]}}$	you guys	have	yet	never	in jail$_{\text{[Sranan]}}$	been	right?

shit you guys have never been in jail right?

2.2.5 Moroccorp

The Moroccorp [Ruette and Van de Velde 2013] is a dataset that documents the written form of the the Moroccan-Dutch ethnolect: mixing Straattaal, Dutch, Arabic, Berber and English. I will use this dataset for my research goals and describe the properties of this dataset in more detail in Section 3.1, while diving into its content here.

Although being certain about the identity of the chatters is nearly impossible, many clues were given about their ties to Morocco. Usernames often refer to their Moroccan identity, e.g. 'femmedumaroc' (woman from Morocco), 'marokkaans20jaar' (morroccan20years). Moreover, the corpus' authors report that approximately 1% of the lines in the corpus contain a reference to Morocco. Ruette and Van de Velde [2013] performed a lexical analysis to collect evidence that the Moroccorp is representative for Dutch chat language of users with a Moroccan background, as it was spoken in the summer of 2012. For this analysis, they used a Stable

Words	Description
salaam, salam, wslm, ewa, beslama	Moroccan greetings
marokkaanse, marokko, marokkanen	self references
gwn, wrm	internet acronyms
wollah, hmdl	Moroccan exclamations
broeder, trouwen, dame, vader	topic words
islam, moslim, ramadan, allah	religion

Table 2.2 Table from Ruette and Van de Velde [2013], showing evidence for affiliation with Morocco of corpus

Lexical Marker analysis from Speelman et al. [2008] in which they compare frequencies of certain words in this corpus to a reference corpus. As a reference corpus, they used the ConDiv corpus [Grondelaers et al. 2000], another Dutch chat corpus, without any ties to a Moroccan community. Ruette et al. found six categories of words that received a particularly high score, meaning that these words appeared statistically more frequently in the Moroccorp than in the ConDiv corpus. See Table 2.2. Especially the first, second, fourth and sixth categories are strongly Moroccan-themed.

Another indication which shows that this corpus can be used to study the Moroccan-Dutch ethnolect is the statistically more prevalent existence of a wrongly inflected adjective before a noun with a neutral gender, e.g. 'een mooi**e** huis' (a beautiful house), instead of the correct 'een mooi huis'. This is a common mistake for any learner of Dutch as a second language, as well as for Moroccan Dutch speakers, but very uncommon for native Dutch speakers without a migration background [Ziemann et al. 2011]. Ruette et al. performed a statistical analysis on the corpus, and found that this construction was present in about a third of the cases, whereas the reference corpus ConDiv showed barely any signs of it.

2.3 Code-switching and Language Identification in NLP

Language Identification (LID) has been a topic of interest for quite a long time in the context of information science [Gold 1967]. Typically, this was done to identify the language a monolingual document was written in. However, other forms of LID, such as obtaining word-level language labels for multilingual code-switched texts are more complicated and therefore interesting to investigate. Methods for LID range from simply tagging individual words without taking context into account, using dictionaries or statistics, to more complex models that use LSTMs or Transformer architectures. Incorporating context does improve performance compared to static dictionaries, but a high focus on the context can also reduce performance in bilingual documents [Nguyen and Doğruöz 2013].

Over the past few years, NLP has seen several initiatives for LID. For instance, several shared tasks [Barman et al. 2014, Molina et al. 2016, Solorio et al. 2014] and benchmarks on code-switched data were created for various NLP tasks, including LID [Aguilar et al. 2020, Khanuja et al. 2020]. Each of these works have similarities and small differences in their approach on dataset annotation, and the training and evaluation of types of models, which I discuss in Section 2.3.2 and Section 2.3.3, respectively. First, I will elaborate on the challenges in Language Identification.

2.3.1 Challenges in Language Identification

Multilingual documents bring different challenges depending on the languages involved. For instance, two languages can share similar linguistic features, such as vocabulary or grammatical structure, making it difficult to determine which is which [Molina et al. 2016]. This is even more difficult when the task is to distinguish between different dialects of the same language, like Modern Standard Arabic and Egyptian Arabic, which have a significantly large word overlap [Solorio et al. 2014]. However, the same words often do not mean the same thing in each dialect [Aguilar et al. 2020].

Likewise, research shows that language pairs that are more different from each other, e.g. Nepali and English, are easier for a deep learning model to distinguish than languages that are more similar [Molina et al. 2016, Solorio et al. 2014]. Differences between languages are most apparent when the languages involved in a document each use a different writing system, such as Arabic-Dutch or Hindi-English. This is called *script variance*. However, in online text forums or other informal settings, like social media, the Roman script is most often used for languages that originally use another script, for convenience of typing [Jose et al. 2020, Rosowsky 2010]. Romanized versions of languages that originally use a different script do not have standardized spellings, which makes it more difficult for models to recognize words [Khanuja et al. 2020]. Especially for a lower-resource language like Nepali, this is a challenge, as most monolingual resources for Nepali are written in its own script, making it difficult to align the two versions [Aguilar et al. 2020].

Moreover, Barman et al. [2014] observes that romanizing a language that originally uses a non-roman writing system, increases the chance of code-switching with a language that uses the Roman alphabet. Grammatical difference also encourages the amount of code-switching points, since languages preserve their grammatical structure when code-switching occurs. For example, Nepali has a Subject-Object-Verb (SOV) structure, whereas English has a Subject-Verb-Object (SVO) structure. This calls for more nuanced alternations to adhere to both SOV and SVO structures.

Aguilar et al. [2020] created a Benchmark for code-switched data on various NLP tasks: Linguistic Code-switching Evaluation (LinCE). It comprises ten datasets, spanning over four language pairs, Spanish-English, Hindi-English, Nepali-English and Modern Standard Arabic-Egyptian Arabic and four NLP tasks, including Language Identification. Findings by

Aguilar et al. [2020] confirm that the code-switching rate in a Nepali-English dataset was above average, compared to datasets that involved languages that are both traditionally written in the Roman script and use the same grammatical structure. Despite the high prevalence of code-switching in the Nepali-English dataset, the models deployed in LinCE were able to reach results on the LID task that were similar to those reached on a code-switched Spanish-English dataset [Aguilar et al. 2020]. Conversely, the dataset consisting of code-switching between Egyptian Arabic and Modern Standard Arabic proved to be much more difficult for these models, suggesting that vocabulary overlap is one of the hardest challenges to overcome for LID.

Another possible issue is that the algorithm that performs LID may not be trained on the same character encoding standard as the one present in the to be processed document [Das and Gambäck 2014]. For example, the algorithm is trained on ASCII-encoded text, but the text to be identified is encoded using the UTF-8 character set, it can lead to problems [Jauhiainen et al. 2018]. If the language used in a document is new, or relatively rare, it may not exist in the algorithm's set of labels, making it impossible to identify the language accurately. For instance, Python library LangID is trained on 97 languages, but not on Berber, so the Berber language would be inaccurately identified [Lui and Baldwin 2012]. To be able to identify rare languages, like Straattaal or Berber, a model can be trained on custom-defined labels for each relevant language used within a dataset.

2.3.2 Annotation decisions for code-switched datasets

Choosing target labels for a dataset is an important part of Language Identification, as it specifies what a model trains on to learn. There are two parts to this: the span of a label and its content.

Most datasets are annotated on token-level [Maharjan et al. 2015, Molina et al. 2016, Solorio et al. 2014], but sometimes take into account longer sequences. This can be forum posts [Nguyen and Doğruöz 2013], sentences or even fragments, like in Barman et al. [2014]. See Table 2.3 for an overview in which I color-coded the different levels of annotation. Nguyen and Doğruöz [2013] annotated forum posts on token-level, but additionally annotated posts as either bilingual (BL), for the appearance of both languages, or either Turkish (TR) or Dutch (NL). Barman et al. [2014] tag four different levels of code-mixing: sentence, fragment, inclusion and word-level code-mixing. A sentence is identified with its base language. If this language was mixed, a fragment for each language was also identified. The fragment tag showcases intrasentential code-switching.

Barman et al. [2014] define an *inclusion* as a foreign word in a sentence or fragment which is assimilated in the native language. This label is similar to lexical borrowing, such as used in [Mellado and Lignos 2022], but not equivalent, as it depends on the definition of lexical borrowing and code-mixing one works with. Barman et al. [2014] do not give a clear definition

Annotation Level	Example
Post	[BL] [NL] Mijn dag kan niet stuk :) [/NL]
	[TR] Cok guzel bir haber aldim [/TR] [/BL]
Token	[TR] kahvalti[/TR][NL]met vriendinnen by my thuis [/NL]
Sentence	[EN] what a.....6 hrs long...but really nice tennis.... [/EN]
Fragment	[MIXD] [HI] oye hoye .. angreji me kahte hai ke[/HI]
	[EN] I love u.. !!! [/EN][/MIXD]
Inclusion	[BN] Na re [“EN”] seriously [/incl]
	ami khub kharap achi.[/BN]
Word-level code-mixing	[“en-and-bn-suffix”] classe [/wlcm]

Table 2.3 Levels of annotation beyond token-level, first two examples from Nguyen and Doğruöz [2013] and others from Barman et al. [2014]

on when a word is ’assimilated’ and when not, whereas Mellado and Lignos [2022] do make explicit what they count as a lexical borrowing, which I listed in Section 2.1.2.

On the most fine-grained level, word-level code-mixing was identified if a word from one language was used with a prefix or suffix of another language. The labels in Barman et al. [2014] also specified which languages were present and how they were mixed in the word, distinguishing ‘en-and-bn-suffix’ from ‘bn-and-en-suffix’, instead of using simply ‘mixed’.

Even if longer sequences were taken into account, the data was still annotated on token level in order to make that happen. This prompted me to decide to also annotate on token-level.

2.3.2.1 Labels used for LID

Most researchers use labels for the most common languages in a dataset, as well as labels for named entities and language independent utterances. Nguyen and Doğruöz [2013] ignored language independent tokens altogether and deleted phrases that consisted exclusively of tokens that did not fit in with either language they were investigating and would have been categorized as *other*. In addition to the most commonly found labels, other labels can be applied depending on the focus of a project. These additional labels are listed in Table 2.4.

A label for ambiguous, unknown or otherwise hard to categorize tokens is often added as well, like *unk, undef, ambiguous* [Aguilar et al. 2020, Barman et al. 2014, Das and Gambäck 2014]. Some take into account intra-word code-switching in the form of a *mixed* label [Maharjan et al. 2015, Solorio et al. 2014], whereas others specify which language mixes with which, in the form *en+hi_suffix* [Das and Gambäck 2014]. Especially in morphologically rich languages, intra-word code-switching is used frequently, which makes sub-word level Language Identification necessary [Mager et al. 2019]. Lexical borrowing can be distinguished

Label names	Description
Foreign word (Fw), Other	Language different from the focus languages
Ambiguous	Possible to be either of the focus languages
Unk, Undef	Unrecognizable words, unable to classify otherwise
Mixed, Bilingual	Words or sequences partially in both focus languages
Mixed+en-hi-suffix	Like Mixed, specifiying present languages
Borrow	Lexical borrowing

Table 2.4 Overview of other possible labels to include for LID, from Aguilar et al. [2020], Barman et al. [2014], Das and Gambäck [2014], Maharjan et al. [2015], Mellado and Lignos [2022], Solorio et al. [2014]

from code-switching with a separate label, if there are clear boundaries as to what counts as a borrowing and what does not [Mellado and Lignos 2022].

In the annotated datasets, it can happen that created labels are rarely found in the dataset. For instance, in Aguilar et al. [2020] the labels *unk* (unknown) and *fw* (other language) were almost not used. *Mixed* and *ambiguous* were also infrequent.

2.3.3 Model Approaches and Evaluations for LID

Language Identification has seen many different approaches, from n-grams to deep learning. I will discuss several approaches in this section.

Solorio et al. [2014] organized a shared task on language detection. Different implementations were submitted, of which Support Vector Machines and Conditional Random Fields were the most popular. Models were tested on four types of code-switched data, each containing a different language pair, Nepali-English (NEP-EN), Dialectal Arabic-Modern Standard Arabic (DA-MSA), Spanish-English (SPA-EN) and Mandarin-English (MAN-EN). They found that the language pairs that are more different from each other, e.g. Nepali and English, are easier for a model to distinguish, than languages that are more closely related, like Modern Standard Arabic and Dialect Arabic, which received the lowest scores. The highest avg F-measure was 0,417, reached by a Support Vector Machine (SVM). It was also the only language pair for which at least one model was not able to surpass a lexicon-based baseline. The language pair NEP-EN reaches the highest average F-measure of 0,977, using an SVM. The highest score for SPA-EN was also reached using SVMs (F-measure: 0,95). However, For MAN-EN, an extended Markov model performed the best (F-measure : 0,894), followed by a Conditional Random Field (CRF) model.

Molina et al. [2016] also did a shared task two years later. They only used two datasets, again the language pairs SPA-EN and DA-MSA. The most popular implementation was still a Conditional Random Field. For this second shared task, two Deep Learning models were present, one CNN and a LSTM model. The LSTM model performed the best on the DA-MSA dataset, with a weighted average F1-score of 0,876. Although the same model performed high (Avg-F: 0,968) on the SPA-EN dataset, it was outperformed by a Logistic Regression model, which received a weighted average F1-score of 0,973.

In many cases, deep learning approaches return improved results over traditional machine learning approaches. Iliescu et al. [2021] compared methods like n-grams, support vector machines and logistic regression to BERT-based approaches for code-switch detection, and found that semi-supervised methods are performing worse. The best in this category was a Viterbi model, with a weighted F1-score of 92.23%, whereas the BERT model scored 98.43%.

In the same category, Lui and Baldwin [2012] created a single python file that loads in a fast-working naive Bayes classifier that can identify up to 97 languages, receiving an accuracy score of up to 99% on certain domains, and 91.3% on a Wikipedia dataset. The model has not been evaluated on datasets that use code-switching, and is most successful on document-level.

Another way to facilitate working with multiple languages in one model, is multilingual token representation. Cross-lingual embedding techniques, like BiCCA [Faruqui and Dyer 2014] or BiSkip [Luong et al. 2015], have shown to perform well on cross-lingual tasks. However, they are not ideal for processing code-switched texts [Pratapa et al. 2018].Khanuja et al. [2020] found that Multilingual BERT (M-BERT) [Pires et al. 2019] outperforms cross-lingual embedding techniques on language detection (F1-score: 96,6), as well as the other tasks of GLUECoS, a benchmark for code-switched language processing. This includes Language Identification from text, POS tagging, Named Entity Recognition, Sentiment Analysis, Question Answering and Natural Language Inference for code-switching [Khanuja et al. 2020]. They used two versions of M-BERT, one that is called *bert-base-multilingual-cased* and a modified M-BERT, that was fine-tuned by Sun et al. [2019]. Fine-tuned M-BERT performed better or nearly the same as baseline M-BERT on each task of the GLUECoS benchmark.

Aguilar et al. [2020] created another Benchmark for code-switched data on various NLP tasks: Linguistic Code-switching Evaluation (LinCE). Aguilar et al. [2020] used BiLSTM, ELMo and M-BERT for the LID task. For this benchmark M-BERT also yielded the highest accuracy for each language pair, the highest being Spanish-English (0,9853) and the lowest MSA-EA (0,8414).

In the next section, I elaborate on M-BERT and its architecture, the transformer.

2.4 Transformer models

The Large Language Models of today, like GPT-3 [Brown et al. 2020], XL-net [Yang et al. 2019] and BERT [Devlin et al. 2019] are all run on the Transformer architecture, that was introduced by Vaswani et al. [2017]. The transformer uses a self-attention mechanism that makes it possible to create contextualized embeddings, that create a different embedding for every occurrence of the same word in a different context, as well as facilitate working with long-range dependencies in sentences, and therefore create a more accurate representation of language than simpler models.

Within the transformer architecture, there are various types of models that are usually trained for a certain task. A transformer can be found as encoder-decoder, encoder-only or decoder-only. An encoder-only transformer is designed to process input data, without generating an output. It is used for tasks where the primary objective is to encode and represent tokens into a meaningful way, like text classification. Conversely, a decoder-only transformer's main function is to use pre-existing representations to generate an output sequence in the form of readable text. The BERT models are encoder-only and therefore mainly used for tasks that require Language Understanding and the GPT models are decoder-only, which is more suitable for Text Generation. An encoder-decoder model would be suitable for tasks that need to understand language as well as generate language, like Machine Translation.

Large Language Models (LLM) that are transformer-based have shown to be good at language processing tasks [Devlin et al. 2019, Liu et al. 2019, Pires et al. 2019]. Generally, a LLM is pre-trained once on a large, unlabeled training dataset and then fine-tuned on a smaller dataset for each time the model needs to perform a specific downstream task, like Question Answering or Language Identification.

As the transformer models I use are both BERT models, I discuss their architecture and pre-training in the next sections.

2.4.1 Pre-training BERT models

Pre-training a transformer-based language model can be split into two steps: tokenization and training. A BERT model is comprised of multiple layers of encoders, usually 12. The first encoder is provided with embeddings from a tokenization algorithm, and each following encoder works with the output from the one before. An encoder has two parts: a self-attention layer and a feed forward neural network. For a visual representation, I added Figure 2.2, which depicts the journey of a token through an encoder layer.

2.4.1.1 Tokenization

Each LLM uses a tokenizer, that splits the text into pieces: words and/or subwords. To represent these (sub)words in the model, they are embedded with a vector filled with real-valued numbers that contains semantic and syntactic information about a word, based on frequent co-occurrences with other words. There are several strategies to split words into tokens. For example, BERT and Multilingual BERT [Devlin et al. 2019] use WordPiece [Wu et al. 2016], an algorithm for tokenizing a set amount of tokens, prioritizing words that appear frequently. Some newer models, like RobBERT and RoBERTa [Delobelle et al. 2020, Liu et al. 2019], use Byte Pair Encoding (BPE) [Sennrich et al. 2016]. The BPE algorithm incrementally constructs its vocabulary by replacing the most frequently occurring sequential tokens with a

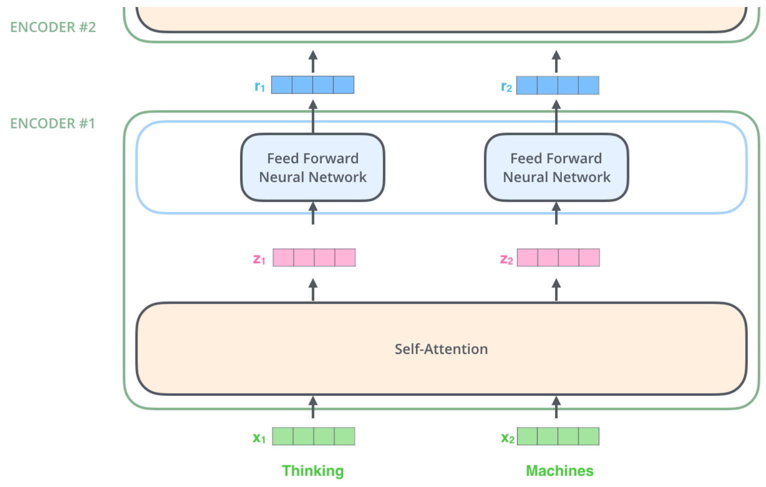


Figure 2.2 The contents of one encoder, by Alammar [2018]

novel, combined token. For example, ‘c’ occurs often with ‘at’, creating ‘cat’. For RobBERT, a vocabulary size constraint of 40,000 words was imposed. An advantage of this technique is being able to encode larger texts while retaining a smaller vocabulary. However, BPE also tends to add words multiple times, e.g. including and excluding white spaces or capitalized and lowercase versions of the same words.

2.4.1.2 Training Architecture: Self-attention

The tokenized vocabulary is then mapped to word embeddings of a set length (768 for BERT). These are initially random, and are trained in each layer. The created word embeddings are given to the first self attention layer, where three vectors are made: a query, a key and a value vector for each word embedding. To make this more clear, I included a visualization by Alammar [2018], see Figure 2.3. The figure depicts the different vectors that are used as a representation for a word. These vectors are calculated by multiplying the input vectors by a weight matrix, one for each of the three. Typically, the three vectors are much smaller than the input or output vectors for computation time reduction. The three vectors are used to calculate self-attention for each token. A score is calculated by comparing a token’s query vector with the key vector from every other word. This makes clear which words in a sentence depend on each other or are related to each other. Then the value vectors are multiplied by this score and added together, to create an output vector which is sent along to the feed forward neural network. This creation of the query, key and value vectors and subsequent processing is called an attention head. Most models use more than one attention head (Both BERT and RobBERT use twelve), each randomly initialized and then trained. Instead of sending each

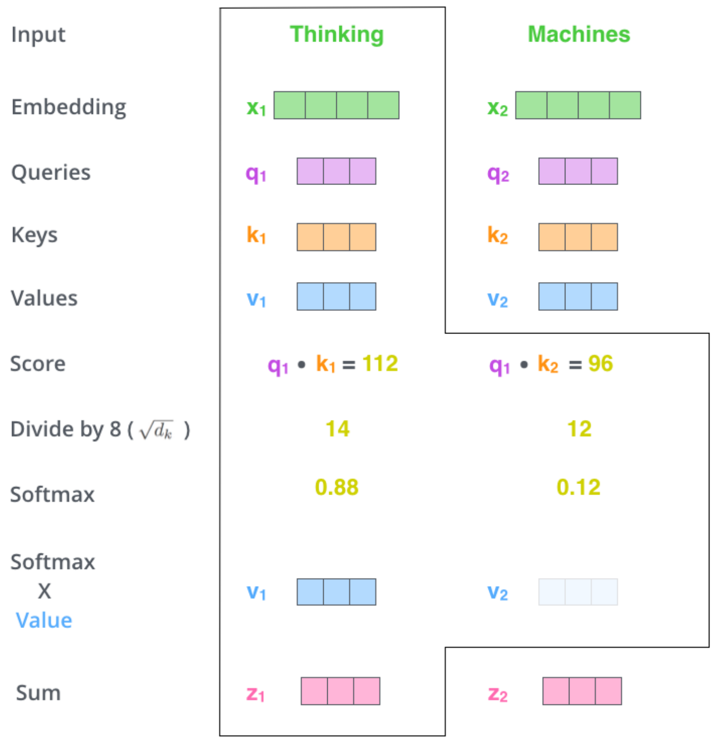


Figure 2.3 Example of self-attention per word, from Alammar [2018]

of these vectors to the feed forward network individually, the model concatenates the output vectors from each attention head to one large vector and multiplies it with another trained weight matrix, in order to produce a smaller vector that captures the information from all attention heads. For clarity of this process, I again added a visual from Alammar [2018], see Figure 2.4. This recaps the process from the tokenization of an input sentence to the end of a self-attention layer.

