Combining Recurrent Neural Networks and Factored Language Models During Decoding of Code-Switching Speech

Conference Paper · September 2014

DOI: 10.13140/2.1.1292.6089

Heike Adel
Universität Stuttgart
46 PUBLICATIONS **609** CITATIONS

Dominic Telaar
Karlsruhe Institute of Technology
9 PUBLICATIONS **409** CITATIONS

Thang Vu
Universität Stuttgart
124 PUBLICATIONS **1,820** CITATIONS

Katrin Kirchhoff
University of Washington Seattle
99 PUBLICATIONS **2,672** CITATIONS

Combining Recurrent Neural Networks and Factored Language Models During Decoding of Code-Switching Speech

Heike Adel[1,2], *Dominic Telaar*[1], *Ngoc Thang Vu*[1], *Katrin Kirchhoff*[2], *Tanja Schultz*[1]

[1]Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany
[2]University of Washington, Department of Electrical Engineering, USA

heike.adel@student.kit.edu

Abstract

In this paper, we present our latest investigations of language modeling for Code-Switching. Since there is only little text material for Code-Switching speech available, we integrate syntactic and semantic features into the language modeling process. In particular, we use part-of-speech tags, language identifiers, Brown word clusters and clusters of open class words. We develop factored language models and convert recurrent neural network language models into backoff language models for an efficient usage during decoding. A detailed error analysis reveals the strengths and weaknesses of the different language models. When we interpolate the models linearly, we reduce the perplexity by 15.6% relative on the SEAME evaluation set. This is even slightly better than the result of the unconverted recurrent neural network. We also combine the language models during decoding and obtain a mixed error rate reduction of 4.4% relative on the SEAME evaluation set.

Index Terms: Code-Switching, language modeling, factored language models, recurrent neural network language models

1. Introduction

Code-Switching (CS) may be defined as speech with more than one language. Speakers change languages within or between utterances. For the task of language modeling of CS speech, multilingual models are necessary to capture probabilities for both languages. The training data for these models should also contain more than one language since a language model (LM) built on monolingual texts of both languages cannot capture language changes within utterances. In this work, we use the transcriptions of the CS speech database SEAME to train our models. It is described in detail in Section 5.1. To overcome data sparseness problems, we integrate more general features than words into the language modeling process. In particular, we develop factored language models (FLMs) and a recurrent neural network language model (RNNLM). Both language model types have been able to outperform traditional n-gram approaches in the past [1, 2, 3, 4]. However, recurrent neural network language models cannot be used efficiently during decoding because of their high computational demands. This is why we convert our recurrent neural network into a backoff language model. This paper has two main contributions: 1) Both language model types are combined not only for perplexity computations but also during decoding. 2) Detailed error analyses are provided to show why the combination outperforms the single language models.

2. Related work

This section differentiates previous studies about LM for CS speech from this paper. Furthermore, it presents research on recurrent neural network language models (RNNLMs) and factored language models (FLMs).

2.1. Language models for Code-Switching

Several studies showed that features like part-of-speech (POS) tags are useful to predict Code-Switching [3, 5, 6, 7]. In [3], we built an RNNLM for CS speech. It used POS tags as input features and clustered the output vector into language classes. Although this model led to large perplexity improvements, we did not achieve significant improvements in rescoring experiments. We assume that the reason lies in errors made by the n-gram during decoding. Therefore, we now convert the network into a backoff language model to integrate it directly into the decoding process. This reduces the error rate significantly. In [4], we combined our RNNLM with an FLM trained also with POS tags and language identifiers. There are two main differences of our previous work to this paper: 1) The FLM of our previous study contained less features and a feature set which led to higher perplexity values. 2) The combination of the FLM with the RNNLM was restricted to perplexity computations. We did not use the models in the decoding process. In [8], we described different features for FLMs and their performance on the CS database SEAME in detail.

