Multilingual Code-switching Identification via LSTM Recurrent Neural Networks

Younes Samih
Dept. of Computational Linguistics
Heinrich Heine University,
Düsseldorf, Germany
samih@phil.hhu.de

Suraj Maharjan
Dept. of Computer Science
University of Houston
Houston, TX, 77004
smaharjan2@uh.edu

Mohammed Attia
Google Inc.
New York City
NY, 10011
attia@google.com

Laura Kallmeyer
Dept. of Computational Linguistics
Heinrich Heine University,
Düsseldorf, Germany
kallmeyer@phil.hhu.de

Thamar Solorio
Dept. of Computer Science
University of Houston
Houston, TX, 77004
solorio@cs.uh.edu

Abstract

This paper describes the HHU-UH-G system submitted to the EMNLP 2016 Second Workshop on Computational Approaches to Code Switching. Our system ranked first place for Arabic (MSA-Egyptian) with an F1-score of 0.83 and second place for Spanish-English with an F1-score of 0.90. The HHU-UH-G system introduces a novel unified neural network architecture for language identification in code-switched tweets for both Spanish-English and MSA-Egyptian dialect. The system makes use of word and character level representations to identify code-switching. For the MSA-Egyptian dialect the system does not rely on any kind of language-specific knowledge or linguistic resources such as, Part Of Speech (POS) taggers, morphological analyzers, gazetteers or word lists to obtain state-of-the-art performance.

1 Introduction

Code-switching can be defined as the act of alternating between elements of two or more languages or language varieties within the same utterance. The main language is sometimes referred to as the 'host language', and the embedded language as the 'guest language' (Yeh et al., 2013). Code-switching is a wide-spread linguistic phenomenon in modern informal user-generated data, whether spoken or written. With the advent of social media, such as Facebook posts, Twitter tweets, SMS messages, user comments on the articles, blogs, etc., this phenomenon is becoming more pervasive. Code-switching does not only occur across sentences (inter-sentential) but also within the same sentence (intra-sentential), adding a substantial complexity dimension to the automatic processing of natural languages (Das and Gambäck, 2014). This phenomenon is particularly dominant in multilingual societies (Milroy and Muysken, 1995), migrant communities (Papalexakis et al., 2014), and in other environments due to social changes through education and globalization (Milroy and Muysken, 1995). There are also some social, pragmatic and linguistic motivations for code-switching, such as the the intent to express group solidarity, establish authority (Chang and Lin, 2014), lend credibility, or make up for lexical gaps.

It is not necessary for code-switching to occur only between two different languages like Spanish-English (Solorio and Liu, 2008), Mandarin-Taiwanese (Yu et al.,) and Turkish-German (Özlem Çetinoglu, 2016), but it can also happen between three languages, e.g. Bengali, English and Hindi (Barman et al., 2014), and in some extreme cases between six languages: English, French, German, Italian, Romansh and Swiss German (Volk and Clematide, 2014). Moreover, this phenomenon can occur between two different dialects of the same language as between Modern Standard Arabic (MSA) and Egyptian Dialect (Elfardy and Diab, 2012), or MSA and Moroccan Arabic (Samih and Maier,

Proceedings of the Second Workshop on Computational Approaches to Code Switching, pages 50–59,
Austin, TX, November 1, 2016.

2016a; Samih and Maier, 2016b). The current shared task is limited to two scenarios: a) code-switching between two distinct languages: Spanish-English, b) and two language varieties: MSA-Egyptian Dialect.

With the massive increase in code-switched writings in user-generated content, it has become imperative to develop tools and methods to handle and process this type of data. Identification of languages used in the sentence is the first step in doing any kind of text analysis. For example, most data found in social media produced by bilingual people is a mixture of two languages. In order to process or translate this data to some other language, the first step will be to detect text chunks and identify which language each chunk belongs to. The other categories like named entities, mixed, ambiguous and other are also important for further language processing.

