Combination of Recurrent Neural Networks and Factored Language Models for Code-Switching Language Modeling

Heike Adel **Ngoc Thang Vu** **Tanja Schultz**
Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT)
heike.adel@student.kit.edu thang.vu@kit.edu tanja.schultz@kit.edu

Abstract

In this paper, we investigate the application of recurrent neural network language models (RNNLM) and factored language models (FLM) to the task of language modeling for Code-Switching speech. We present a way to integrate part-of-speech tags (POS) and language information (LID) into these models which leads to significant improvements in terms of perplexity. Furthermore, a comparison between RNNLMs and FLMs and a detailed analysis of perplexities on the different backoff levels are performed. Finally, we show that recurrent neural networks and factored language models can be combined using linear interpolation to achieve the best performance. The final combined language model provides 37.8% relative improvement in terms of perplexity on the SEAME development set and a relative improvement of 32.7% on the evaluation set compared to the traditional n-gram language model.

Index Terms: multilingual speech processing, code switching, language modeling, recurrent neural networks, factored language models

1 Introduction

Code-Switching (CS) speech is defined as speech that contains more than one language ('code'). It is a common phenomenon in multilingual communities (Auer, 1999a). For the automated processing of spoken communication in these scenarios, a speech recognition system must be able to handle code switches. However, the components of speech recognition systems are usually trained on monolingual data. Furthermore, there is a lack of bilingual training data. While there have been promising research results in the area of acoustic modeling, only few approaches so far address Code-Switching in the language model. Recently, it has been shown that recurrent neural network language models (RNNLMs) can improve perplexity and error rates in speech recognition systems in comparison to traditional n-gram approaches (Mikolov et al., 2010; Mikolov et al., 2011). One reason for that is their ability to handle longer contexts. Furthermore, the integration of additional features as input is rather straightforward due to their structure. On the other hand, factored language models (FLMs) have been used successfully for languages with rich morphology due to their ability to process syntactical features, such as word stems or part-of-speech tags (Bilmes and Kirchhoff, 2003; El-Desoky et al., 2010).
The main contribution of this paper is the application of RNNLMs and FLMs to the challenging task of Code-Switching. Furthermore, the two different models are combined using linear interpolation. In addition, a comparison between them is provided including a detailed analysis to explain their results.

2 Related Work

For this work, three different topics are investigated and combined: linguistic investigation of Code-Switching, recurrent neural network language modeling and factored language models.
In (Muysken, 2000; Poplack, 1978; Bokamba, 1989), it is observed that code switches occur at positions in an utterance where they do not violate the syntactical rules of the involved languages. On the one hand, Code-Switching can be regarded as a speaker dependent phenomenon (Auer, 1999b; Vu, Adel et al., 2013). On the other hand, particular Code-Switching patterns are shared across speakers (Poplack, 1980). It can be observed that part-of-speech tags may predict Code-Switching points more reliable than words themselves. The

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 206–211,
Sofia, Bulgaria, August 4-9 2013.

authors of (Solorio et al., 2008a) predict Code-Switching points using several linguistic features, such as word form, language ID, part-of-speech tags or the position of the word relative to the phrase (BIO). The best result is obtained by combining those features. In (Chan et.al., 2006), four different kinds of n-gram language models are compared to predict Code-Switching. It is discovered that clustering all foreign words into their part-of-speech classes leads to the best performance.

In the last years, neural networks have been used for a variety of tasks, including language modeling (Mikolov et al., 2010). Recurrent neural networks are able to handle long-term contexts since the input vector does not only contain the current word but also the previous hidden layer. It is shown that these networks outperform traditional language models, such as n-grams which only contain very limited histories. In (Mikolov et al., 2011), the network is extended by factorizing the output layer into classes to accelerate the training and testing processes. The input layer can be augmented to model features, such as part-of-speech tags (Shi et al., 2011; Adel, Vu et al., 2013). In (Adel, Vu et al., 2013), recurrent neural networks are applied to Code-Switching speech. It is shown that the integration of POS tags into the neural network, which predicts the next language as well as the next word, leads to significant perplexity reductions.

