Automatic Detection of Intra-Word Code-Switching

Dong Nguyen[12] **Leonie Cornips**[23]
[1]Human Media Interaction, University of Twente, Enschede, The Netherlands
[2] Meertens Institute, Amsterdam, The Netherlands
[3]Maastricht University, Maastricht, The Netherlands
d.nguyen@utwente.nl, leonie.cornips@meertens.knaw.nl

Abstract

Many people are multilingual and they may draw from multiple language varieties when writing their messages. This paper is a first step towards analyzing and detecting code-switching within words. We first segment words into smaller units. Then, words are identified that are composed of sequences of subunits associated with different languages. We demonstrate our method on Twitter data in which both Dutch and dialect varieties labeled as Limburgish, a minority language, are used.

1 Introduction

Individuals have their own linguistic repertoire from which they can draw elements or codes (e.g., language varieties). In both spoken and written communication, multilingual speakers may use multiple languages in a single conversation, for example within a turn or even within a syntactic unit, often referred to as intra- and extra-sentential code-switching.

In online communication the usage of multiple languages is also prevalent. Over 10% of the Twitter users tweet in more than one language (Hale, 2014) and code-switching has been observed on various social media platforms as well (Androutsopoulos, 2013; Johnson, 2013; Jurgens et al., 2014; Nguyen et al., 2015). The occurrence of code-switching in online communication has sparked interest in two research directions.

First, the presence of code-switching in text introduces new challenges for NLP tools, since these tools are usually designed for texts written in a single language. Recently, various studies have focused on automatic language identification at a more-fine grained level, such as words instead of documents (Solorio et al., 2014), to facilitate the processing of such texts. Several studies have adapted NLP tools for code-switched texts (e.g., Solorio and Liu (2008) and Peng et al. (2014)).

Second, the availability of social media data has enabled studying code-switching patterns in a multitude of social situations and on a larger scale than datasets collected using more traditional methods. To fully leverage these large amounts of data, several recent studies have employed automatic language identification to study code-switching patterns in social media (Kim et al., 2014; Nguyen et al., 2015).

Research in both these directions has so far studied code-switching by assigning concrete languages to messages or individual words. However, the notion of *languages* or *a language* implies that languages are concrete, stable, countable identities that can be distinguished unproblematically from each other. In reality, however, people use *language*: linguistic resources (features, items, nouns, morphemes, etc.) that are recognized by the speakers or others as belonging to two or more sets of resources (Jørgensen and Juffermans, 2011). From this perspective, code-switching can thus occur *within* words. For example, in *oetverkocht* ‘sold out’, the particle *oet* ‘out’ is used that is associated with Limburgish whereas *verkocht* ‘sold’ is associated with Dutch.

This study is a first step towards detecting code-switching within words using computational methods, which could support the processing of code-switched texts and support sociolinguists in their study of code-switching patterns. We focus on tweets from a province in the Netherlands where a minority language is spoken alongside Dutch (see Section 3). We automatically segment the words into smaller units using the Morfessor tool (Section 4). We then identify words with subunits that are associated with different languages (Section 5).

Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 82–86,
Berlin, Germany, August 11, 2016. ©2016 Association for Computational Linguistics

2 Related Work

This paper builds on research on morphology and automatic language identification.

Morphology We focus on tweets written in Limburg, a province in the Netherlands. Morphological analysis for Dutch using computational approaches has been the focus in several studies. Van den Bosch and Daelemans (1999) proposed a memory-based learning approach. Cast as a classification problem, morpheme boundaries were detected based on letter sequences. De Pauw et al. (2004) built on this work and compared a memory-based learning method with a finite state method. One of the characteristic features of Dutch is diminutive formation (Trommelen, 1983) and computational approaches have been explored to predict the correct diminutive suffix in Dutch (Daelemans et al., 1996; Kool et al., 2000).

McArthur (1998) identified four major types of code-switching, ranging from tag-switching (tags and set of phrases) to intra-word switching, where a change occurs within a word boundary. The occurrence of intra-word switching has only been rarely addressed in computational linguistics research. Habash et al. (2005) developed a morphological analyzer and generator for the Arabic language family. The tool allows combining morphemes from different dialects.