In addition to each sublayer of the encoder (self-attention and feed forward), layer normalization is applied, by adding the original word embeddings to the output vectors.

2.4.1.3 Training: MLM and NSP

The model's next step is to train the embeddings of the created tokens, by having the feed-forward neural network in each layer of the model perform a task. The BERT models are pre-trained on two tasks. One is Masked Language Modeling (MLM), a self-supervised task that has facilitated the use of large datasets, as no annotation is needed, making virtually any dataset instantly usable. During MLM, the model conceals one or more words in a sequence

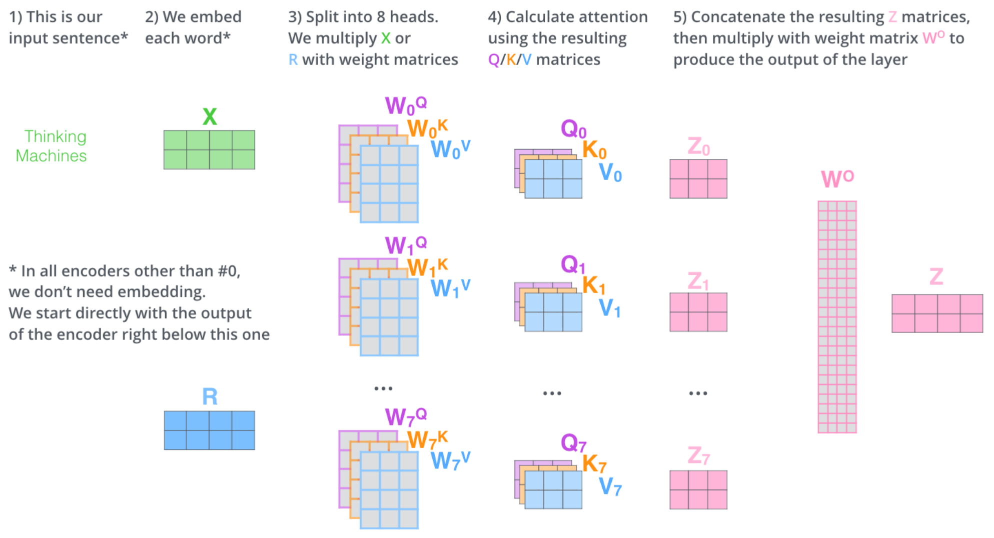

1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z

Figure 2.4 Recap of self-attention layer, from Alammar [2018]

and makes itself provide the word on its own accord, then checks if it was correct. With regard to the difference between the prediction and the ground truth, a Loss score is computed and the weights of the masked words at present are updated to reduce the loss and feed-forwarded into the next layer of the model. This repeats itself until a set amount of training epochs have elapsed. Masking can be done statically, or dynamically. During static masking, masking is performed once in the preprocessing phase, resulting in showing a training sequence with the same mask multiple times. Dynamic masking generates the mask each time a sequence is fed to the model [Liu et al. 2019]. A second task BERT was trained on is called Next Sentence Prediction (NSP), which asks to infer based on two input sentences if they follow each other. One version of this task takes sentences randomly, and another either swaps two sentences or leaves them in their original order. Liu et al. [2019] concluded that leaving out NSP did not alter the results much. This task is left out in both RoBERTa and RobBERT.

2.4.2 Fine-tuning a transformer for token classification

A pre-trained transformer model, like BERT, has trained embeddings for tokens, taking into account its frequent context. However, before it can perform specific tasks, it needs to be trained further on such a task, which is called fine-tuning. Fine-tuning is done by sending labeled training data through all the layers of the pre-trained model, including an additional final neural network layer, which produces a relevant output for the specified task. This layer is often randomly initialized. In case of Token classification this final layer is linear, but for other tasks, such as Sentiment Analysis, it can also be made non-linear with an activation function, like ReLU.

The pre-trained layers of the BERT model outputs a tensor with hidden units, containing information about the input sentence. These are then forwarded to the final layer: the classifier. During fine-tuning, the total network is trained by forwarding batches of data through the network, computing the loss, gradients and then updating the weights of the network using the optimizer. This is repeated for a set amount of epochs.

The loss can be computed in various ways, but a common loss function for token classification is Cross Entropy Loss with softmax. This transforms the outputs from the interim results to probabilities, and compares the model's results to the original labels, which are one-hot encoded vectors. The optimizer then aims to minimize the loss, by updating the weights.

The classifier outputs raw logits, transforming the information from BERT into a pre-defined number of categories. A final prediction can be given by simply taking the logit with the highest value, or first transforming these into probabilities using a softmax function. The category with the highest probability is the one the model assigns to the token, and this is then evaluated by comparing it to the actual label.

Subsequently, the fine-tuned model can be evaluated, and tuned on its hyperparameters, like learning rate, batch size or amount of epochs and eventually used to perform the task it was trained for.

2.4.3 Pre-trained transformer models suitable for handling Moroccan-Dutch ethnolect

My aim is to find a transformer model capable of language identification within a predominantly Dutch dataset containing elements of various languages. Popular and large deep learning models for the Dutch language, like BERT-NL [Brandsen et al. 2019] and BERTje [De Vries et al. 2019] are trained on standard Dutch, as they were trained on Dutch Wikipedia articles. A newer Dutch BERT-model, named RobBERT [Delobelle et al. 2020], uses the same architecture as RoBERTa [Liu et al. 2019], which is a re-implemented and improved version of BERT [Devlin et al. 2019]. The creators of RobBERT initially designed two separate versions. The first version uses the same vocabulary as RoBERTa. Despite RobBERT being designed for another language, it is possible to use the same vocabulary, as this vocabulary was created using BPE [Sennrich et al. 2016], and should therefore be language independent in principle. For the second version Delobelle et al. [2020] created a new vocabulary using a BPE algorithm, but applied to a Dutch text. The second version was shown to perform better, and is henceforth the only version I will refer to.

As shown in Table 2.5, RobBERT outperforms both BERTje and BERT-NL on Sentiment Analysis [Delobelle et al. 2020]. Also, RobBERT performs the best on Die/Dat Disambiguation, which is a Dutch-specific task, asking the model to predict which demonstrative pronoun is needed before a noun, depending on the gender of the noun. Also, RobBERT was shown

Task and metrics	RobBERT	BERTje	BERT-NL	M-BERT
Sentiment Analysis (F1-score)	**95,14**	-	84,0	-
Named Entity Recognition (F1-score)	89,08	88,3	89,7	**90,94**
Part-of-speech Tagging (Accuracy)	96,4	96,3	-	**96,5**
Die/Dat Disambiguation (Accuracy)	**98,75**	94,94	-	90,21

Table 2.5 Comparing scores per model on downstream NLP tasks, as found in Delobelle et al. [2020]. For Sentiment Analysis and Named Entity Recognition the F1-score is used. For the other two tasks, accuracy is used. The highest score per task is highlighted in **bold**.

Model	M-BERT	RobBERT
Architecture	BERT	RoBERTa
Training Task	MLM and NSP	MLM
Data Source	Wikipedia in 104 languages	Dutch part of OSCAR
Size of Training Data	unknown[2]	39GB or 6.6M words
Size	170M Parameters	355M Parameters
Tokenization algorithm	WordPiece	Byte Pair Encoding (BPE)
Vocabulary Size	105.879 tokens	40.000 tokens

Table 2.6 Comparing M-BERT and RobBERT on their pre-training procedures.

to perform better on smaller datasets, making it a suitable model for taks with limited data available.

RobBERT was trained on the Dutch part of the OSCAR corpus. The OSCAR corpus is a 12GB multilingual corpus, obtained by Language Identification and filtering of the CommonCrawl corpus, a huge dataset containing plain text extracts of web pages written in a large variety of languages and covering all possible types of topics [Ortiz Suárez et al. 2020]. Pre-processing for RobBERT made sure that sequences start and end in complete sentences with a maximum length of 512 tokens.

Another model that performs well on Dutch text, but is also proficient in 103 other languages, is Multilingual BERT (M-BERT). In Table 2.5, M-BERT is shown to reach the highest score on Named Entity Recognition and Part-of-speech tagging, scoring higher than Dutch-only models. M-BERT is a BERT-model trained on Wikipedia articles, written in the 104 languages that are used the most on Wikipedia, including Dutch, English and Arabic (not specified which dialect). These Wikipedia articles contain relatively formal texts, with dialectal text and other varieties, such as multi-ethnolects largely absent. The model uses a shared vocabulary, with no markers denoting which language a word is from.

Notably, languages with large wikipedias were downsampled whereas languages with smaller wikipedias were upsampled. M-BERT was not trained on Berber or any other languages that are the origin of certain Straattaal vocabulary. However, Pires et al. [2019] found that M-BERT, although exclusively trained on monolingual corpora, has the ability to transfer between languages that are written in different scripts, thus proving useful for multilingual objectives. Interestingly, M-BERT performed better on Part-of-speech categorization on a code-switched dataset, than on a monolingual one [Pires et al. 2019]. It performed POS on two datasets with code-switched Hindi and English. One where both languages appeared in the Latin script, and one where Hindi was written in their original Devanagari script. The second time M-BERT performed the POS task almost twice as well.

3 Annotating and Analyzing the Moroccorp

This chapter is structured in the following way: The original, unlabeled Moroccorp is discussed in Section 3.1. Then, I present the custom labels I use in Section 3.2.1. In Section 3.2.2, I discuss details on the procedure of annotating. Interesting findings and difficulties during annotation are elaborated on in Section 3.2.3. Finally, the annotated dataset is evaluated and analyzed in Section 3.3.

3.1 Dataset

I use the Moroccorp, a large text corpus that contains over 10 million words, taken from *maroc.nl*, a Dutch forum-like chat website [Ruette and Van de Velde 2013]. Whereas there is no certainty about the ethnicity of the forum users, there is reason to assume that the majority is affiliated with Morocco, which I discussed in Section 2.2.5.

The dataset was created and made public to encourage sociolinguistic research, and in particular, the consequences of language contact between Dutch and Moroccan Arabic and/or Berber. I have not been able to find other publicly available corpora that contain Moroccan Dutch, unless they only contain spoken Dutch, instead of written, like the Dutch Bilingual Database [Boumans and Crevels 2005].

The authors created the corpus by logging the chat directly from the website. Logging was not done continuously. This was both for technical reasons, as well as to remain unnoticed by the present chatters, as logging could solely be done 'online', visible to every present user. The conversations are from two separate channels, #maroc and #maroc.nl, each accessible from a different website. The conversations in the corpus were logged during the summer of 2012, the first channel in the first two months, and the second channel in the last two months of the in total four month period of logging.

Initially, the scraped corpus contained 24 million words, before Ruette et al. cleaned up the dataset and reduced this to about 10 million words. During the cleaning they removed mostly functional messages that were written by bots, such as local prayer times. The time at which a message was sent was also deleted, leaving lines of the following format: '<author> message'
Like in (21):

Metric	**Moroccorp**
Number of Words	10M words
Vocabulary Size	200k unique words
Number of Lines	2.033.927 lines
Average Length of a Line	6,81 words per line
Unique Lines	1.788.273 unique sentences
Number of Unique Users	30.156 users

Table 3.1 Size of the Moroccorp

(21) ‘<Thee-drinker> Pff.. Wat een bruiloft was dat’
‘<Tea-drinker> Pff.. What a wedding that was’

To illustrate the size of the Moroccorp, I display various metrics in Table 3.1. Notably, the number of lines in the Moroccorp does not correspond completely with the linguistic concept of sentences, as it frequently happens that chat users send a message before they finish a sentence, see (22). Users can also type multiple sentences in the same line, although that occurs less frequently:

(22) ”<badmeester> vier”
”<badmeester> 5”
”<badmeester> mag ook”
”<badmeester> met chillie saus”

3.2 Annotation

This section covers the process of annotating the dataset, from the selection of labels for the classification task in Section 3.2.1, to the process of sampling and decision making in Section 3.2.2 and interesting, ambiguous cases in Section 3.2.3.

3.2.1 Selecting Classification Labels

In Section 2.3.2, I discussed differences in levels of granularity for annotation of multilingual text, for example sentence-level or word-level annotation. Researchers also differ in number and content of unique labels for the Language Identification (LID) task. I made an overview of the most widely used labels for LID in Table 3.2. Most researchers use labels for each language that appears regularly in the data, in the form *lang1, lang2*, as well as labels for language independent utterances such as named entities *NE (named entity)* and tokens like ‘@’, ‘:)’ and ‘!!!’, although each approach seems to use a different label name for this: *other,*

Label names	Description	Examples
lang1, English, EN	Words in English	‘text’, ‘book’, ‘example’
lang2, Spanish, SPA	Words in Spanish	‘ejemplo’, ‘libro’, ‘texto’
Named entity, NE	Named entities	‘John’, ‘Facebook’, ‘Chicago’
Other, Univ, None	Language independent utterances	‘:)’, ‘hahahha’, ‘!!!’

Table 3.2 Overview of most common LID labels for Spanish-English code-switched data

univ, none [Aguilar et al. 2020, Das and Gambäck 2014, Maharjan et al. 2015, Solorio et al. 2014].

The labels *lang1, lang2* and *NE* see little controversy or ambiguity, but the label *other* is also used to describe words in a language other than the ones focused on [Mellado and Lignos 2022], whereas some studies separate this category in the form of ‘Foreign word (Fw)’ [Aguilar et al. 2020].

The labels I used for my annotation are showcased in Table 3.3 and partially overlap with these widely recognized labels from Table 3.2. Specifically, lang1 corresponds to NL and lang2 to ENG. I also introduced a third label, MOR, which can be considered a third language. NE corresponds to NAME. The common label Other from Table 3.2 corresponds with my label NON in Table 3.3. Additionally, I will also use a ‘Foreign word’ label, denoted as OTH, following the example of Aguilar et al. [2020]. Language independent utterances will be referred to as NON.

Lastly, I have introduced a novel and distinct label denoted as VAR. Given the source of the dataset I am annotating, a Moroccan-Dutch internet forum, I have reason to expect use of the Moroccan-Dutch ethnolect in the text, which is partly characterized by the use of phonetic misspellings and grammatical mistakes like wrong inflections of adjectives, as I stated in Section 2.2. VAR is added to capture spelling variations of Dutch. This is interesting, because it quantifies the amount of spelling and grammatical variations in this dataset and it may help recognizing the Moroccan-Dutch ethnolect.

Subsequently, each label will be explained in detail.

- NL, for tokens that are well formed Dutch, without any grammatical or spelling errors. If an ambiguous word appeared that could be labeled as either Dutch or something else, I always chose Dutch. To decide whether a word with roots in another language, such as English or Sranan Tongo, qualified as Dutch, I used online dictionary *van Dale*[1], like was done similarly in Mellado and Lignos [2022]. Words that are not in the dictionary, I would not annotate as NL, but as ENG or OTH, depending on the word. Words unknown to me I would first search on the Internet.

[1] https://www.vandale.nl/

Label	Short Description	Example
NL	Well formed Dutch words	Ik ben nog lang niet jarig
ENG	English words	You are so cute.
MOR	Moroccan Arabic or Berber words	Hmdl ieshem a ochti
VAR	Spelling variations and ill-formed Dutch	jaaa iuk oooook
OTH	Words from other languages	aah tu parle francais?
NON	Emoji's, human sounds etc.	hahaha :)
NAME	Named entities	catherine zeta jones

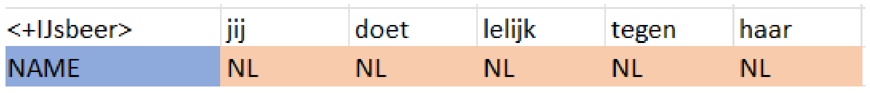
Table 3.3 The classification labels for my annotation of the Moroccorp. The abbreviations stand for Dutch, English, Moroccan languages, Variation of Dutch, Other, None and Name, respectively.

- ENG, for tokens that are written in English. Words that are borrowed so often that they have become part of the Dutch lexicon, are not included and labeled NL instead. To decide to which category a word belongs, I used *van Dale*, like was done similarly in Mellado and Lignos [2022].
- MOR for tokens that are Moroccan Arabic or Berber. As I do not speak Arabic or Berber and am unqualified to spot the difference, I haven't created separate labels for the two Moroccan languages, but treat them as a single category instead.
- VAR, for words that are Dutch, but contain at least one spelling variation, and for words that are used in a wrong way, either grammatically or semantically. This includes letter repetitions, ellipses and grammatical mistakes.
- OTH, for words from languages that I did not expect to find, like French, German or Turkish.
- NON, for language independent utterances or words that cannot be put in any of the established categories. For instance, utterances that only convey an act of emotion, like an emoticon, e.g. ':D', or sounds in text form, like laughter, e.g. 'hahaha', 'pff'.
- NAME, for the usernames of the forum users or references to these usernames, as well as other named entities, like 'Johan Cruijf', or 'Ajax'.

3.2.2 Annotation Process

I annotated in two batches. For the first batch I used Doccano[2], a tool that facilitated the annotation process by providing a graphic interface [Nakayama et al. 2018], for the second I used Microsoft Excel, using a tile for each word and label, as shown in Figure 3.1. Both times

[2] https://github.com/doccano/doccano



<+IJsbeer>	jij	doet	lelijk	tegen	haar
NAME	NL	NL	NL	NL	NL

Figure 3.1 Annotating in Microsoft Excel for the second batch, making use of shortkeys (1=NL, 2=NAME, etc.) and different highlights for each label to make the annotation clear and efficient.

	Sample length	Amount of samples	Total lines after filtering
First batch	200	17	3312
Second batch	25	100	2068
Total	-	-	5380

Table 3.4 Size of two annotated samples of the Moroccorp for annotation

I annotated a randomly generated subset of the Moroccorp. For the first batch, I randomly sampled a line in the dataset and selected 200 lines starting from that line. I repeated this process 17 times, resulting in 3400 lines. For the second batch, I sampled in a similar way, but decided to use 100 shorter samples of 25 lines each, resulting in 2500 lines. During both batches of annotation, I did not completely randomize the lines from the corpus to preserve the conversation-like nature of the text, so I was able to use contextual clues to label difficult or ambiguous words. See Table 3.4, for an overview of the size and samples of the two batches.

During the first batch, I initially wanted to preserve the conversational nature of the text by sampling a relatively long piece of the conversation. Yet, I found that some forum users are dominant and overrepresented in certain parts of the corpus. To increase the amount of data and potentially improve model performance I decided to annotate a second batch of lines. This also served to diversify the amount of different speakers in the dataset. This second batch of the dataset is more diverse than the first batch, as I took shorter samples from multiple parts of the large corpus.

Despite the random sampling, several lines appeared twice in the sample. It also occurred that a spam user would repeat the same message continuously. After deleting these duplicate sentences, the dataset contained a total of 5380 lines.

I am annotating each word, counting a whitespace as a separator. For each word I annotated which label is most accurate, without ever assigning multiple labels to the same word. For an example, see (23).

(23) ”[’Nabila:’, ’alikoem’, ’salleem’]”
”[’NAME’, ’MOR’, ’MOR’]”

During the first batch of annotation, I initially used three extra separate labels. One for lexical borrowings from English (Borrow), one for words that are intra-word switched (Mixed), as well as a label for Straattaal. After going through the first half of the annotation process, I found that the labels Straattaal (101 occurrences), Borrow (135 occurrences) and Mixed (8 occurrences) were not only very scarce, but also difficult to distinguish from other labels. For the Borrowed category I created a definition, but for many words that fall into that category it seems unnatural to label the words as such, because the words in question are so assimilated into the Dutch language that they are simply Dutch. Most of the words I annotated as Straattaal could also be annotated as Moroccan, English or VAR, except for certain words that originate from Sranan Tongo, which appear infrequently.

I re-evaluated the words from each of these three categories. Borrow words were changed to either NL, VAR or ENG, depending on their existence in the Dutch dictionary and spelling, ensuring my decision aligned with the decision process from Section 3.2.2. Most words that I had annotated as Straattaal could become NL or VAR. The rest of the words I annotated as either MOR or OTH, depending on the origin of the word. Mixed only had eight words to re-evaluate, and most became VAR.