A factored language model refers to a word as a vector of features, such as the word itself, morphological classes, POS tags or word stems. Hence, it provides another possibility to integrate syntactical features into the language modeling process. In (Bilmes and Kirchhoff, 2003), it is shown that factored language models are able to outperform standard n-gram techniques in terms of perplexity. In the same paper, generalized parallel backoff is introduced. This technique can be used to generalize traditional backoff methods and to improve the performance of factored language models. Due to the integration of various features, it is possible to handle rich morphology in languages like Arabic or Turkish (Duh and Kirchhoff, 2004; El-Desoky et al., 2010).

3 Code-Switching Language Modeling

3.1 Motivation

Since there is a lack of Code-Switching data, language modeling is a challenging task. Traditional n-gram approaches may not provide reliable estimates. Hence, more general features than words should be integrated into the language models. Therefore, we apply recurrent neural networks and factored language models. As features, we use part-of-speech tags and language identifiers.

3.2 Using Recurrent Neural Networks As Language Model

This section describes the structure of the recurrent neural network (RNNLM) that we use as Code-Switching language model. It has been proposed in (Adel, Vu et al., 2013) and is illustrated in figure 1.

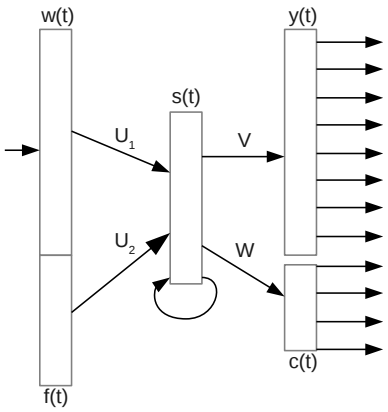

Figure 1: RNNLM for Code-Switching
(based upon a figure in (Mikolov et al., 2011))

Vector $w(t)$, which represents the current word using 1-of-N coding, forms the input of the recurrent neural network. Thus, its dimension equals the size of the vocabulary. Vector $s(t)$ contains the state of the network and is called 'hidden layer'. The network is trained using back-propagation through time (BPTT), an extension of the back-propagation algorithm for recurrent neural networks. With BPTT, the error is propagated through recurrent connections back in time for a specific number of time steps t. Hence, the network is able to remember information for several time steps. The matrices U_1, U_2, V, and W contain the weights for the connections between the layers. These weights are learned during the training phase. Moreover, the output layer is factorized

into classes which provide language information. In this work, four classes are used: English, Mandarin, other languages and particles. Vector $c(t)$ contains the probabilities for each class and vector $y(t)$ provides the probabilities for each word given its class. Hence, the probability $P(w_i|history)$ is computed as shown in equation 1.

$$P(w_i|history) = P(c_i|s(t))P(w_i|c_i, s(t)) \quad (1)$$

It is intended to not only predict the next word but also the next language. Hence according to equation 1, the probability of the next language is computed first and then the probability of each word given the language. Furthermore, a vector $f(t)$ is added to the input layer. It provides features (in this work part-of-speech tags) corresponding to the current word. Thus, not only the current word is activated but also its features. Since the POS tags are integrated into the input layer, they are also propagated into the hidden layer and backpropagated into its history $s(t)$. Hence, not only the previous feature is stored in the history but also features from several time steps in the past.

3.3 Using Factored Language Models

Factored language models (FLM) are another approach to integrate syntactical features, such as part-of-speech tags or language identifiers into the language modeling process. Each word is regarded as a sequence of features which are used for the computation of the n-gram probabilities. If a particular sequence of features has not been detected in the training data, backoff techniques will be applied. For our task of Code-Switching, we develop two different models: One model with only part-of-speech tags as features and one model including also language information tags. Unfortunately, the number of possible parameters is rather high: Different feature combinations from different time steps can be used to predict the next word (conditioning factors), different backoff paths and different smoothing methods may be applied. To detect useful parameters, the genetic algorithm described in (Duh and Kirchhoff, 2004) is used. It is an evolution-inspired technique that encodes the parameters of an FLM as binary strings (genes). First, an initializing set of genes is generated. Then, a loop follows that evaluates the fitness of the genes and mutates them until their average fitness is not improved any more. As fitness value, the inverse perplexity of the FLM corresponding to the gene on the development set is used. Hence, parameter solutions with lower perplexities are preferred in the selection of the genes for the following iteration. In (Duh and Kirchhoff, 2004), it is shown that this genetic method outperforms both knowledge-based and randomized choices. For the case of part-of-speech tags as features, the method results in three conditioning factors: the previous word W_{t-1} and the two previous POS tags P_{t-1} and P_{t-2}. The backoff graph obtained by the algorithm is illustrated in figure 2. According to the result of the genetic algorithm, different smoothing methods are used at different backoff levels: For the backoff from three factors to two factors, Kneser-Ney discounting is applied. If the probabilities for the factor combination $W_{t-1}P_{t-2}$ could not be estimated reliably, absolute discounting is used. In all other cases, Witten-Bell discounting is applied. An overview of the different smoothing methods can be found in (Rosenfeld, 2000).