2 Related Works

Code-switching has attracted considerable attention in theoretical linguistics and sociolinguistics over several decades. However, until recently there has not been much work on the computational processing of code-switched data. The first computational treatment of this linguistic phenomenon can be found in (Joshi, 1982). He introduces a grammar-based system for parsing and generating code-switched data. More recently, the detection of code-switching has gained traction, starting with the work of (Solorio and Liu, 2008), and culminating in the first shared task at the "First Workshop on Computational Approaches to Code Switching" (Solorio et al., 2014). Moreover, there have been efforts in creating and annotating code-switching resources (Özlem Çetinoglu, 2016; Elfardy and Diab, 2012; Maharjan et al., 2015; Lignos and Marcus, 2013). Maharjan et al. (2015) used a user-centric approach to collect code-switched tweets for Nepali-English and Spanish-English language pairs. They used two methods, namely a dictionary based approach and CRF GE and obtained an F1 score of 86% and 87% for Spanish-English and Nepali-English respectively at word level language identification task. Lignos and Marcus (2013) collected a large number of monolingual Spanish and English tweets and used ratio list method to tag each token with by its dominant language. Their system obtained an accuracy of 96.9% at word-level language identification task.

The task of detecting code-switching points is generally cast as a sequence labeling problem. Its difficulty depends largely on the language pair being processed.

Several projects have treated code-switching between MSA and Egyptian Arabic. For example, Elfardy et al. (2013) present a system for the detection of code-switching between MSA and Egyptian Arabic which selects a tag based on the sequence with a maximum marginal probability, considering 5-grams. A later version of the system is named AIDA2 (Al-Badrashiny et al., 2015) and it is a more complex hybrid system that incorporates different classifiers and components such as language models, a named entity recognizer, and a morphological analyzer. The classification strategy is built as a cascade voting system, whereby a conditional Random Field (CRF) classifier tags each word based on the decisions from four other underlying classifiers.

The participants of the "First Workshop on Computational Approaches to Code Switching" had applied a wide range of machine learning and sequence learning algorithms with some using additional online resources like English dictionary, Hindi-Nepali wiki, dbpedia, online dumps, LexNorm, etc. to tackle the problem of language detection in code-switched tweets on Nepali-English, Spanish-English, Mandarin-English and MSA Dialects (Solorio et al., 2014). For MSA-Dialects, two CRF-based systems, a system using language-independent extended Markov models, and a system using a CRF autoencoder have been presented; the latter proved to be the most successful.

The majority of the systems dealing with word-level language identification in code-switching rely on linguistic resources (such as named entity gazetteers and word lists) and linguistic information (such as POS tags and morphological analysis), and they use machine learning methods that have been typically used with sequence labeling problems, such as support vector machine (SVM), conditional random fields (CRF) and n-gram language models. Very few, however, have recently turned to recurrent neural networks (RNN) and word embedding with remarkable success. (Chang and Lin, 2014) used a RNN architecture and combined it

with pre-trained word2vector skip-gram word embeddings, a log bilinear model that allows words with similar contexts to have similar embeddings. The word2vec embeddings were trained on a large Twitter corpus of random samples without filtering by language, assuming that different languages tend to share different contexts, allowing embeddings to provide good separation between languages. They showed that their system outperforms the best SVM-based systems reported in the EMNLP'14 Code-Switching Workshop.

Vu and Schultz (2014) proposed to adapt the recurrent neural network language model to different code-switching behaviors and even use them to generate artificial code-switching text data. Adel et al. (2013) investigated the application of RNN language models and factored language models to the task of identifying code-switching in speech, and reported a significant improvement compared to the traditional n-gram language model.

Our work is similar to that of Chang and Lin (2014) in that we use RNNs and word embeddings. The difference is that we use long-short-term memory (LSTM) with the added advantage of the memory cells that efficiently capture long-distance dependencies. We also combine word-level with character-level representation to obtain morphology-like information on words.

3 Model

In this section, we will provide a brief description of LSTM, and introduce the different components of our code-switching detection model. The architecture of our system, shown in Figure 1, bears resemblance to the models introduced by Huang et al. (2015), Ma and Hovy (2016), and Collobert et al. (2011).

3.1 Long Short-term Memory

A recurrent neural network (RNN) belongs to a family of neural networks suited for modeling sequential data. Given an input sequence $x = (x_1, ..., x_n)$, an RNN computes the output vector y_t of each word x_t by iterating the following equations from $t = 1$ to n:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