At the start of the first batch, I first annotated about a 100 lines and subsequently asked two fellow students to annotate the same 100 sentences for me. I did this to make sure my annotations stayed consistent. For each disagreement I discussed with the volunteers as well as my supervisors to make a plan for future instances of such cases, so that I can be as consistent as possible.

Most of the disagreements happened with words that could fall into more than one category. For instance, words that both expressed a sound of emotion (NON), but were also spelled in a non-standard way (VAR), like ‘hhh’ (laughter). I then decided to use VAR only for words that would be NL if spelled right, therefore making all language independent utterances, no matter their spelling: NON. Another example, some words can refer to the name of a forum user, but are also a perfectly normal Dutch word, like ‘badmeester’ (lifeguard). I opted to always choose the category that fitted the context best. So, for ‘badmeester’, I would choose NAME, as it was used in the context of a username, and not about an actual lifeguard.

After analysis of the disagreements I created the following decision process: Can I label the word immediately without more information? If yes, I would. If I needed more information, I would first look in the dataset if it was a reference to a username or if it appeared more than once. This was mostly applicable to words from Moroccan languages, due to my lack of knowledge. If the word proved to be a reference, then I would assign NAME. If I still did not know the word I would search the internet and assign the label I found the word to be. For instance, if the search results would be only other Moroccan websites, I would assume the word is Moroccan, assign MOR and do the same when the word occurred again.

For the words I did know but had doubts whether a word was spelled right or belonged in the Dutch language I searched the word in Van Dale. If the word could be found there, then I always assigned NL. For other words, I would then determine if the word is English, VAR or Other.

Although this decision process worked well in general, it would be challenged as I came across ambiguous cases, as I discuss in Section 3.2.3.

After annotating the second batch, I again asked a fellow student to annotate 100 sentences and compare the annotations to evaluate them and reach inter annotator agreement. I do this mathematically with Cohen's Kappa, of which the equation is given in Equation (3.1).

$$\mathrm{K} = \frac{p_o - p_e}{1 - p_e} = 0,7028 \tag{3.1}$$

In which p_o is the observed agreement and p_e is the agreement which would occur by chance. The closer kappa is to 1, the higher the agreement. I calculated this in Python using the cohenskappa function from sklearn. In total, 553 words were compared in this calculation. As shown in the equation, the agreement was 0,7028.

This score indicates substantial agreement beyond chance in the classification task. It suggests a reasonably strong level of reliability between annotators or models. Generally, a kappa value above 0,6 represents an adequate annotator agreement [Nowak and Rüger 2010].

As the student who also annotated had no knowledge from Arabic or Berber, annotating those went wrong the most. Also they missed some Names. There were also a few occasions on which it was me who made a mistake and the annotation by the other person was the one it should have been according to my own rules.

3.2.3 Difficulties and Ambiguities

During the annotation, I encountered various ambiguous cases that I deemed interesting to report upon. I will first discuss ambiguous cases per label that I eventually assigned. Most of these are about VAR, but also about MOR and OTH. Then, I will describe some miscellaneous ambiguities.

3.2.3.1 VAR

I found VAR to be relatively difficult to assign during annotation. To be categorized VAR, the token must be a spelling variation of a recognizable Dutch word. So, if an English word is spelled wrong, it counts as ENG, not VAR. I define spelling variations as words that are spelled differently from its standard dictionary form, like in (24). Also, when a correctly spelled word is used but it is likely that another word was meant to be used, I annotate it as 'VAR', like in (25). As in the context of this message the chat user was trying to convince another chat user that they were incorrect about something, it is likely that they meant 'verkeerd' (wrong), instead of 'verkeer' (traffic).

(24) hallllOOOO allemaaal
VAR VAR
hello everybody

(25) je ziet *verkeer*
NL NL VAR
you see traffic

Although spelling mistakes were made often enough and were usually easily identified, grammar mistakes appeared frequently as well. However, labeling them proved to be more difficult. For example, the nature of many grammar mistakes was the absence of certain function words, like in (26) which lacks the article 'een', as the context of this message was to offer someone a drink. Without the article, the offer is one of the material glass, not the object one drinks from. The words that were present and up for labeling had no mistake themselves, and were thus labeled as correct Dutch. As I chose to label each word individually, the grammar mistakes caused by omission of words were not caught in the annotation.

(26) indo hier heb je glas
NAME NL NL NL NL
indo, here is a glass

Grammar mistakes caused by the usage of a wrong word or an unnecessary word were labeled as VAR. Wrongly inflected adjectives, like mentioned in Section 3.1, were found occasionally, see (27, 28).

(27) volgens mij heb je een *dubbele* gesprek
NL NL NL NL NL VAR NL
according to me you have a double conversation

(28) *klein* kinderen zijn groot achter de pc
VAR NL NL NL NL NL NL
small children are big behind the pc

Sometimes the mistake was not grammatical, but idiomatic. For instance in (29), the expression should be 'doe wat je niet laten kan' (do what you can't resist), but the user used the word for 'later', providing a (given the context suspected to be unintentional) variation on an already existing idiom. The same happens in (30), in which the prefix 'voor' is not found in the original idiom, which means to scold someone. However, I judged this mistake to be too close to the original idiom, semantically, so I annotated the word as NL.

(29) doe wat je niet *later* kan
NL NL NL NL VAR NL
do what you can't do later

(30) de les *voor*lezen
NL NL NL
reading the lesson *aloud*

3.2.3.2 MOR

The MOR (Moroccan) label stands out from the other language labels NL (Dutch) and ENG (English), as it doesn't encompass words from a single language, but words in this category from one of two languages: either Moroccan Arabic or Berber. Unlike Dutch and English, for which accessible dictionary tools are available, I did not find such resources for these two languages, possibly because I can read neither the Arabic script nor the Berber orthographies. Therefore, I sought confirmation from a native Berber speaker who also has knowledge of Moroccan Arabic. I provided them with a list of tokens that I had assigned MOR and appeared more than once in the dataset. They were able to translate at least 2/3 of the words and confirmed they were indeed from either Moroccan language, sometimes specifying which one. As the words in the list appeared separately and without context, not all words could be easily translated. They did not flag any words to be clearly not Moroccan. However, as they only looked at tokens that appeared more than once, the possibility that my annotations contain mistakes remains, more on this in Section 5.2.

Words like muslim, islam are spelled in a certain way in Dutch, but as these are essentially loanwords from Arabic, the words are often spelled in various ways: 'moslim' or 'muslim', 'islam', 'islaam', or with article 'de islam'. In Dutch, spelling is standardized as 'moslim' and 'de islam', as a noun with a definite article and not without it. Each occurrence of these words that differ from these standard spellings I annotated as 'VAR', unless used in a Moroccan Arabic context, then I labeled them 'MOR'.

3.2.3.3 OTH

Words that fit neither the established language categories (NL, VAR, ENG, MOR) nor language independent categories (NAME, NON), I annotated as 'Other' (OTH), like in (31). I annotated common internet acronyms, like 'lol' and 'omg' as OTH, as these tokens transcend monolingual language use, and might be used in Dutch, English or another language, such as Turkish in (32). I contemplated whether NON would be the correct label, but I figured that the acronyms are not completely language independent, making OTH the more logical choice.

(31) Je vous souhaite une bonne soirée!!!
OTH OTH OTH OTH OTH OTH
[French] I wish you a good night!!!

(32) askimmm seniii seviyorum lol
OTH OTH OTH OTH
[Turkish] my lovee I love youuu lol

Several other loanwords, originating in English or other languages, like Sranan Tongo, have been admitted to the official Dutch dictionary, van Dale. I chose to label each loanword that was spelled in the way it appeared in van Dale, as NL. Initially I had created a label for 'Straattaal', but it became too unclear which words did belong to this category and which did not, as it is not an official language, but rather a way of speaking, taking vocabulary from a mix of languages. Words that are used in Straattaal that were not found in the van Dale dictionary, I labeled as 'Other', as these words would then originate in Sranan Tongo or another root language for Straattaal words. If this root language is Arabic, Berber or English, I assign MOR or ENG, respectively.

3.2.3.4 Encoding errors

Some users would consistently write in a way that may indicate an encoding problem with the scraping of the forum, like in the following example:

(33) 3e2en 3b2anaan 3u2it 3j2e 3t2as 3h2alen
VAR VAR VAR VAR VAR VAR
take a banana from your bag

One forum user writes in Dutch, mostly without spelling errors, but the first letter of each word they type is encompassed by a 3 and a 2, thus creating a typo for each word written. Therefore, I labeled each otherwise correct Dutch word written by this user as VAR.

As I split the sentences on whitespaces, various typos by the forum users unintentionally created new words, where instead of a whitespace, one or more periods or commas's ('.' or ',') meant to separate two words. I annotated them all as VAR because of this, but some are actually two different categories, like 'name...dutch' as in (34). Conversely, some words are split into two or more words because of the extensive use of whitespaces, such as in (35). This seems to happen intentionally for certain swear words, such that the user may not be flagged as such and banned from the forum, but for other words it happens as well.

(34) hicham32...moeders
VAR
hicham32...mothers

(35)

Ikroep	je	en	je	zei	zo	..	h	o	m	o
VAR	NL	NL	NL	NL	NL	NON	VAR	VAR	VAR	VAR

I call you and you said like .. g a y

3.2.4 Postprocessing

After annotating each line the annotation process was not finished, as some lines needed to be revised.[3] In addition to that, I found inconsistencies in my own work that I was able to fix by filtering the dataset on words that receive different labels on different occasions, and revising the labels for these tokens. I found about 500 words for which this was the case. For example, I changed definition of OTH after the first batch, by including internet acronyms, like 'omg', but before, I had annotated them as VAR, or even ENG if they appeared in an English sentence. To make the annotations consistent, I changed them all to OTH. For some tokens it is justified to receive different labels at different occurrences, as the correct label for a word is context-dependent. For example, the words 'in' and 'me' can be either English or Dutch, depending on its context. The latter even occurs as a spelling error when used as a possessive pronoun and therefore can be VAR. 220 words still receive a different label at different occurrences. Many of these are words that can be correct in one context but incorrect in another, making either NL and VAR the assigned label, depending on the context. Also many NAME words are usernames that originate from actual Dutch or English words, which are sometimes used outside of the username context, and therefore assigned NL, VAR or ENG. After identifying the shifts in the label lists and revising these duplicate words, I made sure that each line contained the same amount of tokens and labels.

Lastly, I have removed the nicknames that each message starts with in the corpus, as they make the label NAME too predictable for a model, and do not add anything to the recognition of code-switching or the Moroccan-Dutch ethnolect. Also, removing the names benefits anonymization of the data.

In the next section I will present the results of the annotation.

3.3 Data Analysis

This section has three parts. First, I discuss the general size of the annotated dataset along with the frequency of each label in Section 3.3.1. Then, in Section 3.3.2, I investigate in what share of the data code-switching takes place. Lastly, in Section 3.3.3, I calculate metrics used

[3] Both annotation tools export the annotated dataset in a different format, so these needed to be matched. The exportation also resulted in some words erroneously receiving no label at all, or receiving labels meant for words next to them, due to a shift in the label list.

Metric	**Annotated Moroccorp**
Number of Words	26.947 words
Vocabulary Size	8.032 unique words
Number of Sentences (lines)	5380 lines
Average Sentence Length	5,30 words per line
Unique Sentences	4935 unique sentences
Number of Unique Users	636 unique users

Table 3.5 Size of the Annotated Moroccorp

Label	Total Words	Percentage of Total Words	Unique Instances	Percentage of Unique Instances
NL	19.305	71,64%	3.979	20,61%
NAME	2.116	7,85%	992	46,88%
VAR	1.782	6,61%	1.246	69,92%
MOR	1.649	6,11%	1.085	65,80%
NON	1.158	4,30%	419	36,18%
ENG	731	2,71%	416	56,91%
OTH	206	0,76%	153	74,27%
Total	26.947	100%	8290	30,76%

Table 3.6 Total amount of words per label in annotated part of Moroccorp, as well as the amount of unique instances per label and both of their relative amounts. Unique instances are calculated per label, and the total amount is their sum.

for measuring code-switching, to be able to compare the amount of code-switching in the Moroccorp with other datasets.

3.3.1 The Labeled Dataset

I labeled the dataset[4]. Table 3.5 illustrates the size of the annotated dataset in various general metrics. The distribution of labels in the annotated dataset on word level is depicted in Table 3.6.

The vast majority of words in the corpus (71,64%) are correct Dutch (NL). 6,6% of the dataset is labeled as VAR. As VAR is defined as misspellings or otherwise incorrect variations of Dutch, I can take the sum of NL and VAR to calculate a total amount of Dutch words (21.087), which thus makes up 78,3% of the dataset. Then, I can calculate how many of the Dutch words in the dataset is an incorrect variation, see Equation (3.2).

$$\frac{1.782}{21.087} \cdot 100\% = 8,45\% \tag{3.2}$$

[4] https://huggingface.co/datasets/Tommert25/extradata0908/tree/main

Almost 8,5% of Dutch words in the dataset appear in a way that is incorrect and are therefore VAR.

About 12% of words in the dataset are NAME or NON, and therefore language independent. Moroccan languages (MOR) make up over 6 percent of the dataset. If I add the other two non-Dutch language labels ENG and OTH to that number, the share of languages other than Dutch in the dataset reaches just under 10 percent (9,58%).

The Moroccan ethnolect is characterized by insertions of Arabic or Berber function words, as well as deliberate misspellings (rebellion spelling) and wrong inflections of Dutch adjectives. This is roughly captured by the combination of MOR and VAR, making almost 13% of the dataset part of the Moroccan ethnolect. Not all VAR words are typical instances of rebellion spelling, as it is difficult to assert when a misspelling is deliberate. Many VAR words are typical cases of letter repetition, showing excitement in computer-mediated communication, and wrong inflections of Dutch adjectives are also frequently found in the sample.

I also calculated how many unique words occur in each category, as well as how large the share of unique words is compared to the total amount of words in a category. This is relevant, as variety within a category affects a classification model's ability to generalize and accurately classify diverse instances of a given category. As seen in Table 3.6, about 30% of the total amount of words are unique instances in this dataset. If I examine each label individually, there are notable variations. For four labels this amount is much higher: MOR, VAR, ENG and OTH. For ENG and OTH this is explainable as both are relatively infrequent, making each new occurrence likely to be a unique instance. The VAR category is inherently highly variant and expected to contain a high number of unique instances. Interestingly, there are percentually almost as many unique tokens from Moroccan languages as there are from the VAR category. It can be explained, as just like VAR, there is no set way of spelling Arabic or Berber words in the Roman alphabet. As an illustration, the Arabic greeting 'salam alaykum' appears in the labeled dataset in more than five variations of spelling.

The total amount of unique instances in the dataset in Table 3.6 does not match the vocabulary size of the total dataset from Table 3.5. However, this can be explained. I calculated the number of unique instances in Table 3.6 by taking the sum of the unique instances per label, but there are 258 words that are found in multiple categories and therefore counted more than once as a unique instance. Like I discussed in Section 3.2.4, this holds because the same words can be used in different ways, depending on context. For example, 'jonge', which is used as correct Dutch (NL), but also as an incorrect inflection (VAR), therefore counted twice as a unique token. Another example is the username 'koe', which counts as NAME, but 'koe' is also the Dutch word for 'cow', counting as NL.

3.3.2 Quantifying Code-Switching in the Annotated Corpus

Out of the two main different types of code-switching I introduced in Section 2.1, I focus on intra-sentential code-switching, as my labels provide a good framework to quantify this and

I am interested in examining code-switching within shorter linguistic units, which are most of the instances in the dataset. The chat messages in the dataset can be sent with an unknown amount of time between them, leaving ambiguous whether the text is of a conversation-like nature. I could ignore this, and define an intersentential code-switch if there is a new matrix language or if there is a new label present in a following sentence. However, I leave this for future work. An intersentential code-switch would also occur when multiple sentences are sent within one message, but messages in chat logs are generally short, making this unlikely to occur frequently.

I want to know how much intra-sentential code-switching occurs in the annotated dataset. This only happens when at least one word in the sentence receives a different unique label than another. I exclude the labels NAME and NON from being counted as a unique label, as these labels are largely language-independent. The VAR label is language related, as it is used only for words that are supposed to be Dutch, but are used incorrectly, due to misspelling or another linguistic error. Although it is a relevant label because the Moroccan-Dutch ethnolect is characterized by frequent mistakes like this, switching from NL to VAR is not code-switching, for there is no switch to another language present. I also don't count the VAR labels as a unique label for the code-switching subset, but include them as if they were NL. Switching from VAR to MOR, for example, does count as code-switching.

I speak of intra-sentential code-switching when at least one word in the sentence receives a label with a different *language* than another, counting VAR as a part of NL and excluding the language independent labels NAME and NON.

I calculate in how many lines of the corpus there is only one unique label present. There are 2709 mono-labeled lines in the annotated dataset: lines with only one unique label. This is roughly half of the dataset. Inversely, this also means that the other half of the dataset consists of lines with two or more unique labels. Then, only lines remain that include code-switching between different languages NL/VAR, ENG, MOR and OTH. In 527 lines in the annotated dataset, there is an occurrence of code-switching. That is 9,80% of the total amount of data. In Table 3.7, I made an overview of my calculations. Then, in Table 3.8, I provide insight as to how much each label occurs with each other in a line.

I found several clear examples of intra-sentential code-switching in the dataset. 324 lines for code-switching between Dutch and Arabic (36), 114 lines for Dutch-English (37) and 24 lines in which English-Arabic code-switching occurred (38).

(36) Ik ben *shab taghennant.*
NL NL MOR MOR

(37) Ilovebeing Het was slechts mij liefde voor Islam die meteen in oprukken kwam
NAME NL NL NL VAR NL NL NL NL NL NL NL NL
So don't blame me.
ENG ENG ENG ENG

Set of sentences	Amount of lines	Percentage
Total	5380	100%
Mono-labeled lines	2709	50,4%
Code-switching only	527	9,80%

Table 3.7 Amount of lines in the annotated corpus with different levels of plurality. Lines with only one unique label in Mono-labeled lines, compared to amount of lines that contain code-switching: more than one unique language label in the annotated corpus, excluding NAME and NON, and counting each VAR label as if it was NL.

Label combinations	NL	ENG	MOR	VAR	OTH
NL	1537	114	324	933	58
ENG	-	94	24	37	6
MOR	-	-	194	89	12
VAR	-	-	-	196	12
OTH	-	-	-	-	59

Table 3.8 Combinations of labels lines in the annotated Moroccorp. The numbers on the diagonal are the number of monolingual lines in which only one label occurred. The other numbers are the amount of lines for each combination.

(38) Salaam *again!!*
MOR ENG

I checked the occurrence for each combination of two labels within one line. Each label occurs with every other label at least once, as depicted in Table 3.8. The numbers on the diagonal show how many lines were monolingual in each category. Out of all different combinations of languages, Dutch-Arabic occurred the most within the same sentence. The low number of Dutch-English combinations surprised me. Although a lot of words with English origins are also used in the corpus, most of these are assimilated in the Dutch language to the point that they have an official dictionary entry, and therefore simply counted as NL.

3.3.3 Code-switching metrics

To measure how much code-switching occurs in a dataset, the Code Mixing Index (CMI) was introduced, a formula that measures how many words in a document do not belong to the most frequent language (*the matrix language*) [Das and Gambäck 2014, Gambäck 2014]. It finds the matrix language, then counts the percentage of words belonging to all other languages present, excluding language independent utterances.

Label	Amount	Percentage	CS-only	CS-only percentage
NL+VAR	21.087	89,1%	2.978	76,7%
MOR	1.649	7,0%	607	15,6%
ENG	731	3,1%	214	5,5%
OTH	206	0,9%	82	2,1%
Total	23.673	100%	3.881	100%

Table 3.9 Total amount and percentages of words, excluding language independent labels and taking NL and VAR together as one label. First for the total annotated dataset, then for only lines with code-switching (CS-only).

$$CMI = 100 \cdot \frac{\sum_{i=1}^{N}(w_i) - \max(w)}{n-u} \tag{3.3}$$

The formula for CMI is shown in Equation (3.3), where $\sum_{i=1}^{N}(w_i)$ is the sum over all N languages present in the utterance of their respective number of words, $\max(w)$ is the number of words present from the language with the highest word count, n is the total number of words, and u is the number of words given language independent tags.

Because the sum of all language-related words is the same value as $n-u$ we can also write the formula as:

$$CMI = \begin{cases} 100 \cdot \left(1 - \frac{\max(w)}{n-u}\right) & \text{if } n > u \\ 0 & \text{if } n = u \end{cases} \tag{3.4}$$

The higher the CMI, the more multilingual a dataset is, making it plausible to also contain more code-switching. The index provides insight into the extent to which multiple languages coexist within the dataset.

In Table 3.10, I show several CMI scores for other datasets on which Language detection was performed. CMI indices on datasets of the LinCE benchmark [Aguilar et al. 2020] vary from 2,82 (Modern Standard Arabic-Egyptian Arabic) to 19,85 (Nepali-English).

In the following part of the section, I calculate the CMI for the Moroccorp, of which the results are showcased in Table 3.12. If I compare these CMI indices to those found in the datasets used by Aguilar et al. [2020], then I find that on document-level these numbers are most similar to the CMI scores from Mave et al. [2018], and seem to fit in well with other datasets used for language identification.