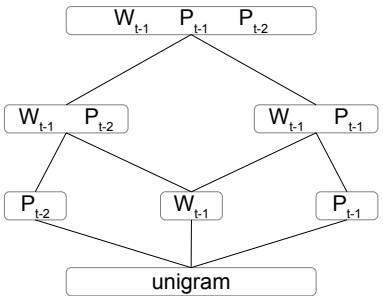

Figure 2: Backoff graph of the FLM

4 Experiments and Results

4.1 Data Corpus

SEAME (South East Asia Mandarin-English) is a conversational Mandarin-English Code-Switching speech corpus recorded from Singaporean and Malaysian speakers (D.C. Lyu et al., 2011). It was used for the research project 'Code-Switch' jointly performed by Nanyang Technological University (NTU) and Karlsruhe Institute of Technology (KIT). The recordings consist of spontanously spoken interviews and conversations of about 63 hours of audio data. For this task, we deleted all hesitations and divided the transcribed words into four categories: English words, Mandarin words, particles (Singaporean and Malaysian discourse particles) and others (other languages). These categories are used as language information in the language models. The average number of Code-Switching points between Mandarin and English

is 2.6 per utterance and the duration of monolingual segments is quite short: The average duration of English and Mandarin segments is only 0.67 seconds and 0.81 seconds respectively. In total, the corpus contains 9,210 unique English and 7,471 unique Mandarin vocabulary words. We divided the corpus into three disjoint sets (training, development and test set) and assigned the data based on several criteria (gender, speaking style, ratio of Singaporean and Malaysian speakers, ratio of the four categories, and the duration in each set). Table 1 lists the statistics of the corpus in these sets.

	Train set	Dev set	Eval set
# Speakers	139	8	8
Duration(hrs)	59.2	2.1	1.5
# Utterances	48,040	1,943	1,018
# Token	525,168	23,776	11,294

Table 1: Statistics of the SEAME corpus

4.2 POS Tagger for Code-Switching Speech

To be able to assign part-of-speech tags to our bilingual text corpus, we apply the POS tagger described in (Schultz et al., 2010) and (Adel, Vu et al., 2013). It consists of two different monolingual (Stanford log-linear) taggers (Toutanova et al., 2003; Toutanova et al., 2000) and a combination of their results. While (Solorio et al., 2008b) passes the whole Code-Switching text to both monolingual taggers and combines their results using different heuristics, in this work, the text is splitted into different languages first. The tagging process is illustrated in figure 3.
Mandarin is determined as matrix language (the main language of an utterance) and English as embedded language. If three or more words of the embedded language are detected, they are passed to the English tagger. The rest of the text is passed to the Mandarin tagger, even if it contains foreign words. The idea is to provide the tagger as much context as possible. Since most English words in the Mandarin segments are falsely tagged as nouns by the Mandarin tagger, a postprocessing step is applied. It passes all foreign words of the Mandarin segments to the English tagger in order to replace the wrong tags with the correct ones.

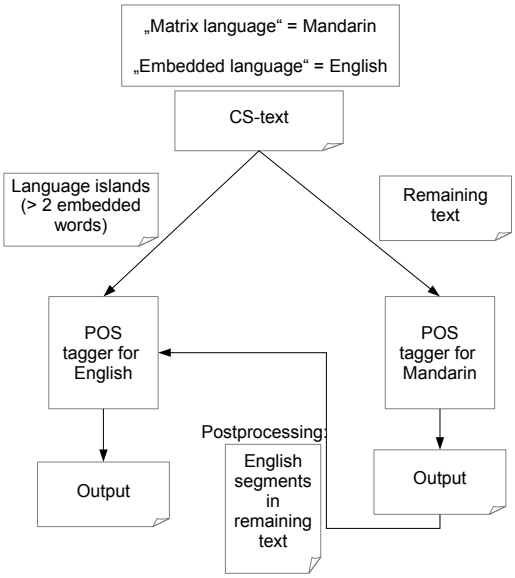

Figure 3: Tagging of Code-Switching speech

4.3 Evaluation

For evaluation, we compute the perplexity of each language model on the SEAME development and evaluation set und perform an analysis of the different back-off levels to understand in detail the behavior of each language model. A traditional 3-gram LM trained with the SEAME transcriptions serves as baseline.