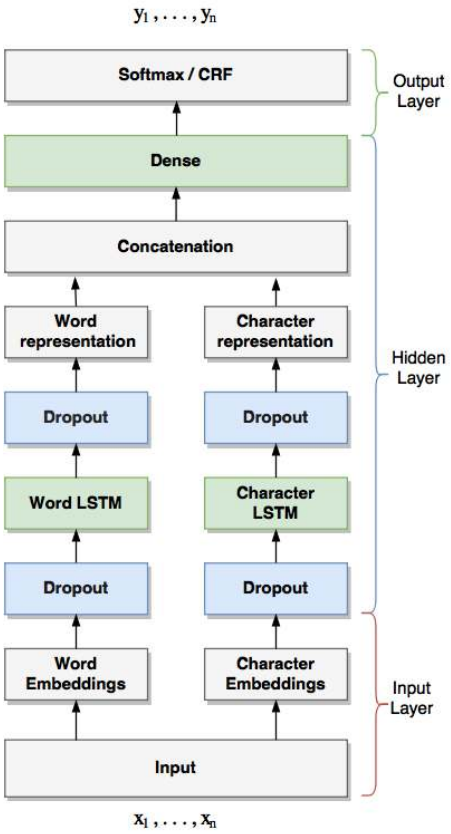


Figure 1: System Architecture.

where h_t is the hidden states vector, W denotes weight matrix, b denotes bias vector and f is the activation function of the hidden layer. Theoretically RNN can learn long distance dependencies, still in practice they fail due the vanishing/exploding gradient (Bengio et al., 1994). To solve this problem , Hochreiter and Schmidhuber (1997) introduced the long short-term memory RNN (LSTM). The idea consists in augmenting the RNN with memory cells to overcome difficulties with training and efficiently cope with long distance dependencies. The output of the LSTM hidden layer h_t given input x_t is computed via the following intermediate calculations: (Graves, 2013):

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$
$$h_t = o_t \tanh(c_t)$$

where σ is the logistic sigmoid function, and i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors. More interpretation about this architecture can be found in (Lipton et al., 2015). Figure 2 illustrates a single LSTM memory cell (Graves and Schmidhuber, 2005)

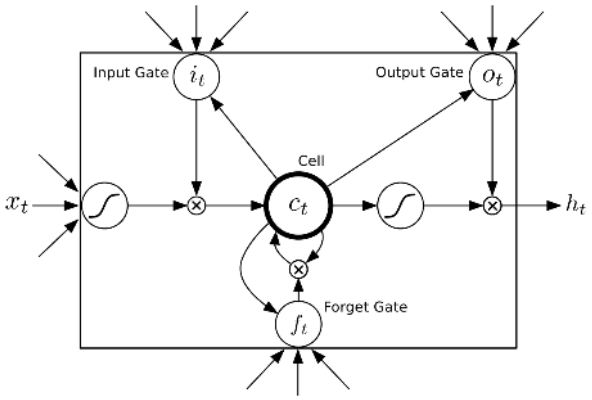


Figure 2: A Long Short-Term Memory Cell.

3.2 Word- and Character-level Embeddings

Character embeddings A very important element of the recent success of many NLP applications, is the use of character-level representations in deep neural networks. This has shown to be effective for numerous NLP tasks (Collobert et al., 2011; dos Santos et al., 2015) as it can capture word morphology and reduce out-of-vocabulary. This approach has also been especially useful for handling languages with rich morphology and large character sets (Kim et al., 2016). We also find this important for our code-switching detection model particularly for the Spanish-English data as the two languages have different orthographic sequences that are learned during the training phase.

Word pre-trained embeddings Another crucial component of our model is the use of pre-trained vectors. The basic assumption is that words from different languages (or language varieties) may appear in different contexts, so word embeddings learned from a large multilingual corpus, should provide an accurate separation between the languages at hand. Following Collobert et al. (2011), we use pre-trained word embeddings for Arabic, Spanish and English to initialize our look-up table. Words with no pre-trained embeddings are randomly initialized with uniformly sampled embeddings. To use these embeddings in our model, we simply replace the one hot encoding word representation with its corresponding 300-dimensional vector. For more details about the data we use to train our word embeddings for Arabic and Spanish-English, see Section 4.

3.3 Conditional Random Fields (CRF)

When using LSTM RNN for sequence classification, the resulting probability distribution of each step is supposed to be independent from each other. Still we assume that code-switching tags are highly related to each other. To exploit these kind of labeling constraints, we resort to Conditional Random Fields (CRF) (Lafferty et al., 2001). CRF, a sequence labeling algorithm, predicts labels for a whole sequence rather than for the parts in isolation as shown in Equation 1. Here, s_1 to s_m represent the labels of tokens x_1 to x_m respectively, where m is the number of tokens in a given sequence. After we have this probability value for every possible combination of labels, the actual sequence of labels for this set of tokens will be the one with the highest probability.

$$p(s_1...s_m|x_1...x_m) \tag{1}$$

$$p(\vec{s}|\vec{x};\vec{w}) = \frac{exp(\vec{w}.\vec{\Phi}(\vec{x},\vec{s}))}{\sum_{\vec{s'}\epsilon S^m} exp(\vec{w}.\vec{\Phi}(\vec{x},\vec{s'}))} \tag{2}$$

Equation 2 shows the formula for calculating the probability value from Equation 1. Here, S is the set of labels. In our case S =*{lang1, lang2, ambiguous, ne, mixed, other, fw, unk }*. $\vec{w}$ is the weight vector for weighting the feature vector $\vec{\Phi}$.