Using the numbers from Table 3.9, where I depict the amount of each relevant label, excluding the NAME and NON labels, I calculate CMI on document level as follows:

Author and Corpus languages	CMI (document-level)	CMI (CS only)
Molina et al. [2016]: SPA-EN	8,29	21,86
Solorio et al. [2014]: NEP-EN	19,85	25,75
Mave et al. [2018]: HIN-EN	10,14	22,68
Molina et al. [2016]: MSA - EA	2,82	23,89
Kalkman [2023]: MOR - NL	10,92	23,27

Table 3.10 Comparing Code Mixing Index for the Moroccorp with other datasets. The left column depicts the CMI for the entire corpus, and the right the CMI for the subset of sentences in the corpus that contain code-switching (CS). The acronyms for languages are EN: English, SPA: Spanish, NEP: Nepali, HIN: Hindi, MSA: Modern Standard Arabic, EA: Egyptian Arabic, MOR: Moroccan Languages, NL: Dutch.

$$CMI = 100 \cdot \frac{(MOR + ENG + OTH + (NL + VAR)) - (NL + VAR)}{TOTAL - NON - NAME} = 10,92 \tag{3.5}$$

The matrix language is NL, and there are four languages present, as well as two language independent tags. The tricky part here is VAR, as it is neither a separate language, nor language independent. Just like in Table 3.7, I counted the VAR words as a part of NL, the matrix language, resulting in a CMI of 10,92, as shown in Equation (3.5).

The Code-mixing index can also be calculated at the utterance level [Gambäck and Das 2016]. It measures how many words do not belong to the matrix language, for each individual line. This is more insightful about how much intra-sentential code-switching occurs in the dataset. The higher the CMI, the more alternations the dataset contains, showing the code-switching behavior to be more complex. I use Equation (3.4) for the calculation of this version of CMI, as it has to be calculated for each utterance individually, counting out utterances that only consist of language independent utterances, or where $n = u$.

I start by finding the matrix language in each line of the dataset. This is defined as the most common label in an utterance, and can therefore be different for every utterance. Most of the utterances in the dataset will have NL as the matrix language, but not all. If there are only language independent words ($n = u$) or the matrix language is NON or NAME, the CMI of that line would be 0. For the other lines, I calculated how many words were not part of the matrix language for that line, divided by the total amount of words in that line, while counting out any NON or NAME words. The average Code-mixing index per utterance was 2,627. Just like on document-level, I counted all VAR words as if they were NL. Compared to the datasets from Gambäck and Das [2016], it is relatively low, but not the lowest.

Language Pair	CMI
Moroccan - Dutch	2,63
German - Turkish	4,11
English - Hindi	1,87
English - Spanish	4,91
English - Nepali	7,98

Table 3.11 Average CMI on utterance level for various datasets. The first number is calculated by me for the annotated Moroccorp. The rest are all from Gambäck and Das [2016].

I also calculated both versions of CMI also for the subset of the dataset that contains lines with code-switching only (CS-only), of which the results are depicted in Table 3.12. Unfortunately I could not find utterance-level scores of CMI for the datasets in Table 3.10, but the scores from Gambäck and Das [2016] are depicted in Table 3.11.

Another metric of interest is the Multilingual index (M-index): a word-count based measure quantifying the inequality of distribution of language tags in a corpus of at least two languages, developed by Barnett et al. [2000]. In Equation (3.6), the equation for this metric is depicted.

$$\text{M-Index} = \frac{1 - \sum {p_j}^2}{(k-1) \cdot \sum {p_j}^2} \tag{3.6}$$

In which k is the number of labels represented in the corpus and p_j the share of words that received label j over the total amount of words, with j ranging over the present labels [Guzmán et al. 2017]. An M-index ranges from 0, which indicates a monolingual dataset, to 1, meaning that each language is equally represented.

In [Khanuja et al. 2020], the M-index was calculated for various datasets, two of which are used for LID. FIRE LID, a code-switched Hindi-English dataset [Roy et al. 2013] and EMNLP, a code-switched Spanish-English dataset [Solorio et al. 2014]. These texts have an M-index of 0,39 and 0,33, respectively.

Now, as depicted in Equation (3.7), this value needs to be calculated while excluding the language independent labels. I recalculated the percentages of label occurrences in Table 3.9, and use these numbers in the formula to calculate an M-Index of 0,08343. As with the CMI, I grouped words from VAR and NL together, and use the sum of these labels for further calculations.

$$\text{M-Index} = \frac{1 - ((NL + VAR)^2 + MOR^2 + ENG^2 + OTH^2)}{(4-1) \cdot ((NL + VAR)^2 + MOR^2 + ENG^2 + OTH^2)} = 0,08343 \tag{3.7}$$

Set of sentences	All lines	Only lines with Code-switching
Amount of sentences	5380	510
Percentage	100%	9,48%
CMI (document-level)	10,92	23,27
CMI (utterance-level)	2,627	27,74
M-index	0,08343	0,2077

Table 3.12 Measuring the amount of code-switching in the annotated Moroccorp.

I also calculate the M-index for the share of subset that only contains lines in which code-switching takes place, using the CS-only percentages from Table 3.9. Naturally, there is more code-switching in the subset than in the entire dataset. Both of the M-index values are relatively low, compared to M-indices found in [Khanuja et al. 2020], showing that Dutch is very dominant in the Moroccorp and the present languages are not equally represented. This is also due to the fact that I calculated the index with four languages in mind (VAR+NL, MOR, ENG and OTH), whereas the two datasets from Khanuja et al. [2020] both have only two focus languages. An overview of the metrics I calculated is given in Table 3.12. I calculated each of the metrics for the entire annotated dataset, and also for the code-switching subset.

In conclusion, the degree of code-switching in the Moroccorp is comparable to datasets used in other code-switching research, like in Aguilar et al. [2020]. Therefore, it can be assumed that the Moroccorp holds utility for further code-switching research.

4 Fine-tuning Classification Models

Using the annotated dataset from the previous chapter, I fine-tuned two pre-trained deep learning models on the word-level Language Identification task, which takes the form of multiclass token classification. For a detailed explanation of fine-tuning a pre-trained model for token classification, see Section 2.4. Both models use the transformer architecture. The first pre-trained model I fine-tune is RobBERT [Delobelle et al. 2020], as it is the state-of-the art Dutch RoBERTa model, as stated in Section 2.4. The second deep learning model I use is Multilingual BERT (M-BERT) [Devlin et al. 2019, Pires et al. 2019], which I also discussed in Section 2.4. Thirdly, I compare the two deep learning models to a logistic regression baseline.

This chapter outlines the steps I took to create models that perform token classification. First, the annotated dataset is altered to fit a transformer architecture, which I discuss in Section 4.1. Then, I discuss the models I fine-tune in Section 4.2 and in Section 4.3, I elaborate on their evaluation as well as hyperparameter tuning. Section 4.4 explores training on an augmented dataset. Finally, the results of the models are shown and compared inSection 4.5.

4.1 Preprocessing data

I divide the corpus into a standard training set, a validation set and a final test set, split in 70/15/15 proportions. The validation set is used to determine on which hyperparameters the models perform best. Using the models that perform best on the validation set, a test set is employed to evaluate the trained model's performance on unseen data.

I split the dataset two separate times, once with randomization and once without, thus retaining the conversational nature of the text. Since the BERT models are attention-based and take more than one line or sentence at a time, a batch, for evaluation during training, I hypothesized that a continuation of the conversation without interruption would benefit the models' performance. However, randomizing the dataset also increases diversity per split. To decide which version I would use, I looked at the label distribution per set, see Table 4.1, as well as amount of code-switching, depicted in Table 4.2. In Table 4.1, the distribution of labels is shown in percentages of the total amount of words in the given set, to showcase if the labels are equally distributed or not. The percentages from the randomized split datasets are similar to each other, showing very little differences in distribution, never deviating more than 2% from the distribution in the 'Total' column of Table 4.1. Conversely, the percentages in the

Randomized	Total	Train	Validation	Test
NL	71,64%	71,23%	72,74%	72,41%
NAME	7,85%	8,07%	6,97%	7,73%
VAR	6,61%	6,47%	7,38%	6,50%
MOR	6,11%	6,24%	6,22%	5,44%
NON	4,30%	4,47%	3,49%	4,33 %
ENG	2,71%	2,72%	2,41%	3,00%
OTH	0,76%	0,79%	0,79%	0,59%
Conversational flow	**Total**	**Train**	**Validation**	**Test**
NL	71,64%	73,24%	67,63%	68,98%
NAME	7,85%	7,28%	9,61%	8,61%
VAR	6,61%	7,64%	5,04%	4,23%
MOR	6,11%	4,38%	5,68%	12,22%
NON	4,30%	4,17%	6,64%	3,17%
ENG	2,71%	2,52%	3,98%	2,51%
OTH	0,76%	0,78%	1,43%	0,28%

Table 4.1 The relative distribution of labels per data split based on the total number of words in that split. A comparison between the randomized dataset and the dataset retaining the conversational flow. Percentages in the splits deviating more than two percent from the percentage in the total data, are denoted in magenta.

part of the table representing the non-randomized split datasets, following the conversational flow deviates more than 2% at various occasions. The most extreme difference is the double amount of MOR (12,22%) in the Test set in comparison to the total distribution (6,11%). Such a difference makes the task more difficult on the test set than on the training set.

Additionally, in Table 4.2, I show that the randomized train, validation and test set have a more equal distribution of mono-labeled lines, or lines consisting of words with only one unique label, than the split sets retaining the conversational flow. However, if I only consider lines with code-switching, there was no real difference between the two approaches. Nonetheless, due to the more equal distribution of labels, I choose to use the randomized sets.

In the randomized splits, I determine how many words overlap between the splits. Examining word overlap between the training and test sets is important to assess how well the model generalizes to unseen data. High word overlap might indicate that the model is memorizing specific instances from the training set, potentially leading to overfitting and reduced performance on new examples. Despite being randomized, there will always be a different amount of word overlap between different sets. The amount of word overlap between the Training set and both other sets is showcased in Table 4.3. I also calculated how many unique words are

Conversational flow	Train	Validation	Test
All lines	3766	807	807
Mono-labeled lines	1990 (52,8%)	412 (51,1%)	307 (38,0%)
Code-switching only	373 (9,90%)	69 (8,55%)	85 (10,53%)
Randomized	Train	Validation	Test
All lines	3766	807	807
Mono-labeled lines	1921 (48,99%)	397 (49,19%)	391 (48,45%)
Code-switching only	366 (9,72%)	87 (10,78%)	74 (9,17%)

Table 4.2 Distribution of amount of lines between the splits. Distribution of lines that are labeled with one unique label and of lines that contain code-switching is also shown. A comparison between the randomized dataset and the dataset retaining the conversational flow.

	Train	Validation	Test	Total
Total amount of Words	18735	4149	4063	26947
Overlap with Training set	-	3117 (75,13%)	2982 (73,39%)	2488 (9,23%)
Unique Words	6177	1930	1996	8032
Unique Word-label pairs	6352	1978	2035	8290
Overlap with Training set	-	989 (50,00%)	1020 (50,12%)	499 (6,02%)

Table 4.3 Word overlap between training, validation and test set. The numbers depicting the overlap in the 'Total' column denote how many (unique) words appear in all three sets.

present in each set, how many unique word-label pairs and how much they overlap between the splits. As shown in the 'Total' column, 499 unique word-label pairs appeared in all three sets. Both the validation and test set have 50% overlap with the unique word-label pairs of the training set. This may be a consequence of the randomization of the corpus, which put different parts of the same conversation in different subsets.

As I annotated the dataset on word-level, split on whitespaces, it is already tokenized to some degree. However, both M-BERT and RobBERT tokenize on subword-level, so the words and labels were split up and aligned. Here I had a choice to label each subword or label each complete word. I tried both, but decide to adhere to my own annotations, and only have the models label as many words as I had done. This also makes it feasible to compare the two transformer models to each other, as well as to the baseline model, as different tokenizers create a different amount of subwords, making a comparison between evaluation scores unfair. Although the models do not receive explicit information from certain subwords, they are nonetheless processed and used as contextual information.

Model	M-BERT	RobBERT
Architecture	BERT	RoBERTa
Training Task	MLM and NSP	MLM
Data Source	Wikipedia in 104 languages	Dutch part of OSCAR
Size of Training Data	unknown[3]	39GB or 6.6M words
Size	170M Parameters	355M Parameters
Tokenization algorithm	WordPiece	Byte Pair Encoding (BPE)
Vocabulary Size	105.879 tokens	40.000 tokens

Table 4.4 Comparing M-BERT and RobBERT. This is an exact copy of the table in Section 2.4.

4.2 Models

I train three models, two transformer-based deep learning models (RobBERT and M-BERT) and one multinomial logistic regression model as a baseline. The classes I train on are the seven labels introduced in Section 3.2.1: NL, ENG, MOR, VAR, NAME, NON and OTH. The checkpoint I use for RobBERT is called 'pdelobelle/robbert-v2-dutch-base'[1].The checkpoint I use for M-BERT is 'bert-base-multilingual-cased'[2] and my setup is the same as the one for RobBERT. To realize the training of the deep learning models, I will use the Huggingface Trainer API from the transformers library [Wolf et al. 2020], and load each model in with a Token Classification Head.

Before training, the input to the model will be labeled and tokenized using the tokenizer of the specified model (M-BERT or RobBERT). For both transformer models, a pre-trained model is fine-tuned. The two pre-trained models have some key differences, some of which I have specified in Table 4.4, which I also placed in Section 2.4.

4.2.1 The pre-trained models

For fine-tuning both M-BERT and RobBERT, the same code was used, changing only the model checkpoint, and therefore also the Tokenizer. The tokenizer is different in each of the two models. M-BERT's tokenizer is based on WordPiece, whereas RobBERT used Byte Pair Encoding (BPE). Both algorithms are a subword tokenization algorithm, but differ in how they merge subwords to whole words. WordPiece tends to produce subword units that align more closely with complete words, while BPE can generate more flexible subword units that may or may not correspond to complete words.

[1] https://huggingface.co/pdelobelle/robbert-v2-dutch-base

[2] https://huggingface.co/bert-base-multilingual-cased

[3] Not specified in any of the papers regarding M-BERT, and as Wikipedia is highly variable, it is difficult to estimate the size of the training data

Aside from the algorithms, both models have a vastly different vocabulary, both in size and content, because they were trained on different datasets. RobBERT was trained on the Dutch part of the OSCAR corpus. The OSCAR corpus is a 12GB multilingual corpus, obtained by Language Identification and filtering of the CommonCrawl corpus, a huge dataset containing plain text extracts of web pages written in a large variety of languages and covering all possible types of topics [Ortiz Suárez et al. 2020]. M-BERT was trained on Wikipedia articles, written in the 104 languages that are used the most on Wikipedia, including Dutch, English and Arabic (dialect unspecified). Berber languages are not part of the training data for M-BERT. The model uses a shared vocabulary of 105.879 tokens, with no markers denoting which language a word is from. This makes it difficult to say how large the Dutch part of the vocabulary is. Notably, languages with large Wikipedias were downsampled whereas languages with smaller Wikipedias were upsampled. Although this is a larger vocabulary than RobBERT has (40.000 tokens), RobBERT's vocabulary is completely Dutch, whereas the Dutch part of M-BERT's vocabulary is only one out of 104 languages in the vocabulary.

Each model also uses a different range of special tokens. M-BERT uses '[CLS]', '[SEP]', '[PAD]' and starts a token that is not the start of a word with '##', whereas RobBERT uses '<s>', </s> and starts every beginning of a word with 'Ġ'.

The pre-training tasks for the two models are only slightly different. RobBERT uses the RoBERTa architecture [Liu et al. 2019], meaning it was only trained on Masked Language Modeling (MLM) and not on Next Sentence Prediction (NSP). M-BERT was trained on both MLM and NSP. Pre-processing for RobBERT made sure that sequences start and end in complete sentences with a maximum length of 512 tokens. For a more detailed explanation on the pre-training, see Section 2.4.

4.2.2 Setup for the transformer models

To fine-tune the pre-trained models, they are loaded with a classification head. Specifically, I use AutoModelForTokenClassification, from the transformers library. This adds a final layer to the pre-trained neural network. This layer is designed to generate an output for classification as opposed to another NLP task, like question answering. However, the layer is initialized randomly, and will therefore still need to be trained. As also explained in Section 2.4, a classification head is a feedforward neural network layer, that outputs a tensor of logits that has three dimensions. The first two dimensions are equal to the shape of the amount of tokens (10x10 means 10 lines of each 10 tokens). The third, is the shape of the amount of possible labels, which is 7. Concretely, this tensor outputs seven values for each word in the batch. The highest value is the label that will be predicted. The lines of text from the dataset are given to the model in so-called batches. Theoretically, a batch can consist of one line, but typically the batch size will be larger (e.g. 4, 8, 16 or 32). During training, a loss function computes for each batch the distance between the outputs and the target labels. This function first converts the outputs to probabilities and comparing them to the actual labels (that has probability 1 for the correct label and 0 for all others). The objective of training is to minimize this loss. After each batch, gradients are computed to change the weights of the neural network in order to minimize the loss function.

I use CrossEntropyloss, for which I denote the equation in Equation (4.1).

$$L(y,p) = -\sum_i y_i \log(p_i) \tag{4.1}$$

$L(y,p)$: Cross-entropy loss.
y_i: The actual (true) probability of the i-th class (1 for the correct class, 0 for others).
p_i: The predicted probability of the i-th class.

The optimizer then minimizes the loss. For this, I use AdamW [Loshchilov and Hutter 2019].

I trained both models many times, trying various configurations, which I discuss in Section 4.3.1.

4.2.3 Baseline

To provide insight into the effectiveness of sophisticated approaches, compared to more straightforward models for a challenging task like Token Classification, it is valuable to compare results between advanced transformer models and simpler baselines. This comparative analysis helps assess whether the increased complexity and resource requirements of advanced models yield significant improvements in performance over simpler alternatives.

I use Multi-class logistic regression as a baseline, for which the equation is given in Equation (4.2):

$$P(Y=k|X) = \frac{e^{\beta_k \cdot \mathbf{X}}}{\sum_{l=1}^{7} e^{\beta_l \cdot \mathbf{X}}} \tag{4.2}$$

Equation (4.2) calculates the probability that a word falls into category k, given a vector of predictor variables **X**. β_k denotes the coefficients associated with the predictor variables for category k, which are similar to the weights in the transformer models. The numerator represents the exponential of the dot product between β_k and the vector of predictor variables **X**. This term reflects the ”likelihood” of the observation belonging to category k.

The summation in the denominator of the fraction normalizes the probabilities across all categories, ensuring that the probabilities add up to 1.

To classify a word, probabilities are computed for each category, and the category with the highest probability will be the prediction for this word.

To represent the data in a vector **X**, as is needed for this method, I use a one-hot encoder, called CountVectorizer. A one-hot encoder creates a new vector for each word, all the size of the vocabulary, with only a single 1 and the rest zeros. I flatten the dataset to one large list of words, treating each word as an individual data point and omitting the context of the sentence-like structure of the lines. The vectors of predictor variables each stand for a word, not a sentence. A multi-class logistic regression model is then fitted on the vectorized data and their label.

Initially, I tried the python library LangID as a second baseline model. It is a standalone python file, pre-trained on 97 languages. Langid.py utilizes a naive Bayes classifier with a multinomial event model [McCallum and Nigam 1998], which is trained on a combination of byte n-grams (where n ranges from 1 to 4). The training data is taken from JRC-Acquis, ClueWeb 09, Wikipedia, Reuters RCV2 and Debian i18n [Lui and Baldwin 2012]. It works best on document level.

To make LangID work, I had to rename my labels to the labels LangID recognizes, meaning one of the 97 languages it is trained on. This was straightforward for NL and ENG. VAR could count as Dutch and MOR could become Arabic, but NAME, NON, and especially OTH were difficult to rename. LangID reached an accuracy of 25%, even when I reduced the amount of possible categories to only 'NL', 'ENG' and 'AR'(Arabic). It did not once predict Arabic, although there is a label for Arabic in LangID and the Arabic language is present in the dataset. Therefore, I assume that the model is confused by the Romanized Arabic, as the dataset is entirely written in the Roman script.

To mitigate this issue, a model could be trained using code from LangID. However, the initial performance of 25% accuracy was not promising and I decided that a second baseline was not needed for this thesis, continuing with only the multinomial logistic regression model.

4.3 Evaluation and Results

For the evaluation, I use Accuracy, Precision, Recall and F1-score, like in Aguilar et al. [2020], Shakeel et al. [2019].

$$Accuracy = \frac{True}{Total} \tag{4.3}$$

As depicted in Equation (4.3), accuracy is label independent, and is calculated based on how many predictions are correct (True), compared to the total amount of predictions (Total). The next three metrics are label dependent. For each category k, I calculate the following scores:

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.6}$$

In which TP (True Positive), is the total of correctly classified items in category k, FP (False Positive) the total of items that were classified as category k, but in reality belong to a different category. As shown in Equation (4.4), Precision uses only these numbers. Recall, for which the equation is given in Equation (4.5), uses FN (False Negative), instead. for the total of items that should have been classified as category k, but was incorrectly classified otherwise. The F1-score combines the precision and recall, by taking the harmonic mean of the two metrics, as shown in Equation (4.6).