4.3.1 LM Performance

The language models are evaluated in terms of perplexity. Table 2 presents the results on the development and test set.

Model	dev set	test set
Baseline 3-gram	285.87	285.25
FLM (pos)	**263.57**	**271.57**
FLM (pos + lid)	263.84	276.99
RNNLM (pos)	233.50	268.05
RNNLM (pos + lid)	**219.85**	**239.21**

Table 2: Perplexity results

It can be noticed that both the RNNLM and the FLM model outperform the traditional 3-gram model. Hence, adding syntactical features improves the word prediction. For the FLM, it leads to no improvement to add the language identifier as feature. In contrast, clustering the words into their languages on the output layer of the RNNLM leads to lower perplexities.

4.3.2 Backoff Level Analysis

To understand the different results of the RNNLM and the FLM, an analysis similar to the one described in (Oparin et al., 2012) is performed. For each word, the backoff-level of the n-gram model is observed. Then, a level-dependent perplexity is computed for each model as shown in equation 2.

$$PPL_k = 10^{-\frac{1}{N_k}\sum_{w_k} log_{10} P(w_k|h_k)} \quad (2)$$

In the equation, k denotes the backoff-level, N_k the number of words on this level, w_k the current word and h_k its history. Table 3 shows how often each backoff-level is used and presents the level-dependent perplexities of each model on the development set.

	1-gram	2-gram	3-gram
# occurences	6894	11628	6226
Baseline 3-gram	5,786.24	165.82	28.28
FLM (pos)	4,950.31	**147.70**	30.99
RNNLM	**3,231.02**	151.67	**21.24**

Table 3: Backoff-level-dependent PPLs

In case of backoff to the 2-gram, the FLM provides the best perplexity, while for the 3-gram and backoff to the 1-gram, the RNNLM performs best. This may be correlated with the better over-all perplexity of the RNNLM in comparison to the FLM. Nevertheless, the backoff to the 2-gram is used about twice as often as the backoff to the 1-gram or the 3-gram.

4.4 LM Interpolation

The different results of RNNLM and FLM show that they provide different estimates of the next word. Thus, a combination of them may reduce the perplexities of table 2. Hence, we apply linear interpolation to the probabilities of each two models as shown in equation 3.

$$P(w|h) = \lambda \cdot P_{M1}(w|h) + (1-\lambda) \cdot P_{M2}(w|h) \quad (3)$$

The equation shows the computation of the pobability for word w given its history h. P_{M1} denotes the probability provided by the first model and P_{M2} the probability from the second model. Table 4 shows the results of this experiment. The weights are optimized on the development set.
The interpolation of RNNLM and FLM leads to the best results. This may be caused by the superior backoff-level-dependent PPLs in comparison to the 3-gram model. While the RNNLM performs better for the 3-gram and for the backoff to the 1-gram, the FLM performs the best in case of backoff to the 2-gram which is used more often than the other levels (table 3).

Model	weight	PPL on dev	PPL on eval
FLM + 3-gram	0.7, 0.3	211.13	227.57
RNNLM + 3-gram	0.8, 0.2	206.49	227.08
RNNLM + FLM	0.6, 0.4	**177.79**	**192.08**

Table 4: Perplexities after interpolation

5 Conclusions

In this paper, we presented two different methods for language modeling of Code-Switching speech: Recurrent neural networks and factored language models. We integrated part-of-speech tags and language information to improve the performance of the language models. In addition, we analyzed their behavior on the different backoff levels. While the FLM performed better in case of backoff to the 2-gram, the RNNLM led to a better over-all performance. Finally, the models were combined using linear interpolation. The combined language model provided 37.8% relative improvement in terms of perplexity on the SEAME development set and a relative improvement of 32.7% on the evaluation set compared to the traditional n-gram LM.