3.3.1 Feature Templates

The feature templates extract feature values based on the current position of the token, current token's label and previous token's label and the entire tweet. These functions normally output binary values (0 or 1). These feature functions can be represented mathematically as $\Phi(\vec{x}, j, s_{j-1}, s_j)$. We use the following feature templates.

Morphological Features: In order to capture the information contained in the morphology of tokens, we used features like, all upper case, title case, begins with punctuation, @, is digit, is alphanumeric,

contains apostrophe, ends with a vowel, consonant vowel ratio, has accented characters, prefixes and suffixes of the current token and of its previous or next token.

Character n-gram Features: character bigrams and trigrams.

Word Features: This feature uses token in its lowercase (hash-tag is removed from the token). Also, it tries to capture the context surrounding the token using the previous and next two tokens as features.

Shape Features: Collins (2002) defined a mapping from each character to its type. The type function blinds all characters but preserves the case information. The digits are replaced by # and all other punctuation characters are copied as they are. For example: "London" is transformed to "Xxxxxx", "PG-500' is transformed to "XX-###'. Another variation of the same function maps each character to its type but the repeated characters and not repeated in the mapping. So "London" is transformed to "Xx*". We use both of these variations in our system. These features are designed to capture the *named entity*.

Word Character Representations: The final representations from the char-word LSTM model before feeding to softmax layers for each token are used as features to the CRF.

3.4 LSTM-CRF for Code-switching Detection

Our neural network architecture consists of the following three layers:

- Input layer: comprises both character and word embeddings.
- Hidden layer: two LSTMs map both words and character representations to hidden sequences.
- Output layer: a Softmax or a CRF computes the probability distribution over all labels.

At the input layer a look-up table is randomly initialized mapping each word in the input to d-dimensional vectors for sequences of characters and sequences of words. At the hidden layer, the output from the character and word embeddings is used as the input to two LSTM layers to obtain fixed-dimensional representations for characters and words. At the output layer, a softmax or a CRF is applied over the hidden representation of the two LSTMs to obtain the probability distribution over all labels. Training is performed using stochastic gradient descent with momentum, optimizing the cross entropy objective function.

3.5 Optimization

Due to the relatively small size the training data set and development data set in both Arabic and Spanish-English, overfitting poses a considerable challenge for our code-switching detection system. To make sure that our model learns significant representations, we resort to dropout (Hinton et al., 2012) to mitigate overfitting. The basic idea of dropout consists in randomly omitting a certain percentage of the neurons in each hidden layer for each presentation of the samples during training. This encourages each neuron to depend less on other neurons to detect code-switching patterns. We apply dropout masks to both embedding layers before inputting to the two LSTMs and to their output vectors as shown in Fig. 1. In our experiments we find that dropout decreases overfitting and improves the overall performance of the system.

4 Dataset

The shared task organizers made available the tagged dataset for Spanish-English and Arabic (MSA-Egyptian) code-switched language pairs. The Spanish-English dataset consists of 8,733 tweets (139,539 tokens) as training set, 1,587 tweets (33,276 tokens) as development set and 10,716 tweets (121,446 tokens) as final test set. Similarly, the Arabic (MSA-Egyptian) dataset consists of 8,862 tweets (185,928 tokens) as training set, 1,117 tweets (20,688 tokens) as development set and 1,262 tweets (20,713 tokens) as final test set.

For Arabic we trained different word embeddings using word2vec (Mikolov et al., 2013) from a corpus of total size of 383,261,475 words, consisting of dialectal texts of Facebook posts (8,241,244), Twitter tweets (2,813,016), user comments on the news (95,241,480), and MSA texts of news articles of 276,965,735 words. Likewise, for Spanish-English, we combined English gigaword corpus (Graff et al., 2003) and Spanish gigaword corpus (Graff, 2006) before we trained different word embeddings on the final corpus.