As shown in Table 3.6, the categories in the dataset are not evenly spread and the label 'NL' will be by far the most common. Though, I am most interested in the capabilities of the models for the classification of labels VAR, ENG, MOR and OTH. Instead of amplifying the Dutch label in the evaluation due to its high occurrence, I use the macro-average F1-score as a final evaluation. This is calculated as the average of the F1-scores of each label, weighing each label equally, despite the high number of NL words.

The human annotation agreement was 0,70. I expect the value for the evaluation metrics to not exceed this number by much.

I compare the macro-average F1-scores of RobBERT, M-BERT and the multiclass logistic regression model, to see which model performs the best, and by how much.

4.3.1 RobBERT: Hyperparameter tuning and Results

Training the best model requires hyperparameter tuning: the process of systematically adjusting the hyperparameters of a machine learning model to optimize its performance. Hyperparameters are configuration settings that influence the model's learning process. Tuning aims to enhance a model's accuracy and generalization on unseen data. For this process, I specifically focus on the hyperparameters *batch size* and *learning rate*.

With a small batch size, each iteration provides the model with a limited sample of the entire dataset. The model's parameter updates are based on this limited view, introducing variability. As the training dataset is not very large, a smaller batch size may result in better generalization. However, too small a batch size might make learning too erratic. Aguilar et al. [2020] use a batch size of 32. I also try this batch size, and three smaller batch sizes of 4, 8 and 16.

The learning rate might need adjustment based on the batch size. Smaller batches often require a smaller learning rate. Aguilar et al. [2020] uses a learning rate (η) of $5 \cdot 10^{-5}$. I try the same configuration, as well as one smaller learning rate of $2.5 \cdot 10^{-5}$ and two larger learning rates of $\eta = 7.5 \cdot 10^{-5}$ and $1.0 \cdot 10^{-4}$.

Like Aguilar et al. [2020], I use the AdamW optimizer [Loshchilov and Hutter 2019]. All of my experimental choices are tuned by observing the performance on the validation sets. The models with the highest perfoming configurations will be exposed to the test set. To ensure reproducibility, I run all my experiments with the same seed, set to 42.

Scores	Batch size = 4	Batch = 8	**Batch = 16**	Batch = 32	Average
$\eta = 2.5 \cdot 10^{-5}$	0,8017	0,8045	0,7927	0,7562	0,788
$\eta = 5.0 \cdot 10^{-5}$	0,8087	0,7899	0,7866	0,8049	0,797
$\eta = 7.5 \cdot 10^{-5}$	0,7934	0,7927	0,8092	0,8131	**0,8021**
$\eta = 1.0 \cdot 10^{-4}$	0,7846	0,7929	0,8207	0,8011	0,7998
Average	0,7971	0,795	**0,8023**	0,792	-

Table 4.5 Best Macro-average F1-scores on validation set for different configurations of RobBERT evaluating on subword level, best epoch out of 8.

I fine-tune 16 different RobBERT models. Each is trained for eight epochs for every combination of the hyperparameters batch size and learning rate. I report the macro-average F1-score for the best epoch of each combination in Table 4.5.

As shown in bold in Table 4.5, a learning rate of $7.5 \cdot 10^{-5}$ (0,000075) receives the highest F1-score on average. The models using a batch size of 16 are on average the highest scoring, so I continue using these hyperparameters.

I initially chose 8 as the number of epochs because there was no improvement in validation loss at all after eight epochs. However, I later realized that loss is not always the best metric for model performance, as it may not perfectly align with the classification metrics I am interested in optimizing, e.g. precision, recall, F1-score. This can be due to class imbalance, if the model focuses on the more prevalent classes to minimize the loss, while the minority classes are neglected. Another reason for the unusual relationship between the two metrics may be the large overlap between the training and validation set. The increasing validation loss indicates overfitting on the training set. However, as the training set and the validation set have a large word overlap, a slight increase in loss may not necessarily indicate a decrease in classification performance. The three classification metrics were often still improving on the last epoch of training, so I decide to train the on average best performing hyperparameter configurations again, for a higher amount of 14 epochs.

Instead of retrying all combinations, I pick the two on average best performing parameters and compare only those. First I set the batch size to 16, then try each of the four learning rates from the previous search. Then, I set the learning rate to 7,5e-5, and try each of the four batch sizes from the previous search. The macro-average F1-scores for these models are depicted in Table 4.6:

Ultimately, the two models with the configurations that perform best on the validation set both use learning rate 7.5e-5, one with batch size 4 and one with batch size 16. To determine which configuration is the best, I let them loose on the test set. The former performed worse on the test dataset, with a macro-average F1-score of 0,79, whereas the latter reached a score of 0,81, which is closer to that of the validation set. Also, based on my hyperparameter search

batch size: 16		learning rate: 7,5 e-5	
learning rate	F1-score	batch size	F1-score
2.5e-5	0,8194	**4**	**0,8369**
5e-5	0,8051	8	0,7956
7.5e-5	**0,8307**	16	0,8307
10e-5	0,8088	32	0,8232

Table 4.6 Best RobBERT F1-scores on the validation set, each with a maximum of 14 epochs. On the left for batch size 16, with different learning rates. On the right, different batch sizes for learning rate 7,5e-5. The highest scoring combination is highlighted in **bold**.

Hyperparameter	Value
batch size	16
learning-rate (η)	$7{,}5 \cdot 10^{-5}$
optimizer	Adam with betas=(0.9,0.999)
ε	$1 \cdot 10^{-8}$
lr_scheduler_type	linear
number of epochs	14

Table 4.7 Configuration of the fine-tuned RobBERT model with the highest scoring hyperparameters on the validation set.

from Table 4.5, batch size 16 seemed to be the better option on average. Therefore I decide to select the following configuration of the model as my representative for the RobBERT model:

I then initially tried four different seeds with these configurations. Based on the then five seeds, I selected the best model for error analysis in Section 5.1. For extra robustness, I subsequently trained five more, reaching a total of 10 seeds of the RobBERT model with the best configuration. The average scores for each label along with their standard deviation are shown in Table 4.8. Unsurprisingly, NL receives the highest score, with NON following closely. The least common label, OTH, is also the lowest scoring one, which is also in line with other work.

4.3.2 M-BERT: Hyperparameter tuning and Results

For the hyperparameter tuning of M-BERT, the exact same dataset and classification metrics are used. A logical starting point is the same configuration as the best one from RobBERT

Labels	Precision	Recall	F1-score	Support
NL	0,97 (σ=0,002)	0,99 (σ=0,001)	0,98 (σ=0,001)	3018
VAR	0,81 (σ=0,012)	0,69 (σ=0,016)	0,75 (σ=0,006)	306
NAME	0,88 (σ=0,017)	0,92 (σ=0,012)	0,90 (σ=0,008)	289
MOR	0,88 (σ= 0,014)	0,87 (σ= 0,014)	0,88 (σ= 0,006)	258
NON	0,91 (σ=0,029)	0,95 (σ= 0,009)	0,93 (σ=0,012)	145
ENG	0,87 (σ=0,028)	0,84 (σ=0,028)	0,85 (σ=0,023)	100
OTH	0,61 (σ=0,085)	0,47 (σ=0,043)	0,53 (σ=0,044)	33
micro avg (acc)	0,94 (σ= 0,001)	0,94	0,94	4149
macro avg	0,84 (σ=0,011)	0,82 (σ=0,007)	0,83 (σ= 0,008)	4149

Table 4.8 Results on the validation set per label of the RobBERT model with the selected configuration. Means and standard deviations based on 10 different seeds.

batch size: 16		learning rate: 7.5e-5	
learning rate	F1-score	batch-size	F1-score
2,5e-5	0,8365	4	0,8374
5e-5	0,8416	8	0,8228
7,5e-5	**0,8450**	**16**	**0,8450**
10e-5	0,8287	32	0,8283

Table 4.9 Best M-BERT F1-scores on the validation set after 14 epochs of training. On the left, different batch sizes on learning rate 7.5e-5. On the right, different learning rates for batch size 16. The highest scoring combination is highlighted in **bold**.

while tuning M-BERT. I again set a seed to 42, as I have done for RobBERT. Starting with the same configuration as (4.7):

Now I compare the same batch sizes as before on the macro-average F1-score, as shown in Table 4.9. This table also depicts the comparison on the learning rates as I have for RobBERT:

Just like the RobBERT model, the combination of batch-size = 16 and learning-rate = 7.5e-5 works best for M-BERT. Again, I run the model on four extra random seeds. Based on these five models, I choose the best model to analyse in Section 5.1. For robustness, I run another five seeds, creating a total of ten models of this configuration. The average evaluation per label on the validation set is shown in Table 4.10. All scores do seem to be a little higher than those of RobBERT. I compare them in section Section 4.5.

Labels	Precision	Recall	F1-score	Support
NL	0,96 (σ= 0,004)	0,98 (σ=0,003)	0,97 (σ=0,001)	3018
VAR	0,76 (σ= 0,024)	0,62 (σ= 0,037)	0,68 (σ= 0,017)	306
NAME	0,89 (σ= 0,017)	0,92 (σ= 0,012)	0,90 (σ= 0,009)	289
MOR	0,90 (σ= 0,025)	0,86 (σ= 0,026)	0,88 (σ= 0,011)	258
NON	0,94 (σ= 0,022)	0,93 (σ= 0,027)	0,93 (σ= 0,009)	145
ENG	0,92 (σ= 0,031)	0,82 (σ= 0,049)	0,87 (σ= 0,025)	100
OTH	0,74 (σ= 0,103)	0,53 (σ= 0,066)	0,62 (σ= 0,054)	33
accuracy	0,94	0,94	0,94 (σ= 0,002)	4149
macro-average	0,87 (σ= 0,017)	0,81 (σ= 0,012)	0,84 (σ= 0,009)	4149

Table 4.10 Validation-set results per label for the M-BERT model that performed the best. Means and standard deviations based on ten different seeds.

Parameter	Options
multiclass	[ovr, multinomial]
solver	[lbfgs, newton-cg, sag, saga]
penalty	[l1, l2]

Table 4.11 Possible parameters for logistic regression used during grid search

4.3.3 Baseline: Logistic Regression

I fit a multinomial logistic regression model on the same training set as I used for the transformer models.

Initially, the model would predict NL, the most common label, for each word. To improve this, I performed a grid search for the best hyperparameters on the training set, using 5 folds of cross-validation on the training set, and eventually got better results.

Table 4.11 shows all the parameters that were tested in the search. Multiclass determines how the model handles multiple classes, with the first strategy (‘ovr’) making the classification for each class a binary problem. Solver denotes the optimization algorithm used during training. ‘lbfgs’(Limited-memory Broyden–Fletcher–Goldfarb–Shanno) is memory-efficient, but less efficient on large datasets. ‘newton-cg’ (Newton Conjugate Gradient) requires more memory, but is suitable for larger datasets. ‘sag’ (Stochastic Average Gradient) is a stochastic optimization algorithm. It is particularly useful for larger datasets, as it is computationally efficient. Finally, saga (SAGA - Shuffling Approximate Gradient Descent), is an extension of sag, supporting l1 regularization. The penalty determines the type of regularization applied

Penalty	l1		l2	
	‘ovr’	‘multinomial’	‘ovr’	‘multinomial’
‘lbfgs’	-	-	0,45	0,51
‘newton-cg	-	-	0,45	0,51
‘sag’	-	-	0,45	0,51
‘saga’	0,48	0,54	0,45	0,51

Table 4.12 Macro-average F1-scores on validation set for combination of hyperparameters of the multinomial logistic regression model. The ‘l1’ penalty is only compatible with the ‘saga’ solver.

Labels	Precision	Recall	F1-score	Support
NL	0,81	1.00	0,89	3018
VAR	0,94	0,19	0,32	306
NAME	0,93	0,64	0,76	289
MOR	0,89	0,26	0,40	258
NON	0,97	0,42	0,59	145
ENG	0,95	0,36	0,52	100
OTH	1.00	0,18	0,31	33
accuracy	0,82	0,82	0,82	4149
macro-average	0,93	0,44	0,54	4149

Table 4.13 Validation-set results per label for the multinomial regression model

to prevent overfitting, for which ‘l1’ means the absolute values of coefficients and ‘l2’ the squared values of coefficients, putting more weight on mistakes.

As shown in Table 4.12, the type of solver did not alter the macro-average F1-score, while using l2 regularization. ‘Multinomial’ reached a higher score than ‘ovr’ in all cases. The best combination is: multiclass=‘multinomial’, solver=‘saga’, penalty=‘l1’, reaching a macro-average F1-score of 0,54.

The results of the logistic regression model with the best combination of parameters are shown per label in Table 4.13 for the validation set, and in Table 4.14 for the test set.

Interestingly, the model predicts NL disproportionately often, reaching an almost perfect Recall score of 0,997, rounded up to 1.00 in both result tables. However, this does result in a lower precision for NL. The model seems to only predict a label other than NL for words of which it is certain. This ensures a very high precision score for each label other than NL.

Labels	Precision	Recall	F1-score	Support
NL	0,81	1,00	0,89	2942
NAME	0,97	0,59	0,73	314
VAR	0,93	0,19	0,31	264
MOR	0,93	0,29	0,45	221
NON	0,96	0,47	0,63	176
ENG	0,93	0,21	0,35	122
OTH	0,83	0,21	0,33	24
accuracy	0,82	0,82	0,82	4063
macro-average	0,91	0,42	0,53	4063

Table 4.14 Test set results per label for the Multinomial logistic regression model

However, for less common words, the prediction often remains NL, resulting in very low Recall and F1 scores, compared to the results from the transformer models.

4.4 Data augmentation

After finding the best model hyperparameters, I still had promise of improvement via approaches that are typical for imbalanced data, like upsampling. This addresses class imbalance by increasing the frequency of tokens from the underrepresented classes in the training data. Typically, tokens from only these classes would be duplicated or synthetically generated. Duplicating only the words from the underrepresented classes was possible, but I opted to leave the words in their natural environment, so the models may use contextual clues for their decision making. I decided to deploy data augmentation by duplicating entire lines in my dataset that contained at least one of the three labels that did not appear as often: OTH (148 occurrences in the training set), ENG (510 occurrences in the training set) and MOR (1169 occurrences in test set).

I approximately doubled the occurrence of the three underrepresented labels by duplicating each line in the training set in which at least one of these three labels were present. As I duplicate entire lines, instead of only the individual words from underrrepresented labels, this intrinsically includes words from overrepresented labels.

The main goal of this procedure is to increase balance in the dataset by reducing the underrepresentation of OTH, ENG and MOR. While I considered the inclusion of the VAR label, I observe that VAR almost exclusively occurs together with the most overrepresented label NL, which may cancel out the effect of upsampling. Also, it is only the fourth least occurring label.

In total, 800 lines are added to the training set, each containing at least one word with either OTH, ENG or MOR for a label.

Labels	Precision		Recall		F1-score		Support
	M-BERT	RobBERT	M-BERT	RobBERT	M-BERT	RobBERT	
NL	0,97 (+0,01)	0,97 (=)	0,98 (=)	0,99 (=)	0,97 (=)	0,98 (=)	3018
VAR	0,74 (-0,02)	0,84 (+0,03)	0,65 (+0,03)	0,67 (-0,02)	0,69 (+0,01)	0,74 (-0,01)	306
NAME	0,88 (-0,01)	0,88 (=)	0,92 (=)	0,92 (=)	0,90 (=)	0,90 (=)	289
MOR	0,89 (-0,01)	0,87 (-0,01)	0,86 (=)	0,88 (+0,01)	0,88 (+0)	0,87 (-0,01)	258
NON	0,90 (-0,04)	0,91 (=)	0,94 (+0,01)	0,94 (-0,01)	0,92 (-0,01)	0,93 (=)	145
ENG	0,95 (+0,03)	(-0,03)	0,71 (-0,09)	0,85 (+0,01)	0,81 (-0,06)	0,85 (=)	100
OTH	0,83 (+0,09)	0,64 (+0,03)	0,58 (+0,05)	0,48 (+0,01)	0,68 (+0,06)	0,55 (+0,02)	33
accuracy	0,94 (=)	0,94 (=)	0,94 (=)	0,94 (=)	0,94 (=)	0,94 (=)	4149
macro avg	0,88 (+0,01)	0,85 (+0,01)	0,81 (=)	0,82 (=)	0,84 (=)	0,83 (=)	4149

Table 4.15 Results on augmented dataset, evaluating on the validation set. The numbers in brackets are the deviations from the average results on the non-augmented training set, from Table 4.8 and Table 4.10. Deviating results are colored if higher, lower or within the standard deviation of the average results on the non-augmented dataset.

This results in a slightly larger training dataset of 4566 lines. I do not augment the validation and test set, as this would increase the difficulty of the task and cancel out the augmented training set, nor would the results be comparable to the previous scores.

I train a RobBERT and a M-BERT model on this augmented dataset, using the same configurations as the best models on the non-augmented dataset, as depicted in (4.7). The results of both M-BERT and RobBERT are shown in Table 4.15. As the results of training a transformer model are sensitive to random initialization, I compare the results to the averages and standard deviations of the ten seeds I previously ran in Table 4.8 and Table 4.10. Results in orange indicate a small change falling within the standard deviations, compared to the scores from Table 4.8 and Table 4.10. In magenta I highlight results that are higher and go beyond the standard deviations. In blue, I highlight the results that are lower and fall out of the standard deviation.

For the upsampled labels MOR, ENG and OTH, the only label with exclusively positive changes compared to the models trained on the non-augmented dataset, is OTH. M-BERT even reached an F1-score that falls out of the standard deviation from the results on the non-augmented dataset. For MOR, the results barely made a difference, and for ENG the models trained on the augmented dataset reached a much lower score, suggesting a negative impact of the augmentation. Generally, the results in Table 4.15 are the same or very similar to the results without data augmentation. For M-BERT there are more negative than positive changes in the augmented results compared to the results before augmentation. For RobBERT, there are no negative changes that exceed the standard deviation from the previous model, but only one positive change that does this.

Labels	Precision		Recall		Support
	M-BERT	RobBERT	M-BERT	RobBERT	
NL	0,96 (σ= 0,004)	0,97 (σ= 0,002)	0,98 (σ= 0,003)	0,98 (σ= 0,002)	2942
NAME	0,91 (σ= 0,017)	0,89 (σ= 0,020)	0,91 (σ= 0,019)	0,92 (σ= 0,012)	314
VAR	0,74 (σ= 0,036)	0,81 (σ=0,012)	0,59 (σ= 0,030)	0,67 (σ=0,023)	264
MOR	0,82 (σ= 0,027)	0,85 (σ=0,016)	0,90 (σ= 0,025)	0,88 (σ=0,018)	221
NON	0,98 (σ= 0,006)	0,93 (σ=0,026)	0,88 (σ= 0,027)	0,91 (σ=0,009)	176
ENG	0,92 (σ= 0,023)	0,89 σ= 0,030)	0,85 (σ= 0,034)	0,87 (σ= 0,015)	122
OTH	0,54 (σ= 0,085)	0,40 (σ= 0,064)	0,44 (σ= 0,095)	0,50 (σ= 0,059)	24
macro avg	0,84 (σ= 0,018)	0,82 (σ= 0,010)	0,79 (σ= 0,015)	0,82 (σ= 0,011)	4063

Table 4.16 Comparison of means of Precision and Recall between the two transformer models, evaluated on the test set. The means and standard deviations are taken from ten seeds of each model. The higher score between the two models is highlighted in magenta for scores significantly higher (p-value < 0,05) or in orange for scores that are not significantly higher. Recall for NL is the same for both, so is not highlighted.

Because I found no indication that using the augmented dataset improves results, I choose to keep working with the models that are trained on the non-augmented data.

4.5 Comparison

As seen in Section 4.3, I trained five versions of the models with the best hyperparameter settings, each with a different seed. To test robustness, I try five extra seeds, making the total ten seeds and report the average scores for both RobBERT and M-BERT.

In Table 4.16, I show the precision and recall scores for each model next to each other. The F1-scores I show separately, in Table 4.17. These are the average scores of ten seeds of the models with the best configuration, performed on the test set. For each label and metric, I highlight which of the two models reaches a higher score on average. The highlight is given in magenta if the score is significantly higher than the other model's average (p <0,05). I tested for significance using a two-sample t-test. Interestingly, M-BERT is better at precision, whereas RobBERT reaches the higher score on Recall. A higher recall indicates that the model is effective at identifying instances of the relevant classes but may be more prone to false positives. A possible explanation for the higher recall for RobBERT is the more extensive vocabulary and pre-training of the Dutch language, compared to M-BERT. This would explain a higher macro-average recall score, as the model is better at recognizing a category as VAR or NL. When recall is higher, precision is often lower and vice versa, due to the inherent trade-off between the two metrics.