2.2. Recurrent neural network language modeling

In the last years, it has been shown that RNNLMs can outperform n-gram models in the task of language modeling [9]. The authors of [1] factorized the output layer of the RNNLM into classes to accelerate training and testing processes. In [3, 10], features were integrated into the input layer to add linguistic information to the LM process. The authors of [11] explore the information learned by an RNNLM. They discover that the network is able to capture syntactic and semantic regularities. For example, the relationship of the vectors for the words "man" and "king" is the same as the relationship of the vectors for "woman" and "queen". In this paper, these word representations are used to derive features for our FLM.
In the literature, only few studies describe conversion approaches for neural networks. In [12], a method for converting feed forward neural networks is proposed. Starting from 2-grams, LMs are developed iteratively for each increasing order n. In each iteration, probabilities for every possible n-gram are obtained from an n-gram network or an n-gram backoff model. Then, backoff weights are calculated and the model is pruned.

The n-grams are restricted to those word combinations whose histories have been left over after pruning the $n-1$-grams. In [13], the authors describe how an n-gram model can be built on text generated by an RNNLM. The resulting n-gram model is close to the RNNLM in terms of Kullback-Leibler divergence.

2.3. Factored language modeling

FLMs integrate various features into the language modeling process. This is why they can be used to handle languages with rich morphology like Arabic [2, 14].

3. RNNLM for Code-Switching

This section describes our RNNLM for CS speech. It is converted to a backoff language model in order to use its information directly during decoding.

3.1. Structure of the RNNLM for Code-Switching

The structure of our CS RNNLM is illustrated in Figure 1 and has been proposed in [3].

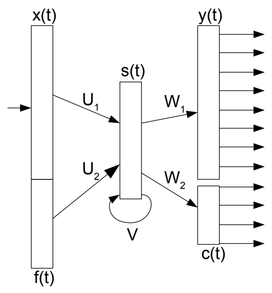

Figure 1: RNNLM for Code-Switching
(based upon a figure in [1])

The network consists of three layers: an input layer, a hidden layer and an output layer. Since the values of the hidden layer do not only depend on the input layer but also on the hidden layer of the previous iteration, the network is called "recurrent". The input layer consists of two vectors. Vector x contains one entry for each word of the vocabulary. Vector f provides additional features to the network. In our case, it consists of one entry for each POS tag. Hence, the values of the hidden layer are computed based on both the current word and the current POS tag. The output vector y provides the probabilities for each word of the vocabulary for being the next word. Its softmax-activation function ensures that its values form a probability distribution. As mentioned before, [1] has factorized the output vector into frequency based classes for acceleration purposes. We have proposed to use language classes instead to model the CS phenomenon [3]. Therefore, the network first computes the probability of the next language class c and, afterwards, based on c the probability of the next word.

3.2. Conversion of the RNNLM into n-grams

For the conversion of our RNNLM into an n-gram model, we used the approach by [13] and adapted the method by [12] to RNNLMs. Finally, the converted LMs are interpolated linearly. In the first approach (text generation), we generated a large amount of text with our RNNLM. We investigated different text sizes and found that generating about 235M words led to the lowest perplexity results on the SEAME development set [15]. Then, we built a 3-gram LM on this text and interpolated it with a 3-gram model built on the SEAME training data.
With the second approach (iterative conversion), we developed a 2-gram LM. The authors of [12] built a 2-gram feed forward neural network and consulted it for every possible bigram. We found that we could not adopt this process to RNNLMs [15]. Due to the recurrence of the hidden layer, it cannot be determined which output value corresponds to a 2-gram probability. Restricting the number of timesteps for unfolding the network during backpropagation or resetting the hidden layer after each bigram in the training text did not lead to good results, either. Therefore, we developed an innovative method to extract 2-gram probabilities from our RNNLM: For each word in the training text, we extracted the probabilities for every possible next word and stored them as 2-gram probabilities. Whenever a word occured more than once, we averaged the probabilities. For words, which occured in the SEAME vocabulary but not in the RNNLM vocabulary, we used a traditional 2-gram model to obtain their probabilities. We regarded this as backoff from the RNNLM to the 2-gram model. Therefore, we adjusted the probabilities of the RNNLM and the 2-gram by multiplying backoff weights with the 2-gram probabilities. This is another difference to the original paper of [12] in which the authors multiplied the probabilities of the network with a history dependent constant. Finally, we pruned the model to a size comparable to a traditional 2-gram model. This improved the perplexity slightly. In order to create a 3-gram model, we performed the same steps for 3-gram probabilities. However, the training text did not include enough trigram histories to obtain many probabilities from the RNNLM. A computation of every possible trigram based on every single word of the text (instead of every bigram) would be too time-consuming. Therefore, we found that interpolating the converted 2-gram model with a 3-gram model built on the SEAME training data led to better results than a converted 3-gram model [15]. Hence, we used the converted 2-gram model for our perplexity and decoding experiments.