Labels	CRF (feats)	CRF (emb)	CRF (feats+ emb)	word LSTM	char LSTM	char-word LSTM
ambiguous	0.00	0.02	0.00	0.00	0.00	0.00
fw	0.00	0.00	0.00	0.00	0.00	0.00
lang1	0.97	0.97	0.97	0.93	0.94	0.96
lang2	0.96	0.95	0.96	0.91	0.89	0.93
mixed	0.00	0.00	0.00	0.00	0.00	0.00
ne	0.52	0.51	0.57	0.34	0.13	0.32
other	1.00	1.00	1.00	0.85	1.00	1.00
unk	0.04	0.08	0.10	0.00	0.00	0.04
Accuracy	0.961	0.960	**0.963**	0.896	0.923	0.954

Table 1: F1 score results on Spanish-English development dataset. (feats = hand-crafted features, emb = representations for each token) The last three columns use softmax.

Labels	CRF (feats)	CRF (emb)	CRF (feats+ emb)	word LSTM	char LSTM	char-word LSTM
ambiguous	0.00	0.00	0.00	0.00	0.00	0.00
lang1	0.80	0.88	0.88	0.86	0.57	0.88
lang2	0.83	0.91	0.91	0.92	0.23	0.92
mixed	0.00	0.00	0.00	0.00	0.00	0.00
ne	0.83	0.84	0.86	0.84	0.66	0.84
other	0.97	0.97	0.97	0.92	0.97	0.97
Accuracy	0.829	0.894	0.896	0.896	0.530	**0.900**

Table 2: F1 score results on MSA-Egyptian development dataset. (feats = hand-crafted features, emb = representations for each token) The last three columns use softmax.

Data preprocessing: We transformed Arabic scripts to SafeBuckwalter (Roth et al., 2008), a character-to-character mapping that replaces Arabic UTF alphabet with Latin characters to reduce size and streamline processing. Also in order to reduce data sparsity, we converted all Persian numbers (e.g. ۲ ،۱) to Arabic numbers (e.g. 1, 2), Arabic punctuation (e.g. '،' and '؛') to Latin punctuation (e.g. ',' and ';'), removed kashida (elongation character) and vowel marks, and separated punctuation marks from words.

5 Experiments and Results

We explored different combinations of hand-crafted features (Section 3.3.1), word LSTM and char-word LSTM models with CRF and softmax classifier to identify the best system. Table 1 and 2 show the results for different settings for Spanish-English and MSA-Egyptian on the development dataset respectively. For the Spanish-English dataset, we find that combining the character and word representations learned with a char-word LSTM system with hand-crafted features and then using CRF as a sequence classifier gives the highest overall accuracy of 0.963. Also, we notice that the addition of character and word representations improves the F1-score for *named entity* and *unknown* tokens. For the MSA-Egyptian dataset, we find that a char-word LSTM model with softmax classifier is better than the CRF as this setting gives us the highest overall accuracy of 0.90. Moreover, the addition of character and word representations to hand-crafted features improves the F1 score for *named entity*. Based on these results, our final system for Spanish-English uses CRF with hand-crafted features and character and word representations learned with a char-word LSTM model and the MSA-Egyptian uses char-word LSTM model with softmax as classifier. We do not use any kind of hand-crafted features for the MSA-Egyptian dataset.

Our final system outperformed all other participants' systems for the MSA-Egyptian dialects in terms of tweet level and token level performance.

Scores	Es-En	MSA
Monolingual F1	0.92	0.890
Code-switched F1	0.88	0.500
Weighted F1	0.90	0.830

Table 3: Tweet level results the test dataset.

Label	Recall	Precision	F-score
ambiguous	0.000	0.000	0.000
fw	0.000	0.000	0.000
lang1	0.922	0.939	0.930
lang2	0.978	0.982	0.980
mixed	0.000	0.000	0.000
ne	0.639	0.484	0.551
other	0.992	0.998	0.995
unk	0.120	0.019	0.034
Accuracy			0.967

Table 4: Token level results on Spanish-English test dataset.

Label	Recall	Precision	F-score
ambiguous	0.000	0.000	0.000
fw	0.000	0.000	0.000
lang1	0.877	0.832	0.854
lang2	0.913	0.896	0.904
mixed	0.000	0.000	0.000
ne	0.729	0.829	0.777
other	0.938	0.975	0.957
unk	0.000	0.000	0.000
Accuracy			0.879

Table 5: Token level results on MSA-DA test dataset.