VAR receives the greatest difference between the two models on these metrics. RobBERT's Recall score for VAR is 0,08 higher than M-BERT's Recall for VAR. It is also the only

Labels	F1-score (M-BERT)	F1-score (RobBERT)	Support
NL	0,97 (σ=0,002)	0,98 (σ=0,001)	2942
NAME	0,91 (σ=0,010)	0,90 (σ=0,010)	314
VAR	0,65 (σ=0,016)	0,73 (σ= 0,015)	264
MOR	*0,86 (σ=0,015)*	*0,86 (σ= 0,012)*	221
NON	0,93 (σ=0,016)	0,92 (σ= 0,013)	176
ENG	*0,88 (σ=0,023)*	*0,88 (σ= 0,019)*	122
OTH	0,48 (σ=0,053)	0,44 (σ= 0,049)	24
accuracy	0,93 (σ= 0,004)	0,94 (σ=0,003)	4063
macro avg F1-score	0,81 (σ=0,013)	0,82 (σ= 0,010)	4063

Table 4.17 Comparison of Test-set results per label for BERT models that performed the best. The higher score between the two models is highlighted in magenta for scores significantly higher (p-value $< 0{,}05$) or in orange for scores that are not significantly higher. Scores that are the same for both are in *italics*.

label for which the same model has a higher score on both precision and recall. This may be explained, as RobBERT's vocabulary has more Dutch words than M-BERT, making it easier to recognize a spelling variation for RobBERT than for M-BERT. The differences between scores for OTH are substantial, and so are their standard deviations, making this the least stable scoring label. This may be due to the small number of instances in both the training and test data. A small number of examples makes a model susceptible to the effects of randomness. Additionally, the correct classification of a single word in this category carries more weight for the final scores compared to a word in a larger category.

In Table 4.17, I compare the F1-scores of the two models for each label, as well as the accuracy and the macro-average F1-score. For this comparison I highlighted the higher score between the two models. If the difference was significant according to a two-sample t-test, the highlight is in magenta, otherwise in orange. If both models reached the same score, I highlighted this label in italics. Most labels receive a very similar score for either of the two models. Only NL and VAR receive an F1-score that is significantly higher for RobBERT than for M-BERT. This can be attributed to the fact that RobBERT was only trained on the Dutch language, resulting in a larger Dutch vocabulary. Although M-BERT has slightly higher scores for three labels, and RobBERT only for two, it is the large difference in score for the VAR label that pulls up the macro-average F1-score for RobBERT. The significantly higher accuracy for RobBERT can be explained by the fact that RobBERT's scores are not significantly lower for the categories that score less than M-BERT, but are significantly higher for those that score higher. Notably, the two categories where RobBERT outperforms M-BERT have a substantial

	M-BERT	RobBERT	Logistic Regression
Precision	0,842 (σ=0,018)	0,820 (σ=0,010)	**0,9082 (σ=0,00)**
Recall	0,793 (σ=0,015)	**0,819 (σ=0,011)**	0,4224 (σ=0,00)
F1-score	0,812 (σ=0,013)	**0,817 (σ=0,010)**	0,5278 (σ=0,00)
Accuracy	0,933 (σ=0,004)	**0,940 (σ=0,002)**	0,82 (σ=0,00)

Table 4.18 Comparison of Average scores based on ten different random seeds. The highest score is denoted in **bold**.

number of instances, contributing significantly to the overall accuracy, particularly in the case of NL. The difference in macro-average F1-score is not significant (p=0,07).

Ultimately, both models perform this task adequately, given that Cohen's kappa for the annotation of this dataset was only 0,70, the scores the two transformer models reached are high. The models are especially good at (F1-score > 0,90) recognizing Dutch (NL), Named entities (NAME) and Language independent utterances (NON). Both English and Moroccan languages also receive high scores (F1-score > 0,85). The labels I personally found the most challenging during annotation, manifest the lowest scores for both models: (spelling or grammatical) Variations of Dutch words (VAR) and words from other languages (OTH).

Table 4.18 showcases the final results of the three different models. The results are based on ten different random seeds of initializing the training process with the previously discussed hyperparameters. For the logistic regression model, different seeds did not yield different results. Logistic regression is a deterministic algorithm, meaning that given the same input and parameters, it will produce the same output every time. Unlike in the transformer models, the random seed in logistic regression typically influences the initialization of the optimization process rather than the randomness in the model itself.

Interestingly, the precision was highest for the logistic regression model. This is explainable, as this baseline classifies most words as NL and seems to only classify a word as another label when it is certain it belongs to that category, because the model has seen it before, in the training set. This results in a precision score of higher than 0,90 for almost all categories. NL is the exception category, which is often attributed to words that are not Dutch, resulting in many false positives. This also results in an almost perfect Recall score for NL (0,997), but a low score for each other label, making the average Recall and F1-score rather low.

The other three metrics: Recall, F1-score and Accuracy are all much higher for the transformer models, and each is the highest for RobBERT. Both the average scores Recall and Accuracy are significantly higher than those for M-BERT (p <0,05). I tested for significance using a two-sample t-test. The difference in F1-score between the two transformer models was not significant.

5 Discussion and Conclusion

In this chapter I discuss the results of the models, as well as limitations of my research and ideas for future work. In Section 5.1, I show what the models can and cannot do, by analyzing the models' predictions. Then, I elaborate on ideas for Future Research in Section 5.2. Finally, I provide a short summary of my findings in Section 5.3.

In short, the models are not good at recognizing spelling variations of recognizable Dutch words (VAR). The inclusion of such a label in combination with research on code-switching may be too confusing for a model to learn next to labels for other languages. However, for recognizing the Moroccan ethnolect or studying other language varieties, like Straattaal, it is essential. The characteristics of the label may be split into various parts, like spelling errors and grammatical errors being separate labels. The combination of the two phenomena may have been too difficult. The models are relatively good at recognizing Dutch (NL), Named entities (NAME), language independent utterances (NON) and English words (ENG). If there had been more instances of words from other languages (OTH), it may have performed better, but for now its results are the lowest. For a full explanation of the labels, see Section 3.2.1.

The recognition of Moroccan languages (MOR) sees quite impressive scores, if I compare the scores to the Cohen's kappa from the annotation evaluation. This can be explained by the fact that the annotations were all done by me, making the models emulate my own decisions. If multiple annotators had been involved, the score may not have exceeded the calculated agreement. I would say that the models can be used to recognize and extract Moroccan languages from an informal Dutch text.

5.1 Error Analysis

After running the first five different seeds for both M-BERT and RobBERT models, the models run with seed 42 came out as the highest performing model. Therefore, these are the models I use for this error analysis[1,2].

Later, I ran five extra models with different seeds for robustness of the results in Chapter 4, of which two reach a higher macro-average F1-score than the model mentioned above, but I choose to remain working with this model, as I had already started on this analysis and the higher scoring models fall within the standard deviations and don't differ much.

[1] https://huggingface.co/Tommert25/RobBERTBestModelOct13

[2] https://huggingface.co/Tommert25/MultiBERTBestModelOct13

	Total errors per label			Percentage of errors per label		
Label	M-BERT	RobBERT	Baseline	M-BERT	RobBERT	Baseline
NL	51	54	9	1,7%	1,8%	0,3%
VAR	110	94	214	41,7%	34,9%	81,1%
NAME	29	23	129	9,2%	7,3%	41,1%
MOR	18	31	156	8,1%	14,0%	70,6%
NON	21	16	94	12,0%	9,1%	53,4%
ENG	19	14	96	15,4%	11,5%	78,7%
OTH	12	11	19	50,0%	45,8%	79,2%
Total	260	238	717	6,4%	5,9%	17,6%

Table 5.1 Absolute and relative amounts of errors per category on the test set of the annotated dataset, for all three trained models.

The finished models reach different scores for each label, as seen in Section 4.5. Generally, the models tend to have problems with the same labels that I had during annotation: VAR, MOR and OTH, see Section 3.2.3. Additionally, several mistakes tend to be model-specific. For instance, RobBERT is slightly better at recognizing VAR, whereas M-BERT makes fewer mistakes at recognizing MOR.

I analyze the diversity of predictions for each category of words, as well as the frequency of the different types of errors. For this analysis I only examine the predictions for the test set. First, I present a quantitative analysis on the errors of the models in Section 5.1.1, followed by a short qualitative analysis in Section 5.1.2 for the types of errors of each label.

5.1.1 Quantitative analysis

In Table 5.1, I made an overview of the amount of errors per category, both absolute and relative. Out of the 4063 words in the test set that were classified, M-BERT misclassified a total of 260 words, while for RobBERT that number is 238 words. Relatively, OTH is misclassified the most by the transformer models, but in absolute, VAR has the most errors, for both transformer models. For the baseline, VAR has the most errors both in relative and absolute.

The heatmaps in Figure 5.1 and Figure 5.2, show for each pair of prediction and true label how many instances were predicted for M-BERT and RobBERT, respectively. VAR and NL are most often mistaken for each other, as shown in both heatmaps, and predicted often for other words as well. OTH, ENG and NON are not often wrongly predicted by M-BERT, but a little more by RobBERT. NON is often mistaken for MOR by M-BERT. Generally, RobBERT is a little more diverse in its predictions, as shown by the slightly more spread out prevalence of green tiles, than M-BERT, whose colored tiles are quite centralized in the top left corner.

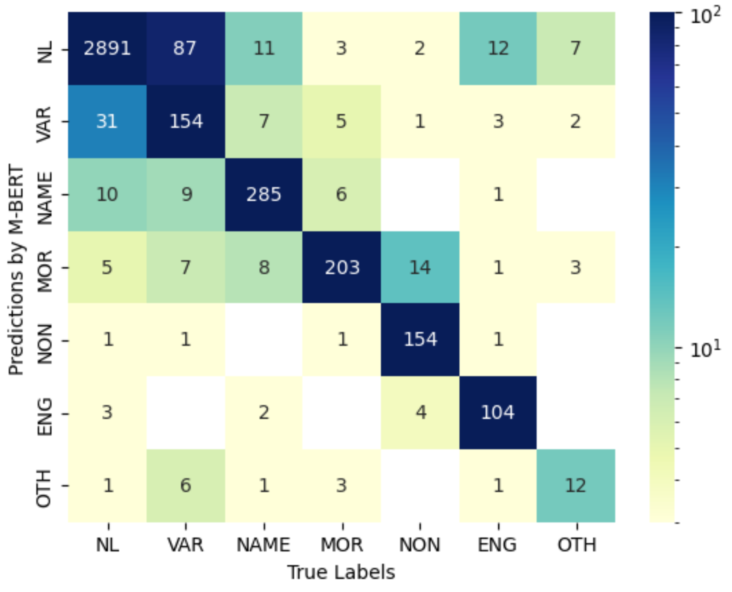


Figure 5.1 Predictions made by M-BERT on each of the 4063 words of the test set. On the x-axis the true label of a word, and on the y-axis the model prediction. The darker blue, the higher the number of instances of a pair of prediction and label. A white block without a number means that there were no instances of this prediction and label pair. On the diagonal, the predictions and the true labels match. These are the correctly classified words. All other squares reflect instances of incorrect predictions.

In Table 5.2, I made an overview of a comparison between predictions of M-BERT and RobBERT per label. M-BERT and RobBERT agreed 3825 times. That is 94.15% of all tokens. They disagreed 238 times (5.85%). In Figure 5.3 I show each prediction for which both RobBERT and M-BERT agree. These are mistakes that both models made in the exact same way (so M-BERT assigned the same wrong label as RobBERT). As visible in Figure 5.3, there are fewer green tiles and therefore less diversity in the mistakes in this figure than in the figures for the individual models.

Words in the VAR category have the highest number of incorrect predictions for both models. Within these errors, NL was predicted the most for the words. Notably, RobBERT makes fewer mistakes (94) than M-BERT (110) on this category. Perhaps the Dutch model is better at distinguishing right from wrong due to its extensive Dutch vocabulary, that M-BERT may not possess. There are 108 words for which both transformer models agree on the predicted label, whereas the true label is different. This leaves 150 errors unique to M-BERT and 128 unique to RobBERT. and a total of 388 unique wrong predictions. This is

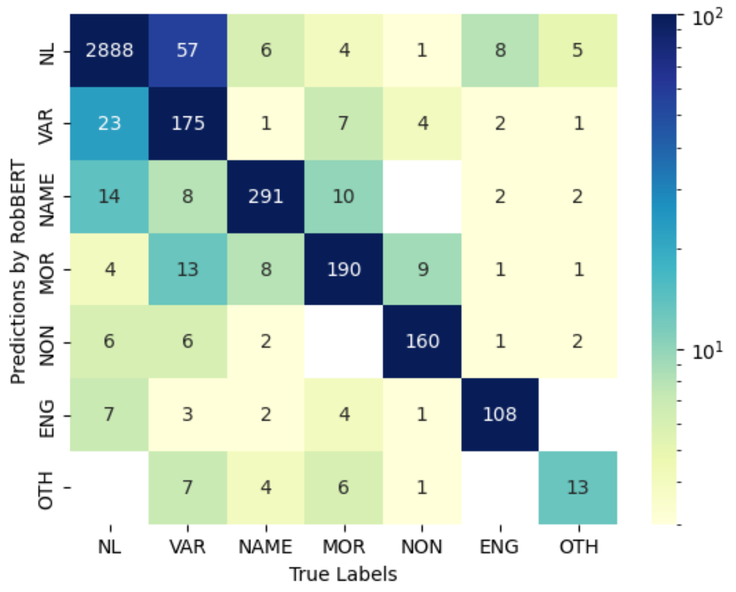


Figure 5.2 Predictions made by RobBERT on each of the 4063 words of the test set. The same color scheme was used as above in Figure 5.1.

the sum of the mistakes from both models, minus 110 mistakes the models have in common. Additionally, various words were misclassified by both models, but M-BERT and RobBERT disagree on a prediction. For example, ‘homoo’ is predicted to be MOR by M-BERT but ENG by RobBERT, when factually it belongs to VAR, as it is a spelling variation of the Dutch word ‘homo’, which means ‘gay’.

So, what *can* the transformer models do that the baseline cannot? The predictions made by the logistic regression model are shown in Figure 5.4. The model predicts NL 3629 times, out of 4063 words to be classified, or 89.3% of the time. This strategy results in very few mistakes for the NL category, but all labels, except for NAME, are more often classified as NL than they are correctly identified. The model rarely makes other mistakes: the word is either correctly predicted or predicted NL. Only a handful of different predictions are made.

As the transformer models make fewer mistakes in each category, except for NL, due to the apparent strategy of the baseline model, it is safe to say that the transformers are a lot better at this task.

5.1.2 Qualitative analysis

Now follows qualitative analysis regarding the words the models are unable to classify correctly. Generally, the mistakes the models agree on are for words I also found difficult or

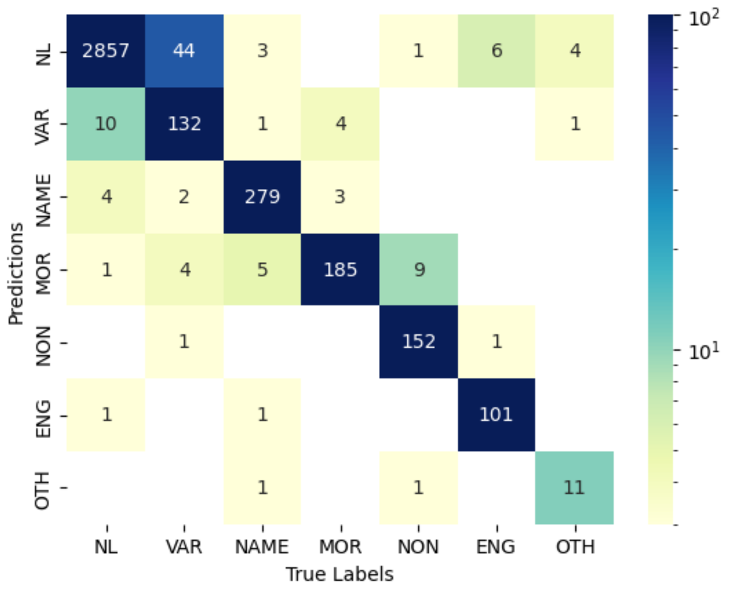


Figure 5.3 Predictions that were the same for both RobBERT and M-BERT. The same color scheme was used as for Figure 5.1.

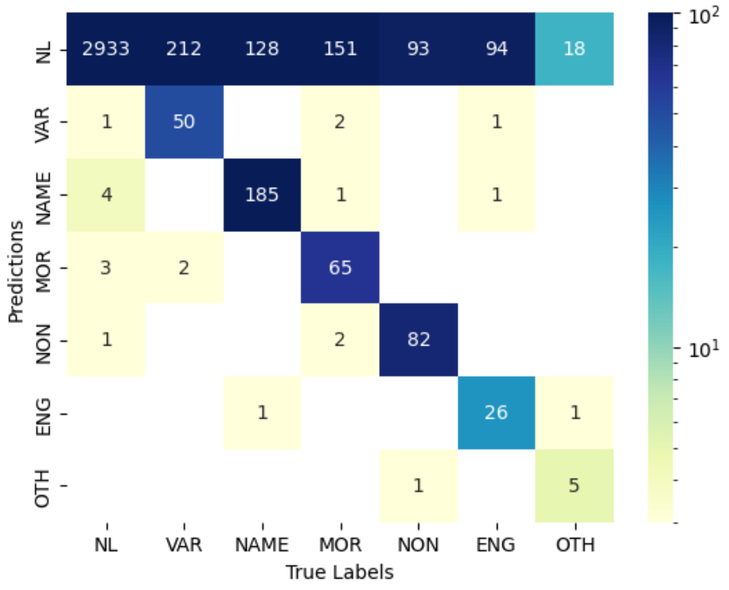


Figure 5.4 Predictions made by the logistic regression baseline model. The same color scheme was used as for Figure 5.1

	The models agree		The models disagree		
Label	Correct	Incorrect	M-BERT	RobBERT	Both wrong
NL	2857	16	34	31	4
VAR	132	51	22	43	16
NAME	279	11	6	12	6
MOR	185	7	18	5	6
NON	152	11	2	8	3
ENG	101	7	3	7	4
OTH	11	5	1	2	5
Total	3717	108	86	108	44

Table 5.2 Amount of predictions the transformer models agreed upon and disagreed upon, listed per true label. If the models disagreed, I also specified which model predicted the label correctly, or if they were both wrong.

ambiguous, as discussed in Section 3.2.3. The words the models disagree on, can sometimes be explained based on the model's properties.

I discuss the errors both models made, as well as which errors are more common in one model or the other. First, I discuss predictions for VAR.

5.1.2.1 VAR

The VAR category is most often misclassified as NL, by M-BERT (87 words), as well as RobBERT (57 words). 43 VAR words in the test set were predicted to be NL by both models. I try to find a theme in difficulty for the models. Examples are shown in the following

- Words that are technically correct Dutch, but were the wrong word to use, like *'schattige'*(wrong inflection), *'het'* (wrong article), *'me'* (wrongly used as a possessive pronoun).
- But also words with a typo, like *'filmpoje'* (from: 'filmpje', video) *'hulen'* (from: 'huilen', cry), or '*belefheid*'(from: 'beleefdheid', politeness).
- Words that are split due to their offensive nature into two words so that a chat moderator wouldn't flag them and ban them: 'hoe', 'rige'. (from: 'hoerige', slutty)

M-BERT misclassifies more VAR words than RobBERT. Especially simple spelling or typing mistakes are more often misclassified by M-BERT.

Such as, *'verkerd'* (from: 'verkeerd', wrong), *'scheinheilig'* (from: 'schijnheilig', hypocrite).*'geneusde'* (from: 'gekneusde', bruised)

Description	Example	Corrected version	Translation
Words lacking a final ‘n’	‘wille’	willen	(to) want
Letter repetitions	‘straksssssss’	straks	soon/later
Common unofficial Dutch acronyms	‘gwn’	gewoon	simply/just/usual
Words with strange encoding error	‘3d2it 3i2s 3e2cht’	dit is echt	this is real

Table 5.3 Types of VAR words the transformer models were typically able to recognize as VAR.

Several VAR words are misclassified as MOR, such as *‘samballl’* (sambal) and *‘wrm’* (acronym for ‘waarom’, why). This can be explained, as words in the MOR category can also be acronyms or may use letter repetition.

In Table 5.3, there is an overview of types of VAR words that both transformer models consistently predicted correctly. The logistic regression model simply predicted correctly any words that appeared in the training set, without a visible theme.

Most correctly classified words were either words that appeared many times, like ‘gwn’, a common way of writing ‘gewoon’ (simply, just). ‘ff’ (dutch acronym for ‘even’, which has many translations, it’s like ‘just’ or sth), ‘k’, which is short for ‘ik’ (I).

Words with letter repetitions were usually classified correctly as VAR, such as ‘babyyyy’ (baby), ‘gaaan’ (gaan, go) or ‘straksssssssss’ (straks, soon). These are typically not words that appeared more than once, as the amount of letter repetitions is rarely consistent

Words that lack a final ‘n’, were also classified correctly. ‘wille’, ‘make’, ‘mense’, ‘manne’.

Words that start with a 3 and a 2 around the first letter are also categorized correctly. ‘3d2it’, 3i2s’, ‘3e2cht’.

Overall, this category is an interesting one! However, I believe now that it is a too confusing category to train on in combination with classification of code-switching, as the label is too volatile and inconsistent, and is too much like NL.

5.1.2.2 NL

Relatively, the NL category has the least mistakes, but as it is such a frequent label, it is the second most misclassified label in absolute numbers.

M-BERT categorizes 31 words in the test set as VAR that I assigned NL, and 20 words as another category, mostly NAME. RobBERT’s most commonly assigned label for NL words is also VAR (23), although the predictions are more diverse than in M-BERT. VAR is followed closely by NAME (14), as well as 17 words something else (either ENG, NON or MOR).

First, I discuss words that the models thought to be VAR. 10 words are deemed VAR by both models, but NL by me. I found that in some cases the model was right where I was not. For instance, ‘dankjee’ (thankss) was classified as VAR, whereas I labeled it NL. The word has a letter repetition, and is normally spelled as ‘dankje’ so the models are correct that it

Category	Example
Laughter	‘hhhh’, ‘hahahahhah’
Emojis	‘:D’, ‘:P’, ‘:-)’
Human sounds	‘hmmm’, ‘pff’

Table 5.4 Types of words easily categorized as NON by transformer models

should be VAR. Another word that I labeled NL: ‘das’ (that’s), should be written as ‘da’s’ and may then still qualify for VAR, as it is an informal short version of ‘dat is’ (that is).

M-BERT does not recognize certain words that are clearly Dutch, but probably a little infrequent, so not part of M-BERT’s vocabulary, like ‘Afghaan’ (Afghan), ‘waterijsje’ (popsicle), ‘hoogmoed’ (pride).

In contrast, RobBERT does recognize these words as correct Dutch, but instead appears to be stricter than my annotations in two instances. Like for ‘naief’ (naive), which misses a diaeresis and thus should be ‘naïef’. RobBERT classified this word as VAR, whereas I assigned NL. The same goes for ‘beinvloedt’(influences), which should be ‘beïnvloedt’. As RobBERT is trained on informal data from the internet, this surprised me. During annotation, I chose to ignore punctuation-like mistakes, as this is very common in the dataset.

5.1.2.3 MOR

MOR generally does quite well, between 8 and 14 percent wrongly classified. Unlike VAR, the MOR words don’t really have one label they are confused with. NAME is most often predicted, followed closely by VAR.

RobBERT makes more mistakes than M-BERT in this category, especially on words that occur more often, despite being spelled differently, like ‘Wallah’, ‘imam’ and ‘hamdoelilah’, all being mistaken for NAME.

Some words are classified as OTH, by both models, such as ‘denia’ (Arabic for: world), ‘ilaha’ (Arabic for: worshipped) and ‘mlieh’ (Berber for: good). Although I wasn’t certain during annotation, they turned out to be correctly annotated by me.

5.1.2.4 NON

The NON category is generally easy for the transformer models. Laughter, emojis, website links, separate punctuation characters and human sounds such as ‘hmm’ and ‘pff’ were consistently categorized correctly as NON by both transformer models. See Table 5.4 for an overview. The logistic regression model categorizes only laughter, website links and human sounds consistently correct, omitting emojis and sole punctuation marks entirely from the category. Interestingly, the logistic regression model did recognize ‘bla’ as NON, whereas M-BERT did not. RobBERT did catch this word.

Most mistakes the models agreed on was one user who spammed ‘lalala’ various times, which both models judged to be MOR, for reasons unknown to me. Also, some human sounds were predicted to be VAR, like ‘ieeeeee’ and ‘aaai’.

Again, I found words that were wrongly labeled by me, such as ‘lol’, which I consistently label OTH, but once by accident NON, so the model remains consistent and labeled it as OTH. Nonetheless, it counts as a mistake.

I also counted links to websites as NON and although the models generally understand this, one link was labeled wrong by both M-BERT and RobBERT.

5.1.2.5 OTH

Despite the lack of instances in the OTH category in the test set, many were mistakes. Most words were labeled NL. Interestingly, many words could have been NL, like ‘pas’, which can mean ‘step’ in Dutch, but was in this context used as the French word for ‘no’, as in ‘pas de problème’ (no problem). Another example could be ‘heil’, which was used in the context of the nazi greeting ‘heil Hitler’, and therefore judged by me to be German. However, ‘heil’ also can mean ‘welfare’ in Dutch. The Dutch abbreviation ‘btw’ can mean either ‘belasting toegevoegde waarde’ (taxes added value), or it’s used as an internet acronym for ‘by the way’ in this context, counting as OTH. However, both models label this word as NL.

Again, some mistakes were made by me, setting up the model for failure. ‘woehoeeeeeeee’ should not have been labeled as OTH, but could either be NON or VAR, depending on if ‘woehoe’ counts as a word in Dutch. Van dale dictionary does not include it, and it would therefore be NON. The models both classified it as VAR, making it wrong anyway, but for a different category. Also, ‘Hitler’ should have been NAME, although both models predicted NL.

Given the multilingual premise of M-BERT, I presumed that M-BERT would perform better on this category than RobBERT would. This is barely the case, as they perform quite similarly. Although they only agree five times, most words in this category were misclassified. An example of M-BERT performing better is that ‘ciao’ was not recognized by RobBERT, but it was by M-BERT.

5.1.2.6 NAME

The NAME category goes relatively well. Most misclassified names on which the models agree are mistaken for MOR, which is understandable, as many names refer to Moroccan people with Moroccan names, like ‘zohra’, ‘A9al’, and ‘shai’. Mistakes in M-BERT are mostly due to wrongly predicting NL, followed by VAR. Sometimes this is understandable. For example, ‘Keukenprinses’ (kitchen princess), ‘ben’ ((I) am) and ‘ajax’ (Dutch soccer club) all appear in the Dutch language.

VAR is often assigned for names that contain letter repetition, like ‘iliass’, ‘balkannn’ and ‘marouannnn’, whereas English usernames can be mistaken for English, such as ‘SWeetheart’ and ‘ice-cream-man’.

RobBERT fails to recognize brand names like ‘sevenup’, ‘bacardi’ or ‘rtl8’, whereas M-BERT only misclassifies ‘bacardi’.

5.1.2.7 ENG

Again, in this category we see a lot words that could have been Dutch, but are used in an English context here, like ‘help’, ‘van’ or ‘room’ (which means cream in Dutch).

M-BERT misclassifies various to a human obviously non-Dutch words as NL, like ‘destruction’, ‘voice’ and ‘weed’. RobBERT does not do this.

5.2 Limitations and Future Work

In this section, I discuss the limitations of my two deliverables, and propose ideas for future work, approaching each component individually. First I elaborate on the annotated dataset, followed by a review of the fine-tuned classification models.

5.2.1 Dataset and Annotations

As discussed in Section 3.2.1, I annotated a subset of the Moroccorp on word-level, using a set of seven labels suitable for language identification. This set of labels includes language labels for Dutch, English, Moroccan languages and Other languages (NL, ENG, MOR, OTH) as well as labels for language independent utterances (NAME, NON), while labeling incorrect and non-standard instances of Dutch words separately (VAR).

The full, unlabeled corpus was scraped from a chat forum by Ruette and Van de Velde [2013], to collect data on informal language production. The authors stress that a corpus of chat language is only partially relevant for linguistic research, as only a small group of people is documented in a highly specific setting. This makes it challenging to generalize any findings. Contribution per individual user differed greatly. The twenty most active users contributed about 10% to the corpus [Ruette and Van de Velde 2013]. This limitation is continued in my subset of the Moroccorp, as the sample is skewed and not representing all possible variations, causing some features to get too much weight. The sampling of the first batch of the dataset was done with longer parts of text (200 lines per batch) in which some users were prominently featured, although they could be huge outliers. A clear example of an outlier is the user that writes each word like ‘3t2his’, encompassing the first letter of each word with a 3 and a 2, which I discussed in Section 3.2.3. Words written by this user are very dominant in the recognition of the VAR label. One in six words of the words that receive the VAR label are written by this user, whereas this user was the only user in the annotated dataset (as well as in the full Moroccorp) that typed like this. Despite this, many words that are correctly labeled VAR are of this type, impacting the scores of VAR greatly. In the future, such outliers could be excluded.

During the error analysis from Section 5.1, I discovered that my own annotations can be low in reliability. While I had ensured that words with at least two occurrences in the dataset were either labeled consistently or are exceptions, other words remained without inspection. While analyzing the models' mistakes in Section 5.1.2, I discovered that various words that had a sole occurrence in the dataset, were mislabeled by me. These are either words with spelling errors that I missed, or words that I labeled as VAR, but were actually used correctly so should have been NL. Other mislabeling was done due to me changing the definitions of a label during annotation. For example, I initially included language independent utterances, like ':D', in the label OTH. When I created a separate label NON for these instances, I relabeled them, but missed several. Another aspect that may be interesting to change is the splitting of words on spaces. In the dataset, various words are concatenated with a comma or a period instead of a whitespace, such as 'Zorro..jij' resulting in several words being treated as a single word by me during annotation, despite consisting of two words that each fall into a different category. In the future, the words could also be split on punctuation marks to prevent this.

Regarding the validity of the annotations, my limited knowledge on the languages present in the dataset made the annotation process more difficult. For example, I treated the wildly different languages of Arabic and Berber as one single language, due to my limitation of knowledge. They are from different language families and therefore quite different. In the future, a label for each Moroccan language could be used. Also, a label like 'unknown' may be introduced, to prevent mislabeling words that are too ambiguous to classify. As stated in Section 3.2.3, a fellow student, who is fluent in Berber and has knowledge of Arabic, helped me verify the Moroccan nature of the words I annotated as such. Their check indicated that the majority of my annotations are valid, but they only reviewed the 200 most frequently occurring unique instances I had annotated as Moroccan words. Moreover, words that I may not have recognized as Moroccan in the first place were not reviewed. In the future, the annotation process could be done by multiple annotators, who can review each other's work. This would limit inconsistencies to some extent, either due to inattentivity or a disagreement on the definitions of the labels.

As became clear from my analysis in Section 3.3, the annotated dataset does not feature each category of words in equal amounts. Especially OTH and ENG appear infrequently, as together they make up only 3,5% of the dataset. Perhaps, future research could combine these two labels, as the main focus of the dataset is code-switching between Dutch and the Moroccan languages as well as the use of the Moroccan-Dutch ethnolect. However, combining labels that are not closely related may also increase the difficulty of the task.

I also think that the use of a VAR label is too broad to capture in this classification task. The range of simple mistakes, like letter repetition to a more advanced mistake of using an existing word in the wrong place, in combination with code-switching seems too much to handle all at once. The label could be split into different types of spelling variations. For instance, spelling errors and grammatical errors separately. This may make it easier to classify.

For future work, intersentential code-switching in the Moroccorp can be investigated as well. Although the dataset does not provide timestamps on each message, making it unknown whether the text is of a conversation-like nature and the lines that follow each other are reactions to each other, it can still be investigated how often a switch happens between lines in the dataset. Additionally, the messages that were uttered by the same user can be grouped as one utterance, making the lines longer. It would be interesting to see if the amount of intersentential code-switching is (dis)similar to the amount of intra-sentential code-switching in the Moroccorp, as each type of code-switching uses different linguistic patterns and serve different communicative functions.

Like I discussed in Section 3.3.3, the Code-Mixing index on document-level measures how many words in the dataset does not belong to the matrix language. Intrinsically, this does not reflect the amount of intra-sentential code-switching, but how multilingual a document is. The Code-Mixing index on utterance-level does reflect the amount of intra-sentential code-switching. Other than the paper that initially proposed this metric, I found no other users of this metric. As the goal of this index is to make the amount of code-switching comparable among datasets, this aspect could be investigated further in the future.

5.2.2 Models and training

As discussed in Chapter 4, I fine-tuned two pre-trained deep learning transformer-based language models, named Multilingual BERT (M-BERT) and RobBERT, on the task of token classification for the annotated dataset. Additionally, I trained a multinomial logistic regression model on the same task.

I performed hyperparameter tuning, to optimize the results of the models. The current hyperparameter tuning could have been more thorough as I started out with a maximum of 8 epochs, whereas later it became clear that 14 epochs generally results in a higher score. Also, I performed the hyperparameter tuning all on the same seed. However, by not trying multiple seeds, some performance differences could simply be random. In the future, more models could be run to ensure that the best hyperparameters are found. Future work could also explore the tuning of different hyperparameters, like weight decay, the type of optimizer algorithm or a dropout rate, as well as their combination.

The amount of word overlap between the training, validation and test set is relatively high (75% between validation and training set). This could be an explanation for the increasing classification scores in combination with increasing validation loss, as the models are actually overfitted on the training set, becoming worse at generalization. However, because the validation and test sets are both highly similar to the training set, the overfitting is not reflected in the evaluation scores but only in the validation loss. In the future, word overlap could be limited as the dataset splits become larger.

Because of the unequal amount of instances for each label in the dataset, I augmented the data, by copying each line in the training set in which one of the three least common labels (MOR, ENG, OTH) were present. As exemplified by the low number of 24 instances of the OTH label in the test set, compared to the 2942 NL words, more data on the less frequent labels is needed. The data augmentation I performed in Section 4.4 did not show improvement, as shown in Table 4.15. Future work could use a larger dataset by annotating more text, until I reach a minimum amount of words that falls in a category. Additionally, I could employ other forms of data augmentation, like the exclusion of monolingual lines or the downsampling of the most common labels. This would amplify the amount of examples in which there is code-switching present, making the models more familiar with it.

In the future, the models I fine-tuned may be used for other language identification tasks, using other datasets that are suitable for it, like the datasets used in the benchmarks GlueCOS [Khanuja et al. 2020] and LinCE [Aguilar et al. 2020], which are composed of code-switched text in various language combinations, such as Spanish-English and Modern Standard Arabic-Egyptian Arabic. It would be interesting to investigate whether the models are able to transfer their learned knowledge to other datasets, despite them being composed of different combinations of languages and may require additional fine-tuning.

Furthermore, the exploration of using the Moroccorp for different tasks, such as Part-of-speech tagging, Named Entity Recognition or Question answering is also interesting, as these tasks have practical applications in the real world, and are especially challenging for low-resource languages.

5.3 Summary and Conclusion

In this thesis I investigated the use of code-switching and the Moroccan ethnolect in a Moroccan-Dutch dataset (Moroccorp) that originates from a chat forum. Then, I researched the capabilities of transformer models to identify the use of these two phenomena using a token classification task.

I conducted a literature review on both the linguistic approach of code-switching in Section 2.1 as well as the origin and characteristics of the Moroccan ethnolect in Section 2.2. Then, I reviewed research approaches on the identification of code-switching using NLP models in Section 2.3 and researched the use of transformer architectures and transformer models that are suitable for Dutch in Section 2.4.

I randomly sampled a subset of Moroccorp to annotate on word-level using a set of labels that I selected in Section 3.2.1, including labels for three languages (NL, ENG, MOR), named entities (NAME), language independent utterances (NON), other languages (OTH) and spelling variations of Dutch (VAR). I explained the full process of this annotation in Section 3.2.2.

The annotated dataset was then analyzed and I measured how much code-switching occurs in Section 3.3. Both code-switching and the Moroccan ethnolect occur in the Moroccorp. Code-switching mostly happens between Dutch and Arabic or Berber, while English-Dutch code-switching is found frequently as well. Words from other languages are also found in the dataset, such as Turkish, French and Sranan Tongo. The frequency of code-switching (CMI = 10,92) is similar to other datasets on which research regarding code-switching is executed, like Mave et al. [2018] (CMI = 10,14). This is an indication that the Moroccorp is suitable for more research on code-switching, perhaps in the form of a more complicated task, like Question answering. As discussed in Section 2.2.3 and Section 2.2.4, the Moroccan ethnolect is characterized by insertions of certain Arabic function words, deliberate misspellings and wrong grammatical inflections of adjectives, all of which are found in the labeled dataset, as stated in Section 3.3.

I then used the annotated dataset for a custom classification task, which includes language identification as well as the identification of language independent utterances, named entitites and spelling variations of Dutch, using the labels previously mentioned. I fine-tuned two pre-trained transformer models on this dataset, RobBERT and Multilingual BERT. Additionally, I trained a logistic regression model as a baseline. The setup for the models is discussed in Chapter 4 and their evaluation and hyperparameter tuning in Section 4.3.

The transformer models perform the classification task relatively well, reaching scores (RobBERT: F1-score = 0,82; M-BERT: F1-score = 0,81) that are higher than the inter-annotator agreement (0,70). The transformer models reach scores that are significantly higher than the simple logistic regression baseline (F1-score = 0,53). Although the two transformer models reach no significant difference (p = 0,07) on the macro-average F1-score, there remain certain differences in performance. RobBERT reached a significantly higher accuracy score, mostly due to the large amount of Dutch words that it classifies correctly, as well as a significantly higher amount of spelling variations of Dutch that it recognizes. For the task of recognizing Moroccan languages, both models score similarly high. For the task of identifying part of the Moroccan ethnolect, in the form of VAR, RobBERT is the better option. Both models can be used for recognizing NL, ENG, NON and NAME.

Ultimately, both models are suitable for the task, but they may achieve higher performance on a more balanced dataset. Also, the task of simultaneously recognizing code-switching in combination with spelling variations of Dutch is complex due to the difference in nature of the two tasks and a strategic approach of focusing on each task individually in the future would yield more beneficial outcomes.

Bibliography

I. Adebara, A. Elmadany, M. Abdul-Mageed, and A. Inciarte. Dec. 2022. AfroLID: A neural language identification tool for African languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1958–1981. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates. https://aclanthology.org/2022.emnlp-main.128. DOI: 10.18653/v1/2022.emnlp-main.128.

G. Aguilar, S. Kar, and T. Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 1803–1813. European Language Resources Association, Marseille, France. ISBN 979-10-95546-34-4. https://aclanthology.org/2020.lrec-1.223.

J. Alammar. 2018. The illustrated transformer. *The Illustrated Transformer–Jay Alammar–Visualizing Machine Learning One Concept at a Time*, 27.

R. Appel. 1999. Straattaal; De mengtaal van jongeren in Amsterdam. *Toegepaste Taalwetenschap in Artikelen*, 62: 39–55.

P. Auer. 2020. *Language contact: Pragmatic factors*, pp. 147–167. Abingdon: Routledge.

K. Bali, J. Sharma, M. Choudhury, and Y. Vyas. Oct. 2014. "I am borrowing ya mixing ?" an analysis of English-Hindi code mixing in Facebook. In M. Diab, J. Hirschberg, P. Fung, and T. Solorio, eds., *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pp. 116–126. Association for Computational Linguistics, Doha, Qatar. https://aclanthology.org/W14-3914. DOI: 10.3115/v1/W14-3914.

O. Bardaï. 2003. *"Les Marocains résidant aux Pays-Bas: caractéristiques démographiques et sociales"*, p. 322–373. International Organization for Migration. https://web.archive.org/web/20060523220016/http://www.alwatan.ma/html/Publication_Fondation/Publication_2006/Publication/Ouvrage.pdf.

U. Barman, A. Das, J. Wagner, and J. Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pp. 13–23. Association for Computational Linguistics, Doha, Qatar. https://aclanthology.org/W14-3902. DOI: 10.3115/v1/W14-3902.

R. Barnett, E. Codó, E. Eppler, M. Forcadell, P. Gardner-Chloros, R. van Hout, M. Moyer, M. C. Torras, M. T. Turell, M. Sebba, M. Starren, and S. Wensing. 2000. The LIDES coding manual: A document for preparing and analyzing language interaction data version 1.1—july, 1999. *International Journal of Bilingualism*, 4(2): 131–132. https://doi.org/10.1177/13670069000040020101. DOI: 10.1177/13670069000040020101.

A. Bell. 2013. *The Guidebook to Sociolinguistics*, 1. Introducing Linguistics, 3. Wiley-Blackwell. ISBN 978-0631228660.

D. Blasi, A. Anastasopoulos, and G. Neubig, 2021. Systematic inequalities in language technology performance across the world's languages.

S. L. Blodgett and B. O'Connor. 2017. Racial disparity in natural language processing: A case study of social media african-american english. *CoRR*, abs/1707.00061. http://arxiv.org/abs/1707.00061.

S. L. Blodgett, L. Green, and B. O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1119–1130. Association for Computational Linguistics, Austin, Texas. https://aclanthology.org/D16-1120. DOI: 10.18653/v1/D16-1120.

Z. Bock. 2013. Cyber socialising: Emerging genres and registers of intimacy among young South African students. *Language Matters*, 44: 68–91. DOI: 10.1080/10228195.2013.784924.

S. Bonjour, 2009. Grens en gezin. Beleidsvorming inzake gezinsmigratie in Nederland, 1955-2005.

L. Boumans and E. Crevels, 2005. The Dutch bilingualism database.

A. Brandsen, A. Dirkson, S. Verberne, M. Sappelli, D. M. Chu, and K. Stoutjesdijk, 2019. BERT-NL a set of language models pretrained on the Dutch SoNaR corpus.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, 2020. Language models are few-shot learners. https://arxiv.org/abs/2005.14165.

CBS, 2023. Hoeveel mensen met een migratieachtergrond wonen in Nederland? https://www.cbs.nl/nl-nl/dossier/dossier-asiel-migratie-en-integratie/hoeveel-mensen-met-een-migratieachtergrond-wonen-in-nederland-.

J. Cheshire, P. Kerswill, S. P. Fox, and E. Torgersen. 2011. Contact, the feature pool and the speech community: The emergence of multicultural London English. *Journal of Sociolinguistics*, 15: 151–196.

L. Cornips and V. de Rooij. 2003. *Kijk, Levi's is een goeie merk: maar toch hadden ze 'm gedist van je schoenen doen 'm niet. Jongerentaal heeft de toekomst*, pp. 131–142. Bert Bakker, Nederland. ISBN 90 351 2571 1.

R. Dale. 2021. GPT-3: What's it good for? *Natural Language Engineering*, 27(1): 113–118. DOI: 10.1017/S1351324920000601.