References

H. Adel, N.T. Vu, F. Kraus, T. Schlippe, and T. Schultz. 2013 *Recurrent Neural Network Language Modeling for Code Switching Conversational Speech* In: Proceedings of ICASSP 2013.

P. Auer 1999 *Code-Switching in Conversation* , Routledge.

P. Auer 1999 *From codeswitching via language mixing to fused lects toward a dynamic typology of bilingual speech* In: International Journal of Bilingualism, vol. 3, no. 4, pp. 309-332.

J.A. Bilmes and K. Kirchhoff. 2003 *Factored Language Models and Generalized Parallel Backoff* In: Proceedings of NAACL, 2003.

E.G. Bokamba 1989 *Are there syntactic constraints on code-mixing?* In: World Englishes, vol. 8, no. 3, pp. 277-292.

J.Y.C. Chan, PC Ching, T. Lee, and H. Cao 2006 *Automatic speech recognition of Cantonese-English*

code-mixing utterances In: Proceeding of Interspeech 2006.

K. Duh and K. Kirchhoff. 2004. *Automatic Learning of Language Model Structure*, pg 148. In: Proceedings of the 20th international conference on Computational Linguistics.

A. El-Desoky, R. Schlüter, H.Ney 2010 *A Hybrid Morphologically Decomposed Factored Language Models for Arabic LVCSR* In: NAACL 2010.

D.C. Lyu, T.P. Tan, E.S. Cheng, H. Li 2011 *An Analysis of Mandarin-English Code-Switching Speech Corpus: SEAME* In: Proceedings of Interspeech 2011.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993 *Building a large annotated corpus of english: The penn treebank* In: Computational Linguistics, vol. 19, no. 2, pp. 313330.

T. Mikolov, M. Karafiat, L. Burget, J. Jernocky and S. Khudanpur. 2010 *Recurrent Neural Network based Language Model* In: Proceedings of Interspeech 2010.

T. Mikolov, S. Kombrink, L. Burget, J. Jernocky and S. Khudanpur. 2011 *Extensions of Recurrent Neural Network Language Model* In: Proceedings of ICASSP 2011.

P. Muysken 2000 *Bilingual speech: A typology of code-mixing* In: Cambridge University Press, vol. 11.

I. Oparin, M. Sundermeyer, H. Ney, J.-L. Gauvain 2012 *Performance analysis of Neural Networks in combination with n-gram language models* In: ICASSP, 2012.

S. Poplack 1978 *Syntactic structure and social function of code-switching* , Centro de Estudios Puertorriquenos, City University of New York.

S. Poplack 1980 *Sometimes ill start a sentence in spanish y termino en espanol: toward a typology of code-switching* In: Linguistics, vol. 18, no. 7-8, pp. 581-618.

R. Rosenfeld 2000 *Two decades of statistical language modeling: Where do we go from here?* In: Proceedings of the IEEE 88.8 (2000): 1270-1278.

T. Schultz, P. Fung, and C. Burgmer, 2010 Detecting code-switch events based on textual features.

Y. Shi, P. Wiggers, M. Jonker 2011 *Towards Recurrent Neural Network Language Model with Linguistics and Contextual Features* In: Proceedings of Interspeech 2011.

T. Solorio, Y. Liu 2008 *Part-of-speech tagging for English-Spanish code-switched text* In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2008.

T. Solorio, Y. Liu 2008 *Learning to predict code-switching points* In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2008.

K. Toutanova, C.D. Manning 2000 *Enriching the knowledge sources used in a maximum entropy part-of-speech tagger* In: Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics, vol. 13.

K. Toutanova, D. Klein, C.D. Manning, and Y. Singer 2003 *Feature-rich part-of-speech tagging with a cyclic dependency network* In: Proceedings of NAACL 2003.

N.T. Vu, D.C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.S. Chng, T. Schultz, H. Li 2012 *A First Speech Recognition System For Mandarin-English Code-Switch Conversational Speech* In: Proceedings of Interspeech 2012.

N.T. Vu, H. Adel, T. Schultz 2013 *An Investigation of Code-Switching Attitude Dependent Language Modeling* In: In Statistical Language and Speech Processing, First International Conference, 2013.

N. Xue, F. Xia, F.D. Chiou, and M. Palmer 2005 *The penn chinese treebank: Phrase structure annotation of a large corpusk* In: Natural Language Engineering, vol. 11, no. 2, pp. 207.