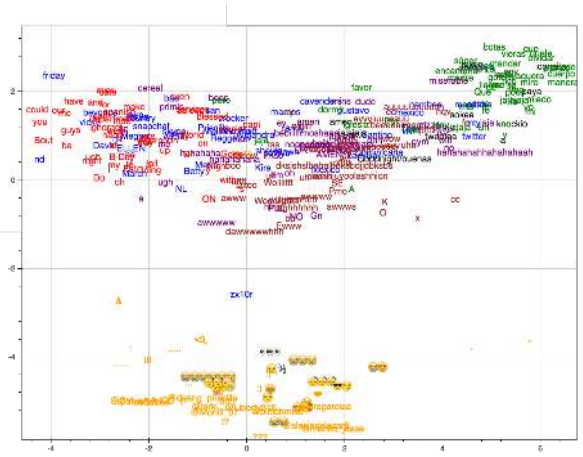
For the Spanish-English dataset, our system ranks second in terms of tweet level performance and third in terms of token level accuracy. Table 3, 4 and 5 show the final results for tweet and token level performance for the Spanish-English and MSA-Egyptian datasets. For the MSA dataset, the difference between our system and the second highest scoring system is 8% and 2.7% in terms of tweet level weighted F1 score and token level accuracy. Similarly for the Spanish-English dataset, the difference between our system and the highest scoring system is 1.3% and 0.6% in terms of tweet level weighted F1 score and token level accuracy. Our system consistently ranks first for language identification for the MSA-Egyptian dataset (5% and 4% above the second highest system for *lang1* and *lang2* respectively). For the Spanish-English dataset, our system ranks third (0.8% below the highest scoring system) and third (0.4% below the highest scoring system) for *lang1* and *lang2* respectively. However, our system has consistently shown weaker performance in identifying *ne*s. Nonetheless, the overall results show that our system outperforms other systems with relatively high margin for the MSA-Egyptian dataset and lags behind other systems with relatively low margin for the Spanish-English dataset.

6 Analysis

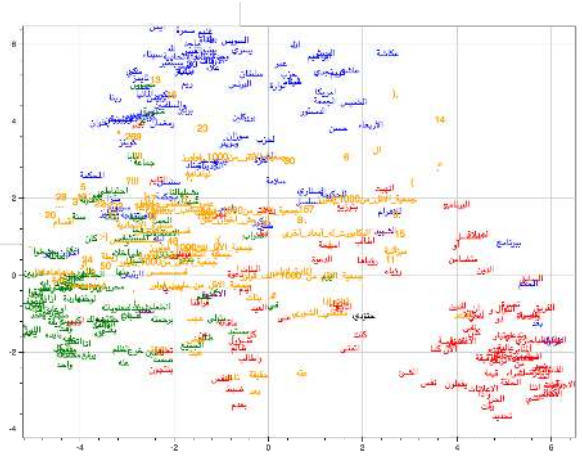
6.1 What is being captured in char-word representations?

In order to investigate what the char-word LSTM model is learning, we feed the tweets from the Spanish-English and MSA-Egyptian development datasets to the system and take the vectors formed by concatenation of character representation and word representation before feeding them into softmax layer. We then project them into 2D space by reducing the dimension of the vectors to 2 using Principle Component Analysis (PCA). We see, in Figure 3, that the trained neural network has learned to cluster the tokens according to their label type. Moreover, the position of tokens in 2D space also revels that *ambiguous* and *mixed* tokens are in between *lang1* and *lang2* clusters.

Figure 3(a) for Spanish-English shows that the char-word LSTM model has learned to separate the



(a) Spanish-English



(b) MSA-Egyptian Dialects

Figure 3: Projection of char-word LSTM representation into 2D using PCA. The tokens belonging to different categories are mapped as *ambiguous*: purple, *ne*: blue, *mixed*: black, *other*: orange, *lang1*: red, *lang2*: green

other tokens from rest of the tokens. These tokens are well separated and are situated at the bottom of the figure. Moreover, the *unknown* token are closer to the *other* tokens. The *ne*s as spread between *lang1* and *lang2* clusters. Named entities like *ELLEN, beyounce, friday, March, Betty* are closer to *lang1* cluster whereas, other named entities like *Mexico, Santino, gustavo, Norte* are closer to *lang2* cluster. Not only the named entities, the *mixed*, *ambiguous* tokens also exhibit the similar phenomena.