Language Identification The prevalence of code-switching in online textual data has generated a renewed interest in automatic language identification. Instead of focusing on document level classification, recent studies have focused on language identification on a word level to support the analysis and processing of code-switched texts (Nguyen and Doğruöz, 2013). In the First Shared Task on Language Identification in Code-Switched Data (Solorio et al., 2014), a small fraction of the words were labeled as '*mixed*', indicating that these words were composed of morphemes from different languages. However, many participating systems had very low performance, i.e., zero F-scores, on this particular category (Chittaranjan et al., 2014; Jain and Bhat, 2014; Bar and Dershowitz, 2014; Shrestha, 2014). Oco and Roxas (2012) focused on detecting code-switching points and noted that intra-word code-switching caused difficulties to a dictionary based approach. In this study, we segment words into smaller units to detect intra-word code-switching.

3 Dataset

We confine our analysis to tweets from users in the Dutch province of Limburg, the southernmost province in the Netherlands. The 'dialects' of Limburg were extended minor recognition in 1997 under the label 'Limburgish' by The Netherlands, a signatory of the 1992 European Charter for Regional and Minority Languages (cf. Cornips (2013)). To collect users located in Limburg, seed users were identified based on geotagged tweets and manual identification. The set was then expanded based on the social network of the users. Users were then mapped to locations based on their provided profile location to create the final set. Tweets are labeled with languages, such as Dutch, Limburgish, and English, using an inhouse language identification tool. The dataset is described in more detail in Nguyen et al. (2015).

4 Morphological Segmentation

The first step in our analysis is to segment the words into smaller units. We use the Morfessor Baseline implementation (Virpioja et al., 2013) to learn a model for what is called morphological segmentation in an unsupervised manner. Morfessor segments the words into *morphs* (usually 'morpheme-like' units), such that words in the data can be formed by concatenation of such morphs.

Training We experiment with two different sources to train Morfessor: tweets and Wikipedia texts. The tweets come from the data described in Section 3. We also downloaded the Dutch and Limburgish Wikipedia versions. More specifically, we have the following datasets:

- Dutch Wikipedia (**NL_WIKI**)
- Limburgish Wikipedia (**LIM_WIKI**)
- Dutch tweets (**NL_TWEETS**)
- Limburgish tweets (**LIM_TWEETS**)

We exclude words that only occur once. Following Creutz and Lagus (2005), we explore two different ways for training Morfessor: based on word tokens (such that the frequencies of words are taken into account) and based on word types. Creutz and Lagus (2005) suggest using word types, which in their experiments led to a higher recall.

		Dutch				Limburgish			
		Word tokens		Word Types		Word tokens		Word Types	
Data	#types	P	R	P	R	P	R	P	R
NL_WIKI	1,377,658	0.976	0.681	0.842	0.765	0.805	0.745	0.662	0.812
LIM_WIKI	68,255	0.743	0.806	0.559	0.867	0.752	**0.788**	0.586	**0.839**
NL_TWEETS	115,319	0.968	0.685	0.833	0.779	0.893	0.745	0.627	0.818
LIM_TWEETS	37,054	0.867	**0.757**	0.648	**0.874**	**0.956**	0.711	0.665	0.826
TWEETS + WIKI	1,460,724	**0.985**	0.674	**0.871**	0.747	0.955	0.689	**0.827**	0.771

Table 1: Results of morphological segmentation using Morfessor, reporting Precision (P) and Recall (R)

Evaluation To evaluate the performance of Morfessor on the Twitter data we randomly annotated a set of tweets attributed to either Dutch or Limburgish, resulting in 330 words from Dutch tweets and 312 words from Limburgish tweets. Table 1 reports the precision and recall as calculated by the Morfessor tool. Overall, the performance differences are small. The best performance is obtained when Limburgish data is part of the training data. Furthermore, training on word tokens results in a higher precision, while training on word types results in a higher recall, matching the findings of Creutz and Lagus (2005).

An analysis of the resulting segmentations in the Twitter data illustrates this even more. We consider models trained on both the Wikipedia and Twitter data. A model trained on word tokens segments only 23.5% of the words, while a model trained on word types segments 71.4% of the words. For our application, a higher recall is preferred, and thus following Creutz and Lagus (2005) we use a model trained on word types in the remaining part of this paper. Example segmentations using this model are *rogstaekersoptocht* as *rogstaeker+s+optocht* ‘carnivalsname+s+parade’, *leedjesaovend* as *leedjes+aovend* ‘songs+evening’ and *zoemetein* as *zoe+metein* ‘immediately’.

5 Detection of Intra-Word Code-Switching

We now identify code-switching within words based on the extracted morphs (e.g., morphemes, particles, bare nouns and character sequences).

5.1 Language Identification

To identify code-switching within words, we first compute the association of the morphs with Dutch and Limburgish. For illustration, we separate morphs that occur on their own in the data from morphs that only occur in combination with other morphs. For each morph, we compute its probability in each language (Dutch and Limburgish) and apply Laplace smoothing. For each morph, the odds ratio is then computed as follows (Mladenic and Grobelnik, 1999), with m being the morph:

$$log(\frac{P(m|NL)(1-P(m|LIM))}{(1-P(m|NL))(P(m|LIM)}) \qquad (1)$$

3	LIM	*roë,wêr, sjw, lië, pke*
	NL	*pje, ful, cre, ary, ica*
4	LIM	*wari, ônne, blié, gesj, tere*
	NL	*isme, tttt, pppp, gggg, oool*
5	LIM	*oetge, raods, telik, erlik, aafge*
	NL	*uitge, erweg, eloos, logie, zwerf*

Table 2: Most distinguishing morphs with lengths 3-5 that do not occur on their own, for Dutch (NL) and Limburgish (LIM) according to the odds ratio.

Since the odds ratio is sensitive to infrequent words, only morphs were considered that occur in at least 5 words. Table 2 displays the most distinguishing morphs that do not occur on their own. While some of the extracted morphs are not strictly morphemes but grapheme sequences, they do seem to reflect the differences between the Dutch and Limburgish language. One example is reflected in *pje* and *pke*. The diminutive *je* is associated with Dutch, while *ke* is associated with Limburgish. We also see the frequent use of diacritics, characteristic of Limburgish orthography. The results are also affected by the informal nature of social media, such as the use of lengthening in the extracted morphs (e.g., *oool*). Furthermore, the occurrence of English words has led to morphs like *ary* (from, e.g., *anniversary*) and *ful*. We also

3	LIM	*oèt, veu, iér, vuu, ôch*
	NL	*gro, hor, cal, tec, ish*
4	LIM	*hoëg, kaup, roop, stök, zurg*
	NL	*rook, rouw, uuuu, ship, doek*
5	LIM	*slaop, sjaol, paort, hoeëg, rieje*
	NL	*fonds, dorps, kruis, kraam, keten*

Table 3: Most distinguishing morphs with lengths 3-5 that do occur on their own, for Dutch (NL) and Limburgish (LIM) according to the odds ratio.

see *oetge* and *uitge* where *oet* 'out' is associated with Limburgish. Table 3 shows the distinguishing morphs that do occur on their own. In this table we find many units that are bare nouns, like *rook* ('smoke'), *rouw* ('mourning'), etc.

5.2 Identified Words

Since many words are cognates in Dutch and Limburgish, we apply a strict threshold to assign the extracted units to a single language (1.5 and -1.5 odds ratio). We then extract all words that are composed of sequences of units that are associated with different languages.

Results In total 50 words were identified. We manually checked whether they were correct, and if not, the type of error that was made (Table 4). Since Limburgish is a label for various dialect varieties, we consulted several sources to determine the Limburgish form(s).[1]

Type	Freq	%
Correct	17	34%
Error: name	15	30%
Error: concatenation	2	4%
Error: English	2	4%
Error: spelling mistake	2	4%
Error: other	12	24%

Table 4: Evaluation of the identified words.

An example of an identified word with code-switching is *cijfer* + *kes* 'small numbers'. The Limburgish plural diminutive *kes* is combined with the Dutch noun *cijfer* 'number' whereas /ˈsiːfəʀ/ is associated with Limburgish. As another example, in *sjlaag* + *boom* ('crossing gate') Limburgish *sjlaag* (palatized /s/) is combined with Dutch *boom* (instead of /bɔːm/).

Error analysis Manual inspection of the identified words shows that the informal nature of the Twitter data makes the task challenging. In particular, spelling mistakes (e.g., *woendag* 'Wednesday' instead of *woensdag*), the occurrence of English words (e.g., *wearable*), and concatenated words (e.g., *kleiduivenschieten* instead of *kleiduiven schieten*) were sometimes incorrectly identified as words containing code-switching. Furthermore, most of the errors were names that were incorrectly identified (*prinsestraat*, *kleistekerstraat*). We therefore expect that more preprocessing, like removing named entities, would improve the system.