4. FLM for Code-Switching

This section presents the basics of FLMs and the features we used to predict CS speech.

4.1. Factored language modeling

FLMs provide the possiblity to integrate a variety of features into the LM process [2]. Each word is regarded as a collection of features (factors). If a particular feature context has not been detected in the training data, backoff techniques will be applied. For this, the traditional backoff has been generalized. At each backoff step, every factor of the current context can be dropped in order to apply a model with less features. Furthermore, various backoff paths can be combined. Due to the integration of features and the generalized backoff, FLMs often outperform traditional n-gram models, especially for morphologically rich languages or in case of little training data. This is why we investigated them in the context of CS speech.

4.2. Features for Code-Switching speech

The selection of appropriate features is an important design choice of FLMs. In [8], we have investigated various features and feature combinations for CS. In this section, we describe the feature combination which led to the lowest perplexity results in combination with our converted RNNLM. It consists of the factors words, POS tags, Brown word clusters and bilingual clusters of open class words.

4.2.1. *Part-of-speech tags*

In many previous studies [3, 5, 7], POS tags were useful features for the prediction of CS. Therefore, we tagged the mixed-speech texts with the tagging process described in [3, 7]. It splits the texts into almost monolingual parts except for small text segments. The idea is to maintain as much context as possible. Finally, monolingual taggers [16, 17] are used to extract the POS information from the segments.

4.2.2. *Brown word clusters*

Since we do not have reference POS tags, we cannot evaluate the performance of our tagging process. The tags may contain errors and, therefore, their CS prediction accuracy may not be optimal. Hence, we also clustered the words of the CS texts with the unsupervised method by Brown et al. [18]. It assigns words to classes based on their distribution in a training text. We investigated different cluster sizes and found that 70 classes led to the lowest perplexities in our FLMs [15].

4.2.3. *Bilingual clusters of open class words*

Besides syntactic and distribution based features, we integrated semantic information in FLMs. Words can be categorized into open class words (content words) and closed class words (function words). While closed class words (e.g. conjugations, prepositions) specify grammatic relations rather than semantic meaning, open class words (e.g. nouns, verbs, adverbs) express meaning or attributes [19]. We extracted open class words from the training text and clustered them into topic like classes. Since we wanted to create semantic classes, we considered the study of [11] which is described in Section 2.2: Based on its results, we trained an RNNLM on open class words and clustered its word vectors. To avoid data sparseness, we used English and Mandarin Gigaword text data[1] to train the RNNLM. In particular, we created a text with alternating lines of English and Mandarin data. The hidden layer of the RNNLM was reset after each utterance. With this method, we built an RNNLM containing vectors for both English and Mandarin words but without mixing the different languages. For clustering, we used the Graclus implementation [20] of spectral clustering [21].

5. Experiments: combination of RNNLM and FLM

5.1. Data corpus: SEAME

The CS speech corpus which we used in this study is called SEAME (South East Asia Mandarin-English). It has been recorded in Singapore and Malaysia by [22] and contains spontaneously spoken interviews and conversations. Originally, it was used for the joint research project “Code-Switch” performed by Nanyang Technological University (NTU) and Karlsruhe Institute of Technology (KIT). The corpus contains about 63 hours of audio data with manual transcriptions. The words can be categorized into four language classes: Mandarin (58.6% of all tokens), English (34.4% of all tokens), particles (Singaporean and Malayan discourse particles, 6.8% of all tokens) and other languages (0.4% of all tokens). The average number of CS events between Mandarin and English is 2.6 per utterance. The average duration of monolingual English and Mandarin segments is only 0.67 seconds and 0.81 seconds, respectively. In total, the corpus includes 9,210 unique English and 7,471 unique Mandarin words. We divided the corpus into three disjoint sets: a training, a development and an evaluation set. The data were assigned based on criteria like gender, speaking style, ratio of Singaporean and Malaysian speakers, ratio of the four language categories and the duration in each set. Table 1 provides statistical information on the different sets.

[1]English Gigaword: LDC2011T07,
Mandarin Gigaword: LDC2011T13

Table 1: Statistics of the SEAME corpus

	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
# Tokens	525,168	23,776	11,294

5.2. Perplexity results

Table 2 presents the perplexity results of the different LMs. A traditional 3-gram LM built on the SEAME training text serves as baseline (1). The perplexities of the unconverted RNNLM are also provided for comparison (4). Furthermore, the two conversion approaches of the RNNLM are compared (2a, 2b) and combined using the SRILM toolkit [23] (2c). The FLM with the features described in Section 4.2 (3b) is compared to an FLM with the same features as the RNNLM (3a). Compared to the results published in [4], the baseline model and the FLM with POS and LID have been further optimized on the SEAME development set. The linear interpolation of (2c) and (3b) with a weight of 0.326 for (3b) performs the best. The interpolation weight has been optimized on the development set. Its perplexity even slightly outperforms the results of the unconverted RNNLM on the evaluation set.

Table 2: PPL results of different LMs for CS speech

Model	PPL dev	PPL eval
(1) CS 3-gram	268.39	282.86
(2a) converted RNNLM (text generation)	249.53	270.55
(2b) converted RNNLM (iterative conv.)	235.12	250.64
(2c) converted RNNLM (both approaches)	233.67	251.29
(3a) FLM POS + LID	257.62	264.20
(3b) FLM POS + Brown cl. + oc clusters	247.24	252.60
(2c) + (3b)	226.91	**238.84**
(4) RNNLM (unconverted)	**219.85**	239.21

A paired t-test shows that the combination of the FLM and the converted RNNLM is significantly better (at a level of 0.0001) than the perplexities of each LM on its own.

5.3. Mixed error rate results

We used BioKIT, a dynamic one-pass decoder, for our experiments [24]. The speaker independent acoustic model uses bottle-neck features and is described in [3]. We used the converted RNNLM for LM lookahead and interpolated the LM score of the FLM with the score of the RNNLM at word ends. For lookahead, an LM of the same order as for score computation (3-grams) is used. To efficiently determine an appropriate interpolation weight, we extracted a subset of the development set (about 20% of all sentences) and tested several weights. We found that a weight of 1.0 for the FLM performs the best.

Table 3: Result analysis after decoding with the single LMs and the LM combination

	CS 3-gram	FLM	conv. RNNLM	combination
Word correct after CS	37.52%	37.61%	38.69%	**38.72%**
Language correct after CS	68.23%	66.79%	**68.27%**	66.86%
WER in English segments	59.40%	**56.02%**	57.14%	57.89%
CER in Mandarin segments	36.48%	36.24%	36.12%	**35.82%**
Deletion of English words	1.65%	1.86%	**1.59%**	1.91%
Deletion of Mandarin words	5.79%	6.31%	**5.73%**	6.29%
Insertion of English words	1.09%	0.84%	1.13%	**0.79%**
Insertion of Mandarin words	2.99%	**2.67%**	3.16%	2.72%
Substitution of EN with EN	4.87%	4.68%	4.76%	**4.67%**
Substitution of EN with MAN	**4.38%**	4.44%	4.40%	4.43%
Substitution of MAN with MAN	15.30%	14.93%	14.85%	**14.71%**
Substitution of MAN with EN	2.85%	2.57%	2.58%	**2.18%**

Hence, the converted RNNLM is only used for LM lookahead while the score at word ends is determined by the FLM alone. This may be unexpected since the converted RNNLM has lower perplexity values than the FLM. However, we assume that the FLM may be more robust in the case of erroneous hypotheses because of the feature integration. We found that using the FLM for LM lookahead led to higher error rates on the development set compared to using the converted RNNLM. We assume that the converted RNNLM is able to better discriminate different contexts and hypotheses because it is based on words while the FLM may cluster the same feature contexts.
As the evaluation measure, we used a mixed error rate (MER) as proposed in [25]. It assigns word error rates to English and character error rates to Mandarin segments. Therefore, results can be compared across different segmentations of Mandarin. In this work, we used manual segmentations. Table 4 presents the MER results of our decoding experiments. Again, a 3-gram LM serves as baseline. For comparison, we also provide results for using only the converted RNNLM and for using only the FLM (with the 3-gram baseline LM for lookahead).

Table 4: MER results for using both the converted RNNLM and the FLM as LM during decoding

Model	MER dev	MER eval
Decoder baseline	39.96%	34.31%
FLM (3b)	39.30%	33.16%
Converted RNNLM (2c)	39.27%	33.73%
Combination (2c) + (3b)	**38.74%**	**32.81%**

A t-test shows that both the FLM and the RNNLM lead to significantly better MER results than the baseline 3-gram model (at a level of 0.002). The combination of the two LM types performs significantly better than the converted RNNLM alone (at a level of 0.002) but it is not significantly different to the FLM alone. A reason for this could be that the score at word ends is computed using only the FLM. Nevertheless, performing the lookahead with the converted RNNLM leads to lower error rates than using the baseline 3-gram model.

5.4. Error analysis

In order to analyze the behavior of the different LMs and to evaluate why their combination leads to the lowest MER results, the MERs are presented in more detail in this section. In particular, we consider different error categories which are shown in Table 3. Since some numbers denote errors and some are accuracy values, the model with the best performance is highlighted for each category. For this analysis, we use the decoding results on the development set. The development set consists of 28.6% English words, 63.0% Mandarin words, 8.2% particles and 0.2% other languages. There are 4595 CS points between Mandarin and English. At those points, both the FLM and the converted RNNLM can improve the recognition results compared to our baseline system. Moreover, their additional information seems to be complementary since the combination of both models further improves the results. Nevertheless, the accuracy at CS points is still below 40%. This could explain the rather high error rates in the monolingual English and Mandarin segments, too: The average length of monolingual segments is 1.6 words for English and 3.6 words for Mandarin. Hence, a trigram based LM suffers from misrecognized CS points for almost all the words in monolingual segments. This is why improving the recognition at CS points is very important for the over-all performance of the decoder. It is also notable that using the converted RNNLM leads to a higher language accuracy at CS points in contrast to the FLM. This could be another explanation why the RNNLM should be used for LM lookahead for the SEAME data set. Furthermore, it can be noticed that the converted RNNLM and the FLM lead to improvements in different categories. The converted RNNLM, for instance, reduces the number of deletions while the FLM improves the number of insertion errors. The LM combination can benefit from both the converted RNNLM and the FLM. It outperforms the single models especially in the monolingual Mandarin segments due to lower substitution rates.

6. Conclusions

This paper presented our investigations of language models for Code-Switching. We investigated and combined two different LM types: recurrent neural network language models and factored language models. We used syntactic and semantic features, namely part-of-speech tags, language identifiers, Brown word clusters and clusters of open class words to overcome data sparseness challenges. In order to combine both LM types during decoding, we converted our RNNLM into a backoff n-gram language model. We found that both for perplexity and decoding experiments, our language models with features outperformed traditional 3-gram approaches. An error analysis for decoding showed that the converted RNNLM led to higher language accuracy at Code-Switching points. The FLM and RNNLM improved the recognition in different error categories. As a result, the combination of both language model types led to the best results which are also statistically significant. We could improve the perplexity by 15.6% relative and the mixed error rate by 4.4% relative on the SEAME evaluation set. Our language model combination even slightly outperformed the unconverted RNNLM in terms of perplexity on the evaluation set.

7. References

[1] Mikolov, T. and Kombrink, S. and Burget, L. and Cernocky, JH and Khudanpur, S., "Extensions of recurrent neural network language model", Proc. of ICASSP. IEEE, 2011.

[2] Kirchhoff, K. and Bilmes, J. A. and Duh, K., "Factored language models tutorial", Tech. Rep. University of Washington, EE Department, UWEETR-2008-004, 2007.

[3] Adel, H. and Vu, N. T. and Kraus, F. and Schlippe, T. and Li, H. and Schultz, T., "Recurrent neural network language modeling for code switching conversational speech", Proc. of ICASSP. IEEE, 2013.

[4] Adel, H. and Vu, N. T. and Schultz, T., "Combination of recurrent neural networks and factored language models for Code-Switching language modeling", Proc. of ACL, 2013.

[5] Solorio, T. and Liu, Y., "Learning to predict code-switching points", Proc. of EMNLP. ACL, 2008.

[6] Chan, J. Y. C. and Ching, P. C. and Lee, T. and Cao, H., "Automatic speech recognition of Cantonese-English code-mixing utterances", Proc. of Interspeech, 2006.

[7] Schultz, T. and Fung, P. and Burgmer, C., "Detecting Code-Switch events based on textual features",

[8] Adel, H. and Kirchhoff, K. and Telaar, D. and Vu, N. T. and Schultz, T., "Features for factored language models for Code-Switching speech", Proc. of SLTU 2014.

[9] Mikolov, T. and Karafiát, M. and Burget, L. and Cernocky, J. and Khudanpur, S., "Recurrent neural network based language model", Proc. of Interspeech, 2010.

[10] Shi, Y. and Wiggers, P. and Jonker, C. M., "Towards recurrent neural networks language models with linguistic and contextual features", Proc. of Interspeech, 2012.

[11] Mikolov, T. and Yih, W.-T. and Zweig, G., "Linguistic regularities in continuous space word representations", Proc. of HLT-NAACL. ACL, 2013.

[12] Arisoy, E. and Chen, S. F. and Ramabhadran, B. and Sethy, A., "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition", Proc. of ICASSP, IEEE 2013.

[13] Deoras, A. and Mikolov, T. and Kombrink, S. and Karafiát, M. and Khudanpur, S., "Variational approximation of long-span language models for LVCSR", Proc. of ICASSP, IEEE 2011.

[14] El-Desoky, A. and Schlüter, R. and Ney, H., "A hybrid morphologically decomposed factored language model for Arabic LVCSR", Proc. of HLT-NAACL. ACL, 2010.

[15] Adel, H. and Kirchhoff, K. and Vu, N. T. and Telaar, D. and Schultz, T., "Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding", Proc. of Interspeech 2014.

[16] Toutanova, K. and Klein, D. and Manning, C. D. and Singer, Y., "Feature-rich part-of-speech tagging with a cyclic dependency network", Proc. HLT-NAACL. ACL, 2003.

[17] Tseng, H. and Chang, P. and Andrew, G. and Jurafsky, D. and Manning, C. D., "A conditional random field word segmenter", SIGHAN Workshop on Chinese Language Processing, 2005.

[18] Brown, P. F. and Desouza, P. V. and Mercer, R. L. and Pietra, V. J. D. and Lai, J. C., "Class-based n-gram models of natural language", Computational Linguistics, vol. 18, no. 4, pp. 467-479, 1992.

[19] Fromkin, V., "An introduction to language", Cengage Learning, 2013.

[20] Dhillon, I. S. and Guan, Y. and Kulis, B., "Weighted graph cuts without eigenvectors: a multilevel approach", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 11, pp. 1944-1957, 2007.

[21] Luxburg, U. von, "A tutorial on spectral clustering", Statistics and computing, vol. 17, no. 4, pp. 395-416, 2007.

[22] Lyu, D. C. and Tan, T. P. and Chng, E. S. and Li, H., "An Analysis of a Mandarin-English Code-switching Speech Corpus: SEAME", Oriental COCOSDA, 2010.

[23] Stolcke, A. and others, "SRILM - an extensible language modeling toolkit", Proc. of SLP, 2002.

[24] Telaar, D. and Wand, M. and Gehrig, D. and Putze, F. and Amma, C. and Heger, D. and Vu, N.T. and Erhardt, M. and Schlippe, T. and Janke, M. and Herff, C. and Schultz, T., "BioKIT - Real-time decoder for biosignal processing", Proc. of Interspeech, 2014.

[25] Vu, N. T. and Lyu, D.C. and Weiner, J. and Telaar, D. and Schlippe, T. and Blaicher, F. and Chng, E.S. and Schultz, T. and Li, H., "A first speech recognition system for Mandarin-English code-switch conversational speech", Proc. of ICASSP. IEEE, 2012.