Similarly, Figure 3(b) for MSA-Egyptian generally shows successful separation of tokens, with *lang1* in red on the right, *lang2* in green on the left

Most likely	Score	Most unlikely	Score
$unk \Rightarrow unk$	1.789	$lang1 \Rightarrow mixed$	-0.172
$ne \Rightarrow ne$	1.224	$mixed \Rightarrow lang1$	-0.196
$fw \Rightarrow fw$	1.180	$amb \Rightarrow other$	-0.244
$lang1 \Rightarrow lang1$	1.153	$ne \Rightarrow mixed$	-0.246
$lang2 \Rightarrow lang2$	1.099	$mixed \Rightarrow other$	-0.254
$other \Rightarrow other$	0.827	$fw \Rightarrow lang1$	-0.282
$lang1 \Rightarrow ne$	0.316	$ne \Rightarrow lang2$	-0.334
$other \Rightarrow lang1$	0.222	$unk \Rightarrow ne$	-0.383
$lang2 \Rightarrow mixed$	0.216	$lang2 \Rightarrow lang1$	-0.980
$lang1 \Rightarrow other$	0.191	$lang1 \Rightarrow lang2$	-0.993

Table 6: Most likely and unlikely transitions learned by CRF model for the Spanish-English dataset.

and *ne* in blue on the top. The *other* token occupies the space between the clusters for *lang1*, *lang2* and *ne* with more inclination toward *lang1*. We also notice that *other* in Arabic contains a large amount of hashtags, due to their particular annotation scheme.

6.2 CRF Model

Table 6 shows the most likely and unlikely transitions learned by the CRF model for the Spanish-English dataset. It is interesting to see that the transition from *lang1* to *lang1* and from *lang2* to *lang2* are much likely than *lang1* to *lang2* or *lang2* to *lang1*. This suggests that people especially in Twitter do not normally switch from one language to another while tweeting. Even, if they switch, there are very few code-switch points in the tweets. However, people tweeting in Spanish have more tendency to use mixed tokens than people tweeting in English. We also dumped the top features for the task and found that *word.hasaps* is the top feature to identify token as English. Moreover, features like *word.startpunt, word.lower:number* are top features to identify tokens as *other*. The features like *char bigram, trigram, words, suffix and prefix* are the top features to distinguish between English and Spanish tokens.

6.3 Error Analysis for Arabic

When we conducted an error analysis on the output of the Arabic development set for our system, we found the following mistagging types:

- Punctuation marks, user names starting with '@' and emoticons are not tagged as *other*.
- Bad segmentation in the text affects the decision, e.g. عمروموسي EamormuwsaY "Amr Musa".
- Abbreviations: أ 'A' "Mr." and م 'm' "eng." are not consistently treated across the dataset.
- There are cases of true ambiguity, e.g. كريم 'kariym', which can be an adjective "generous" or a person's name "Kareem".
- Clitic attachment can obscure tokens, e.g. وطنطاوي waTanoTAwiy "and-Tantawy".
- Spelling errors can increase data sparsity, e.g. اسكنرية Asokanoriy~ap "Alexandria".

Based on this error analysis we developed a post-processor to handle deterministic annotation decision. The post-processor applies the tag *other* in the following cases:

- Non-alphabetic characters, e.g. punctuation marks and emoticons.
- Numbers already receiving *ne* tag, e.g. ٢٥ يناير "25 January".
- Strings with Latin script.
- Words starting with a @ character that usually represents user names.

6.4 Error Analysis for Spanish-English

From Table 1, it is clear that the most difficult categories are *ambiguous* and *mixed*. These are rare tokens and hence the system could not learn to distinguish them. During analyzing the mistakes on the development set, we found that the annotation of frequent tokens like *jaja, haha* with their spelling variations were inconsistent. Hence, even though the system was correctly predicting the labels, they were marked as incorrect. In addition, we also found that some *lang2* tokens like *que, amor, etc* were wrongly annotated as *lang1*.

In most cases, the system predicted either *lang1* or *lang2* for names of series, games, actor, day, apps (*friday, skype, sheyla, beyounce, walking dead, endless love, flappy bird, dollar tree*). We noticed that all these tokens were in lowercase. Similarly, the system mis-predicted all uppercase tokens as *ne*. For eg. *RT, DM, JK, GO, BOY* were annotated as *lang1*

but, the system predicted them as *ne*. Moreover, we found that the tokens like *lol, lmao, yolo, jk* were incorrectly annotated as *ne*.

The system predicted the interjections like *aww, uhh, muah, eeeahh, ughh* as either *lang1* or *lang2* but they were annotated as *unk*.