6 Conclusion

Research using automatic language identification to study code-switching patterns has so far focused on assigning languages to messages or individual words (Nguyen et al., 2016). This study is a first step towards automatic language identification and analysis of code-switching patterns within words. Our experiments demonstrate that Twitter users do code-switch within words and are creative in their language use by combining elements from both the standard language (Dutch) and the minority language (Limburgish).

The precision of the system could be improved by applying more preprocessing steps, such as filtering named entities. Evaluation was challenging due to the difficulty of labeling languages on such a fine-grained level as the extracted morphs. In particular, when focusing on minority languages such as Limburgish for which no standard exists and which shares many cognates with Dutch, it is not always clear whether a certain variant is associated with Dutch, Limburgish, or both. A future study could focus on a more extensive evaluation of the system.

Acknowledgements

This research was supported by the Netherlands Organization for Scientific Research (NWO), grants 314-98-008 (Twidentity) and 640.005.002 (FACT).

References

Jannis Androutsopoulos. 2013. Code-switching in computer-mediated communication. In *Pragmatics of Computer-Mediated Communication*. De Gruyter Mouton.

[1] eWND (www.meertens.knaw.nl/dialectwoordenboeken/), WLD (dialect.ruhosting.nl/wld/zoeken_materiaalbases.html) and Limburghuis (www.limburghuis.nl/).

Kfir Bar and Nachum Dershowitz. 2014. The Tel Aviv university system for the code-switching workshop shared task. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using CRF: Code-switching shared task report of MSR India system. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Leonie Cornips. 2013. Recent developments in the Limburg dialect region. In Frans Hinskens and Johan Taeldeman, editors, *Language and Place. An International Handbook of Linguistic Variation*. De Gruyter Mouton.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Walter Daelemans, Peter Berck, and Steven Gillis. 1996. Unsupervised discovery of phonological categories through supervised learning of morphological rules. In *Proceedings of COLING 1996*.

Guy De Pauw, Tom Laureys, Walter Daelemans, and Hugo Van hamme. 2004. A comparison of two different approaches to morphological analysis of Dutch. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology*.

Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological analysis and generation for Arabic dialects. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*.

Scott A. Hale. 2014. Global connectivity and multilinguals in the Twitter network. In *CHI '14*.

Naman Jain and Riyaz Ahmad Bhat. 2014. Language identification in code-switching scenario. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Ian Johnson. 2013. Audience design and communication accommodation theory: Use of Twitter by Welsh-English biliterates. In *Social Media and Minority Languages: Convergence and the Creative Industries*. Multilingual Matters.

J. Normann Jørgensen and Kasper Juffermans. 2011. Languaging.

David Jurgens, Stefan Dimitrov, and Derek Ruths. 2014. Twitter users #codeswitch hashtags! #moltoimportante #wow. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Suin Kim, Ingmar Weber, Li Wei, and Alice Oh. 2014. Sociolinguistic analysis of Twitter in multilingual societies. In *Proceedings of the 25th ACM conference on Hypertext and social media*.

Anne Kool, Walter Daelemans, and Jakub Zavrel. 2000. Genetic algorithms for feature relevance assignment in memory-based language processing. In *Proceedings of CoNLL-2000 and LLL-2000*.

Tom McArthur. 1998. Code-mixing and code-switching. Concise Oxford companion to the English language.

Dunja Mladenic and Marko Grobelnik. 1999. Feature selection for unbalanced class distribution and Naive Bayes. In *Proceedings of ICML '99*.

Dong Nguyen and A. Seza Doğruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of EMNLP 2013*.

Dong Nguyen, Dolf Trieschnigg, and Leonie Cornips. 2015. Audience and the use of minority languages on Twitter. In *Proceedings of ICWSM 2015*.

Dong Nguyen, A. Seza Doğruöz, Carolyn P. Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *To appear in Computational Linguistics*.

Nathaniel Oco and Rachel Edita Roxas. 2012. Pattern matching refinements to dictionary-based code-switching point detection. In *PACLIC 26*.

Nanyun Peng, Yiming Wang, and Mark Dredze. 2014. Learning polylingual topic models from code-switched social media documents. In *ACL 2014*.

Prajwol Shrestha. 2014. Incremental n-gram approach for language identification in code-switched text. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for English-Spanish code-switched text. In *Proceedings of EMNLP 2008*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale 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*.

Mieke Trommelen. 1983. *The Syllable in Dutch*. Walter de Gruyter.

Antal Van den Bosch and Walter Daelemans. 1999. Memory-based morphological analysis. In *Proceedings of ACL 1999*.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor baseline.