E. Darics. 2013. Non-verbal signalling in digital discourse: the case of letter repetition. *Discourse, Context and Media*, 2: 141–148.

A. Das and B. Gambäck. 2014. Identifying languages at the word level in code-mixed Indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pp. 378–387. NLP Association of India, Goa, India. https://aclanthology.org/W14-5152.

P. Delobelle, T. Winters, and B. Berendt. 2020. RobBERT: a Dutch RoBERTa-based language model. In *Findings*.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota. https://aclanthology.org/N19-1423. DOI: 10.18653/v1/N19-1423.

L. Dixon, J. Li, J. S. Sorensen, N. Thain, and L. Vasserman. 2018. Measuring and mitigating unintended bias in text classification. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*.

M. Dorleijn and J. Nortier. 2009. *Code-switching and the internet*, p. 127–141. Cambridge Handbooks in Language and Linguistics. Cambridge University Press. DOI: 10.1017/CBO9780511576331.009.

M. Dorleijn, M. Mous, and J. Nortier. 2015. *Urban youth speech styles in Kenya and the Netherlands*, p. 271–289. Cambridge University Press. DOI: 10.1017/CBO9781139061896.019.

M. Dorleijn, M. Kossmann, and J. Nortier. 2020. *Urban youth speech styles in multilingual settings*, pp. 366–382. Abingdon and New York: Routledge.

A. El Aissati. 1997. *Language loss among native speakers of Moroccan-Arabic in the Netherlands*. Number 6 in Studies in Multilingualism. Tilburg University Press. ISBN 9036199573. (Bewerkte handelseditie dissertatie 1996) Pagination: 213.

A. El Aissati. 2002. *Verandering en verankering. Taal en identiteit bij jonge Marokkanen*, pp. 251–262. Aksant. Pagination: 12.

S. Esenjul. 2022. Code switching: Definition, types, and examples. *owlcation.com*. https://owlcation.com/humanities/Code-Switching-Definition-Types-and-Examples-of-Code-Switching.

M. Faruqui and C. Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 462–471. Association for Computational Linguistics, Gothenburg, Sweden. https://aclanthology.org/E14-1049. DOI: 10.3115/v1/E14-1049.

C. A. Ferguson. 1959. Diglossia. *The Bilingualism Reader*.

B. Gambäck. 2014. On measuring the complexity of code-mixing. In *Proceedings of the 11th International Conference on Natural Language Processing 1st Workshop on Language Technologies for Indian Social Media.*, p. 1–7. Goa, India.

B. Gambäck and A. Das. May 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1850–1855. European Language Resources Association (ELRA), Portorož, Slovenia. https://aclanthology.org/L16-1292.

F. Genesee. 2016. Shifting perspectives on bilingualism. In *Bilingualism across the lifespan: Factors moderating language proficiency*, pp. 9–19. American Psychological Association. DOI: 10.1037/14939-002.

E. M. Gold. 1967. Language identification in the limit. *Inf. Control.*, 10: 447–474.

S. Groenwold, L. Ou, A. Parekh, S. Honnavalli, S. Levy, D. Mirza, and W. Y. Wang. Nov. 2020. Investigating African-American Vernacular English in transformer-based text generation. In B. Webber, T. Cohn, Y. He, and Y. Liu, eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5877–5883. Association for Computational Linguistics, Online. https://aclanthology.org/2020.emnlp-main.473. DOI: 10.18653/v1/2020.emnlp-main.473.

S. A. Grondelaers, K. Deygers, H. V. Aken, V. V. D. Heede, and D. Speelman. 2000. Digitaal: het CONDIV-corpus geschreven Nederlands. *NEDERLANDSE TAALKUNDE (GRONINGEN)*, 5, no. 4: 356–63.

J. J. Gumperz. 1977. The sociolinguistic significance of conversational code-switching. *RELC Journal*, 8(2): 1–34. https://doi.org/10.1177/003368827700800201. DOI: 10.1177/003368827700800201.

G. A. Guzmán, J. Ricard, J. Serigos, B. E. Bullock, and A. J. Toribio. 2017. Moving code-switching research toward more empirically grounded methods. In *CDH@TLT*.

S. A. Hale. 2014. Global connectivity and multilinguals in the Twitter network. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

E. I. Haugen. 1950. The analysis of linguistic borrowing. *Language*, 26: 210.

J. Henrich, S. J. Heine, and A. Norenzayan. 2010. The weirdest people in the world? *Behavioral and Brain Sciences*, 33(2-3): 61–83. DOI: 10.1017/S0140525X0999152X.

F. Hinskens. 2011. Emerging Moroccan and Turkish varieties of Dutch: Ethnolects or ethnic styles? *Ethnic Styles of Speaking in European Metropolitan Areas*, 8: 101–129.

D. Hovy and A. Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 483–488. Association for Computational Linguistics, Beijing, China. https://aclanthology.org/P15-2079. DOI: 10.3115/v1/P15-2079.

D. Hovy and S. L. Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 591–598. Association for Computational Linguistics, Berlin, Germany. https://aclanthology.org/P16-2096. DOI: 10.18653/v1/P16-2096.

D.-M. Iliescu, R. Grand, S. Qirko, and R. van der Goot. 2021. Much gracias: Semi-supervised code-switch detection for Spanish-English: How far can we get? In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pp. 65–71. Association for Computational Linguistics, Online. https://aclanthology.org/2021.calcs-1.9. DOI: 10.18653/v1/2021.calcs-1.9.

J. James, V. Yogarajan, I. Shields, C. Watson, P. Keegan, K. Mahelona, and P.-L. Jones. 2022. Language models for code-switch detection of te reo Māori and English in a low-resource setting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 650–660. Association for Computational Linguistics, Seattle, United States. https://aclanthology.org/2022.findings-naacl.49. DOI: 10.18653/v1/2022.findings-naacl.49.

T. Jauhiainen, M. Lui, M. Zampieri, T. Baldwin, and K. Lindén. 2018. Automatic language identification in texts: A survey. *J. Artif. Intell. Res.*, 65: 675–782.

A. Jørgensen, D. Hovy, and A. Søgaard. 2015. Challenges of studying and processing dialects in social media. In *Proceedings of the Workshop on Noisy User-generated Text*, pp. 9–18. Association for Computational Linguistics, Beijing, China. https://aclanthology.org/W15-4302. DOI: 10.18653/v1/W15-4302.

N. Jose, B. R. Chakravarthi, S. Suryawanshi, E. Sherly, and J. P. McCrae. 2020. A survey of current datasets for code-switching research. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 136–141.

P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury. July 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6282–6293. Association for Computational Linguistics, Online. https://aclanthology.org/2020.acl-main.560. DOI: 10.18653/v1/2020.acl-main.560.

D. Jurgens, Y. Tsvetkov, and D. Jurafsky. 2017. Incorporating dialectal variability for socially equitable language identification. In *Proceedings of the 55th Annual Meeting of the Association* *for Computational Linguistics (Volume 2: Short Papers)*, pp. 51–57. Association for Computational Linguistics, Vancouver, Canada. https://aclanthology.org/P17-2009. DOI: 10.18653/v1/P17-2009.

S. Khanuja, S. Dandapat, A. Srinivasan, S. Sitaram, and M. Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3575–3585. Association for Computational Linguistics, Online. https://aclanthology.org/2020.acl-main.329. DOI: 10.18653/v1/2020.acl-main.329.

P. Kodali, A. Goel, M. Choudhury, M. Shrivastava, and P. Kumaraguru. May 2022. SyMCoM - syntactic measure of code mixing a study of English-Hindi code-mixing. In S. Muresan, P. Nakov, and A. Villavicencio, eds., *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 472–480. Association for Computational Linguistics, Dublin, Ireland. https://aclanthology.org/2022.findings-acl.40. DOI: 10.18653/v1/2022.findings-acl.40.

M. Kossmann. 2017. Key and the use of Moroccan function words in Dutch internet discourse. *Nederlandse Taalkunde*, 22: 223–248. DOI: 10.5117/NEDTAA2017.2.KOSS.

M. Kossmann. 2019. Is Dutch straattaal a mixed multiethnolect? a Moroccan perspective. *Appl. Linguist. Rev.*, 10(3): 293–316.

G.-A. Levow, E. Ahn, and E. M. Bender. Mar. 2021. Developing a shared task for speech processing on endangered languages. In *Proceedings of the 4th Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, pp. 96–106. Association for Computational Linguistics, Online. https://aclanthology.org/2021.computel-1.12.

D. Li. 2000. Cantonese-English code-switching research in Hong Kong: A Y2K review. *World Englishes*, 19: 305 – 322. DOI: 10.1111/1467-971X.00181.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, 2019. RoBERTa: A robustly optimized BERT pretraining approach. https://arxiv.org/abs/1907.11692.

I. Loshchilov and F. Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bkg6RiCqY7.

M. Lui and T. Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pp. 25–30. Association for Computational Linguistics, Jeju Island, Korea. https://aclanthology.org/P12-3005.

T. Luong, H. Pham, and C. D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pp. 151–159. Association for Computational Linguistics, Denver, Colorado. https://aclanthology.org/W15-1521. DOI: 10.3115/v1/W15-1521.

T. Lynn and K. Scannell. Aug. 2019. Code-switching in Irish tweets: A preliminary analysis. In T. Lynn, D. Prys, C. Batchelor, and F. Tyers, eds., *Proceedings of the Celtic Language Technology Workshop*, pp. 32–40. European Association for Machine Translation, Dublin, Ireland. https://aclanthology.org/W19-6905.

M. Mager, Ö. Çetinoğlu, and K. Kann. 2019. Subword-level language identification for intra-word code-switching. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2005–2011. Association for Computational Linguistics, Minneapolis, Minnesota. https://aclanthology.org/N19-1201. DOI: 10.18653/v1/N19-1201.

S. Maharjan, E. Blair, S. Bethard, and T. Solorio. 2015. Developing language-tagged corpora for code-switching tweets. In *Proceedings of the 9th Linguistic Annotation Workshop*, pp. 72–84. Association for Computational Linguistics, Denver, Colorado, USA. https://aclanthology.org/W15-1608. DOI: 10.3115/v1/W15-1608.

A. Mann and A. de Bruin. 2021. Bilingual language use is context dependent: using the language and social background questionnaire to assess language experiences and test-rest reliability. *International Journal of Bilingual Education and Bilingualism*, 25: 1–16. DOI: 10.1080/13670050.2021.1988049.

L. Martin, B. Muller, P. J. O. Suá rez, Y. Dupont, L. Romary, É. de la Clergerie, D. Seddah, and B. Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. https://doi.org/10.18653%2Fv1%2F2020.acl-main.645. DOI: 10.18653/v1/2020.acl-main.645.

D. Mave, S. Maharjan, and T. Solorio. July 2018. Language identification and analysis of code-switched social media text. In G. Aguilar, F. AlGhamdi, V. Soto, T. Solorio, M. Diab, and J. Hirschberg, eds., *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pp. 51–61. Association for Computational Linguistics, Melbourne, Australia. https://aclanthology.org/W18-3206. DOI: 10.18653/v1/W18-3206.

T. Mcarthur. 1992. *Concise Oxford Companion to the English Language*. Oxford University Press.

A. McCallum and K. Nigam. 1998. A comparison of event models for Naive Bayes text classification. In *AAAI Conference on Artificial Intelligence*.

P. McNamee. 2005. Language identification: a solved problem suitable for undergraduate instruction. *Journal of Computing Sciences in Colleges*, 20: 94–101.

E. Á. Mellado and C. Lignos. 2022. Borrowing or codeswitching? annotating for finer-grained distinctions in language mixing. In *International Conference on Language Resources and Evaluation*.

G. Molina, F. AlGhamdi, M. Ghoneim, A. Hawwari, N. Rey-Villamizar, M. Diab, and T. Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pp. 40–49. Association for Computational Linguistics, Austin, Texas. https://aclanthology.org/W16-5805. DOI: 10.18653/v1/W16-5805.

C. Myers-Scotton. 1995. Duelling languages: Grammatical structure in codeswitching. *Studies in Second Language Acquisition*, 17(1): 117–118. DOI: 10.1017/S027226310001408X.

C. Myers-Scotton. 1997. *Code-Switching*, chapter 13, pp. 217–237. John Wiley & Sons, Ltd. ISBN 9781405166256. https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405166256.ch13. DOI: https://doi.org/10.1002/9781405166256.ch13.

H. Nakayama, T. Kubo, J. Kamura, Y. Taniguchi, and X. Liang, 2018. Doccano: Text annotation tool for human. https://github.com/doccano/doccano. Software available from https://github.com/doccano/doccano.

R. Negrón. 2009. Spanish-English codeswitching in email communication. *Language@Internet*, 6.

D. Nguyen and L. Cornips. 2016. Automatic detection of intra-word code-switching. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 82–86. Association for Computational Linguistics, Berlin, Germany. https://aclanthology.org/W16-2013. DOI: 10.18653/v1/W16-2013.

D. Nguyen and A. S. Doğruöz. Oct. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 857–862. Association for Computational Linguistics, Seattle, Washington, USA. https://aclanthology.org/D13-1084.

J. Nortier. 2016. Characterizing urban youth speech styles in Utrecht and on the internet. *Journal of Language Contact*, 9(1): 163–185. DOI: 10.1163/19552629-00901007.

J. Nortier. 2018. Language and identity practices among multilingual Western European youths. *Language and Linguistics Compass*, 12. DOI: 10.1111/lnc3.12278.

J. Nortier and M. Dorleijn. 2008. A Moroccan accent in Dutch: A sociocultural style restricted to the Moroccan community? *International Journal of Bilingualism - INT J BILING*, 12: 125–142. DOI: 10.1177/13670069080120010801.

J. Nortier and M. Dorleijn. 2013. Multi-ethnolects: Kebabnorsk, Perkerdansk, Verlan, Kanakensprache, Straattaal, etc. In *Contact Languages*, pp. 229–272. DE GRUYTER, Berlin, Boston.

S. Nowak and S. Rüger. 2010. How reliable are annotations via crowdsourcing: A study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the International Conference on Multimedia Information Retrieval*, MIR '10, p. 557–566. Association for Computing Machinery, New York, NY, USA. ISBN 9781605588155. https://doi.org/10.1145/1743384.1743478. DOI: 10.1145/1743384.1743478.

P. J. Ortiz Suárez, L. Romary, and B. Sagot. 2020. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1703–1714. Association for Computational Linguistics, Online. https://www.aclweb.org/anthology/2020.acl-main.156.

E. Papalexakis, D. Nguyen, and A. S. Doğruöz. Oct. 2014. Predicting code-switching in multilingual communication for immigrant communities. In M. Diab, J. Hirschberg, P. Fung, and T. Solorio, eds., *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pp. 42–50. Association for Computational Linguistics, Doha, Qatar. https://aclanthology.org/W14-3905. DOI: 10.3115/v1/W14-3905.

T. Pires, E. Schlinger, and D. Garrette. 2019. How multilingual is multilingual BERT? *CoRR*, abs/1906.01502. http://arxiv.org/abs/1906.01502.

S. Poplack. 1980. Sometimes I'll start a sentence in Spanish y termino en español: toward a typology of code-switching. *Linguistics*, 18: 581–618. DOI: 10.1515/ling.1980.18.7-8.581.

S. Poplack and N. Dion. 2012. Myths and facts about loanword development. *Language Variation and Change*, 24: 279 – 315.

A. Pratapa, M. Choudhury, and S. Sitaram. 2018. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3067–3072. Association for Computational Linguistics, Brussels, Belgium. https://aclanthology.org/D18-1344. DOI: 10.18653/v1/D18-1344.

P. Quist. 2008. Sociolinguistic approaches to multiethnolect: Language variety and stylistic practice. *International Journal of Bilingualism - INT J BILING*, 12: 43–61. DOI: 10.1177/13670069080120010401.

S. Rijhwani, R. Sequiera, M. Choudhury, K. Bali, and C. S. Maddila. 2017. Estimating code-switching on Twitter with a novel generalized word-level language detection technique. In *Proceedings* *of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1971–1982. Association for Computational Linguistics, Vancouver, Canada. https://aclanthology.org/P17-1180. DOI: 10.18653/v1/P17-1180.

A. Rios. 2020. Fuzze: Fuzzy fairness evaluation of offensive language classifiers on African-American English. In *AAAI Conference on Artificial Intelligence*.

A. Rosowsky. 2010. 'writing it in English': script choices among young multilingual muslims in the UK. *Journal of Multilingual and Multicultural Development*, 31: 163 – 179.

R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal. 2013. Overview of the fire 2013 track on transliterated search. In *Proceedings of the 4th and 5th Annual Meetings of the Forum for Information Retrieval Evaluation*, FIRE '12 & '13. Association for Computing Machinery, New York, NY, USA. ISBN 9781450328302. https://doi.org/10.1145/2701336.2701636. DOI: 10.1145/2701336.2701636.

T. Ruette and F. Van de Velde. 2013. Moroccorp: tien miljoen woorden uit twee Marokkaans-Nederlandse chatkanalen. *Lexikos*, 23: 456–475. DOI: 10.5788/23-1-1225.

Y. Samih, S. Maharjan, M. Attia, L. Kallmeyer, and T. Solorio. 2016. Multilingual code-switching identification via LSTM recurrent neural networks. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pp. 50–59. Association for Computational Linguistics, Austin, Texas. https://aclanthology.org/W16-5806. DOI: 10.18653/v1/W16-5806.

H. San. 2009. *Chinese-English Code-switching in Blogs by Macao Young People*. Master's thesis, The University of Edinburgh.

H. Schmeets and L. Cornips. 2021. Talen en dialecten in Nederland: Wat spreken we thuis en wat schrijven we op sociale media? *Centraal Bureau voor de Statistiek*. https://www.cbs.nl/nl-nl/longread/statistische-trends/2021/talen-en-dialecten-in-nederland.

P. Schumacher. 1987. *De minderheden: 700.000 migranten minder gelijk*. Amsterdam: Van Gennep.

M. Sebba. 2003. *Spelling rebellion.*, pp. 151–172. Pragmatics & Beyond. John Benjamins, 110. ISBN 1588113558.

R. Sennrich, B. Haddow, and A. Birch. Aug. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany. https://aclanthology.org/P16-1162. DOI: 10.18653/v1/P16-1162.

M. H. Shakeel, A. Karim, and I. Khan. 2019. A multi-cascaded deep model for bilingual SMS classification. In *Neural Information Processing*, pp. 287–298. Springer International Publishing. https://doi.org/10.1007%2F978-3-030-36708-4_24.

P. Shaw. 2008. Spelling, accent and identity in computer-mediated communication. *English Today*, 24(2): 42–49. DOI: 10.1017/S0266078408000199.

S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black. 2019. A survey of code-switched speech and language processing. *CoRR*, abs/1904.00784. http://arxiv.org/abs/1904.00784.

T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Ghoneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang, and P. Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pp. 62–72. Association for Computational Linguistics, Doha, Qatar. https://aclanthology.org/W14-3907. DOI: 10.3115/v1/W14-3907.

D. Speelman, S. Grondelaers, and D. Geeraerts. 2008. *Variation in the choice of adjectives in the two main national varieties of Dutch*, pp. 205–233. De Gruyter Mouton.

T. Sun, A. Gaut, S. Tang, Y. Huang, M. ElSherief, J. Zhao, D. Mirza, E. Belding, K.-W. Chang, and W. Y. Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1630–1640. Association for Computational Linguistics, Florence, Italy. https://aclanthology.org/P19-1159. DOI: 10.18653/v1/P19-1159.

B. A. Svendsen. 2015. *Language, youth and identity in the 21st century: content and continuations*, p. 3–23. Cambridge University Press. DOI: 10.1017/CBO9781139061896.002.

S. Thara and P. Poornachandran. 2021. Transformer based language identification for Malayalam-English code-mixed text. *IEEE Access*, 9: 118837–118850. DOI: 10.1109/ACCESS.2021.3104106.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, 2017. Attention is all you need. https://arxiv.org/abs/1706.03762.

J. De Vries. 2005. ”Indisch-Nederlands”. In *Wereldnederlands. Oude en jonge variëteiten van het Nederlands, N. van der Sijs (ed.)*, pp. 59–78. Den Haag: SDU.

W. De Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim. 2019. BERTje: A Dutch BERT model. *CoRR*, abs/1912.09582. http://arxiv.org/abs/1912.09582.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJ4km2R5t7.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45. Association for Computational Linguistics, Online. https://aclanthology.org/2020.emnlp-demos.6. DOI: 10.18653/v1/2020.emnlp-demos.6.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. R. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. S. Corrado, M. Hughes, and J. Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, 2019. XLNet: Generalized autoregressive pretraining for language understanding. https://arxiv.org/abs/1906.08237.

H. Ziemann, F. Weerman, and E. Ruigendijk. 2011. Nederlands later geleerd: gebruik van lidwoorden en flexie van bijvoeglijke naamwoorden door Duitstalige kinderen en volwassenen. *Internationale Neerlandistiek*, 49: 183–207. DOI: 10.5117/IVN2011.3.ZIEM.

C. Ziems, J. Chen, C. Harris, J. Anderson, and D. Yang. 2022. VALUE: Understanding dialect disparity in NLU. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3701–3720. Association for Computational Linguistics, Dublin, Ireland. https://aclanthology.org/2022.acl-long.258. DOI: 10.18653/v1/2022.acl-long.258.