In order to improve the performance for *ne*, we tagged each token with Ark-Tweet NLP tagger (Owoputi et al., 2013). We then changed the label for the tokens tagged as proper nouns with confidence score greater than 0.98 to *ne*. This improved the F1-score for *ne* from 0.53 to 0.57.

7 Conclusion

In this paper we present our system for identifying and classifying code-switched data for Spanish-English and MSA-Egyptian. The system uses a neural network architecture that relies on word-level and character-level representations, and the output is fine-tuned (only in the Spanish-English data) with a CRF classifier for capturing sequence and contextual information. Our system is language independent in the sense that we have not used any language-specific knowledge or linguistic resources such as, POS taggers, morphological analyzers, gazetteers or word lists, and the main architecture is applied to both language sets. Our system considerably outperforms other systems participating in the shared task for Arabic, and is ranked second place for the Spanish-English at tweet-level performance.

Acknowledgments

We would like to thank Wolfgang Maier for enlightening discussions and the three anonymous reviewers for helpful comments and suggestions. This work was partially funded by Deutsche Forschungsgemeinschaft (DFG).

References

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 206–211, Sofia, Bulgaria.

Mohamed Al-Badrashiny, Heba Elfardy, and Mona Diab. 2015. AIDA2: A hybrid approach for token and sentence level dialect identification in arabic. In *Proc. CoNLL*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer 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*, pages 13–23, Doha, Qatar.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Joseph Chee Chang and Chu-Cheng Lin. 2014. Recurrent-neural-network for language detection on twitter code-switching corpus. *arXiv:1412.4314v2*.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 489–496. Association for Computational Linguistics.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *In Proceedings of the 11th International Conference on Natural Language Processing*, pages 169–178, Goa, India.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Heba Elfardy and Mona Diab. 2012. Simplified guidelines for the creation of large scale dialectal Arabic annotations. In *Proc. LREC*.

Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. 2013. Code switch point detection in Arabic. In *Proc. NLDB*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Dave Graff. 2006. Ldc2006t12: Spanish gigaword corpus. *Philadelphia: LDC*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012.

Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Aravind K Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Proc. COLING*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

Constantine Lignos and Mitch Marcus. 2013. Toward web-scale analysis of codeswitching. In *87th Annual Meeting of the Linguistic Society of America*.

Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. 2015. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August. Association for Computational Linguistics.

Suraj Maharjan, Elizabeth Blair, Steven Bethard, and Thamar Solorio. 2015. Developing language-tagged corpora for code-switching tweets. In *Proc. LAW IX at NAACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Lesley Milroy and Pieter Muysken. 1995. *One Speaker, Two Languages: Cross-Disciplinary Perspectives on Code-Switching*. Cambridge Univ. Press.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Evangelos E. Papalexakis, Dong Nguyen, and A. Seza Doğruöz. 2014. Predicting code-switching in multilingual communication for immigrant communities. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 42–50, Doha, Qatar.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. *Proceedings of the Conference of American Association for Computational Linguistics (ACL08)*.

Younes Samih and Wolfgang Maier. 2016a. An Arabic-Moroccan Darija code-switched corpus. In *Proc. LREC*.

Younes Samih and Wolfgang Maier. 2016b. Detecting code-switching in moroccan arabic. In *Proceedings of SocialNLP @ IJCAI-2016*, New York.

Thamar Solorio and Yang Liu. 2008. Learning to predict code-switching points. In *Proc. EMNLP*, pages 973–981.

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 *Proc. CS Workshop at EMNLP*.

Martin Volk and Simon Clematide. 2014. Detecting code-switching in a multilingual alpine heritage corpus. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 24–33, Doha, Qatar.

Ngoc Thang Vu and Tanja Schultz. 2014. Exploration of the impact of maximum entropy in recurrent neural network language models for code-switching speech. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 34–41, Doha, Qatar.

Ching-Feng Yeh, Aaron Heidel, Hong-Yi Lee, and Lin-Shan Lee. 2013. Recognition of highly imbalanced code-mixed bilingual speech with frame-level language detection based on blurred posteriorgram. *IEEE*, 2013.

Liang-Chih Yu, Wei-Cheng He, Wei-Nan Chien, and Yuen-Hsien Tseng. Identification of code-switched sentences and words using language modeling approaches. *Mathematical Problems in Engineering*, 978.

Özlem Çetinoglu. 2016. A turkish-german code-switching corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia.