Amharic Part-of-Speech Tagger for Factored Language Modeling

Martha Yifiru Tachbelie and Wolfgang Menzel
Department of Informatics, University of Hamburg
Vogt-Kölln Srt. 30, D-22527 Hamburg, Germany
tachbeli, menzel@informatik.uni-hamburg.de

Abstract

This paper presents Amharic part of speech taggers developed for factored language modeling. Hidden Markov Model (HMM) and Support Vector Machine (SVM) based taggers have been trained using the TnT and SVMTool. The overall accuracy of the best performing TnT- and SVM-based taggers is 82.99% and 85.50%, respectively. Generally, with respect to accuracy SVM-based taggers perform better than TnT-based taggers although TnT-based taggers are more efficient with regard to speed and memory requirement. We have developed factored language models (with two and four parents) for which the estimation of the probability for each word depends on the previous one or two words and their POS. These language models have been used in an Amharic speech recognition task in a lattice rescoring framework and a significant improvement in word recognition accuracy has been observed.

Keywords

POS tagging, Amharic, factored language model

1 Introduction

Language models are fundamental to many natural language applications such as automatic speech recognition (ASR). The most widely used class of language models, namely statistical ones, provide an estimate of the probability of a word sequence W for a given task. However, the probability distribution depends on the available training data — large amounts of training data are required so as to ensure statistical significance.

Even if a large training corpus is available, there may be still many possible word sequences which will not be encountered at all, or which appear with a statistically insignificant frequency (data sparseness problem) [21]. In morphologically rich languages, there are even individual words that might not be encountered in the training data irrespective of its size (Out-Of-Vocabulary words problem).

The data sparseness problem in statistical language modeling is more serious for languages with a rich morphology. These languages have a high vocabulary growth rate which results in a high perplexity and a large number of out of vocabulary words [19]. Therefore, sub-words (morphemes), instead of words, have been and are being used as modeling units in language modeling so as to build more robust language models even if only insufficient training data is available.

1.1 The morphology of Amharic

Amharic is one of the morphologically rich languages. It is a major language spoken mainly in Ethiopia and belongs to the Semitic branch of the Afro-Asiatic super family. Amharic is related to Hebrew, Arabic and Syrian.

Like other Semitic languages such as Arabic, Amharic exhibits a root-pattern morphological phenomenon. A root is a set of consonants (called radicals) which has a basic 'lexical' meaning. A pattern consists of a set of vowels which are inserted (intercalated) among the consonants of a root to form a stem. The pattern is combined with a particular prefix or suffix to create a single grammatical form [4] or another stem [20]. For example, the Amharic root sbr means 'break', when we intercalate the pattern ä_ä and attach the suffix ä we get säbbärä 'he broke' which is the first form of a verb (3rd person masculine singular in past tense as in other semitic languages) [4]. In addition to this non-concatenative morphological feature, Amharic uses different affixes to create inflectional and derivational word forms.

Some adverbs can be derived from adjectives. Nouns are derived from other basic nouns, adjectives, stems, roots, and the infinitive form of a verb by affixation and intercalation. For example, from the noun lIğğ 'child' another noun lIğnät 'childhood'; from the adjective däg 'generous' the noun dägnät 'generosity'; from the stem sInIf, the noun sInIfna 'laziness'; from root qld, the noun qäld 'joke'; from infinitive verb mäsIbär 'to break' the noun mäsIbäriya 'an instrument used for breaking' can be derived. Case, number, definiteness, and gender marker affixes inflect nouns.

Adjectives are derived from nouns, stems or verbal roots by adding a prefix or a suffix. For example, it is possible to derive dIngayama 'stony' from the noun dIngay 'stone'; zIngu 'forgetful' from the stem zIng; sänäf 'lazy' from the root snf by suffixation and intercalation. Adjectives can also be formed through compounding. For instance, hodäsäfi 'tolerant, patient', is derived by compounding the noun hod 'stomach' and the adjective säfi 'wide'. Like nouns, adjectives are inflected for gender, number, and case [20].

Unlike the other word categories such as noun and adjectives, the derivation of verbs from other parts of

speech is not common. The conversion of a root to a basic verb stem requires both intercalation and affixation. For instance, from the root gdl 'kill' we obtain the perfective verb stem gäddäl- by intercalating the pattern ä_ä. From this perfective stem, it is possible to derive a passive (tägäddäl-) and a causative stem (asgäddäl-) using the prefixes tä- and as-, respectively. Other verb forms are also derived from roots in a similar fashion.

Verbs are inflected for person, gender, number, aspect, tense and mood [20]. Other elements like negative markers also inflect verbs in Amharic.

1.2 Language modeling for Amharic

Since Amharic is a morphologically rich language, it suffers from data sparseness and out of vocabulary words problems. The negative effect of Amharic morphology on language modeling has been reported by [1], who, therefore, recommended the development of sub-word based language models for Amharic.

To this end, [17, 18] have developed various morpheme-based language models for Amharic and gained a substantial reduction in the out-of-vocabulary rate. They have concluded that, in this regard, using sub-word units is preferable for the development of language models for Amharic. In their experiment, [17, 18] considered individual morphemes as units of a language model. This, however, might result in a loss of word level dependencies since the root consonants of the words may stand too far apart. Therefore, approaches that capture word level dependencies are required to model the Amharic language. [12] introduced factored language models that can capture word level dependency while using morphemes as units in language modeling. That is why we opted for developing factored language models also for Amharic.

1.3 Factored language modeling

Factored language models (FLM) have first been introduced in [13] for incorporating various morphological information in Arabic language modeling. In FLM a word is viewed as a bundle or vector of K parallel factors, that is, $w_n \equiv f_n^1, f_n^2, ..., f_n^k$. The factors of a given word can be the word itself, stem, root, pattern, morphological classes, or any other linguistic element into which a word can be decomposed. The goal of an FLM is, therefore, to produce a statistical model over these factors.

There are two important points in the development of FLM: choosing the appropriate factors which can be done based on linguistic knowledge or using a data driven technique and finding the best statistical model over these factors. Unlike normal word or morpheme-based language models, in FLM there is no obvious natural backoff order. In a trigram word based model, for instance, we backoff to a bigram if a particular trigram sequence has not been observed in our corpus by dropping the most distant neighbor, and so on. However, in FLM the factors can be temporally equivalent and it is not obvious which factor to drop first during backoff. If we consider a quadrogram FLM and if we drop one factor at a time, we can have six possible backoff paths as it is depicted in Figure 1 and we need to choose a path that results in a better model. Therefore, choosing a backoff path is an important decision one has to make in FLM. There are three possible ways of choosing a backoff path: 1) Choosing a fixed path based on linguistic or other reasonable knowledge; 2) Generalized all-child backoff where multiple backoff paths are chosen at run time; and 3) Generalized constrained-child backoff where a subset of backoff paths is chosen at run time [14]. A genetic algorithm for learning the structure of a factored language model has been developed by [7].

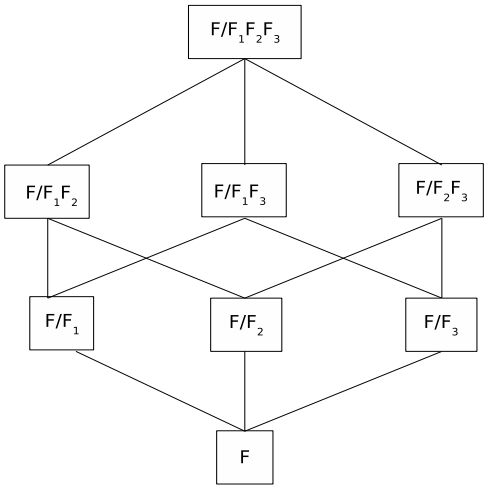

Fig. 1: *Possible backoff paths*

In addition to capturing the word level dependencies, factored language models also enable us to integrate any kind of relevant information to a language model. Part of speech (POS) or morphological class information, for instance, might improve the quality of a language model as knowing the POS of a word can tell us what words are likely to occur in its neighborhood [11]. For this purpose, however, a POS tagger is needed which is able to automatically assign POS information to the word forms in a sentence. This paper presents the development of Amharic POS taggers and the use of POS information in language modeling.

1.4 Previous works on POS tagging

[9] attempted to develop a Hidden Markov Model (HMM) based POS tagger for Amharic. He extracted a total of 23 POS tags from a page long text (300 words) which is also used for training and testing the POS tagger. The tagger does not have the capability of guessing the POS tag of unknown words, and consequently all the unknown words are assigned a UNC tag, which stands for unknown category. As the lexicon used is very small and the tagger is not able to deal with unknown words, many of the words from the test set were assigned the UNC tag.

[3] developed a POS tagger using Conditional Random Fields. Instead of using the POS tagset developed by [9], [3] developed another abstract tagset (consisting of 10 tags) by collapsing some of the categories proposed by [9]. He trained the tagger on a manually annotated text corpus of five Amharic news articles (1000 words) and obtained an accuracy of 74%.

As the data sets used to train both of the above systems are very small it is not possible to apply the taggers to large amounts of text which is needed for training a language model.

In a very recent, but independent development, a POS tagging experiment similar to the one described in this paper has been conducted by [8]. There, three tagging strategies have been compared – Hidden Markov Models (HMM), Support Vector Machines (SVM) and Maximum Entropy (ME) – using the manually annotated corpus [6] (which has also been used in our experiment) developed at the Ethiopian Language Research Center (ELRC) of the Addis Ababa University. Since the corpus contains a few errors and tagging inconsistencies, they cleaned the corpus. Cleaning includes tagging non-tagged items, correcting some tagging errors and misspellings, merging collocations tagged with a single tag, and tagging punctuations (such as '“' and '/') consistently. They have used three tagsets: the one used in [3], the original tagset developed at ELRC that consists of 30 tags and the 11 basic classes of the ELRC tagset. The average accuracies (after 10-fold cross validation) are 85.56, 88.30, 87.87 for the TnT-, SVM- and maximum entropy based taggers, respectively for the ELRC tagset. They also found that the maximum entropy tagger performs best among the three systems, when allowed to select its own folds. Their result also shows that the SVM-based tagger outperforms the other ones in classifying unknown words and in the overall accuracy for the tagset (ELRC) that is used in our experiment too.

2 Amharic part-of-speech taggers

2.1 The POS tagset

In our experiment, we used the POS tagset developed within “The Annotation of Amharic News Documents” project at the Ethiopian Language Research Center. The purpose of the project was to manually tag each Amharic word in its context [6]. In this project, a new POS tagset for Amharic has been derived. The tagset has 11 basic classes: nouns (N), pronouns (PRON), adjectives (ADJ), adverbs (ADV), verbs (V), prepositions (PREP), conjunction (CONJ), interjection (INT), punctuation (PUNC), numeral (NUM) and UNC which stands for unclassified and used for words which are difficult to place in any of the classes. Some of these basic classes are further subdivided and a total of 30 POS tags have been identified as shown in Table 1. Although the tagset contains a tag for nouns with preposition, with conjunction and with both preposition and conjunction, it does not have a separate tag for proper and plural nouns. Therefore, such nouns are assigned the common tag N.

2.2 The corpus

The corpus used to train and test the taggers is the one developed in the above mentioned project — “The Annotation of Amharic News Documents” [6]. It consists of 210,000 manually annotated tokens of Amharic news documents.

Categories	Tags
Verbal Noun	VN
Noun with prep.	NP
Noun with conj.	NC
Noun with prep. & conj.	NPC
Any other noun	N
Pronoun with prep.	PRONP
Pronoun with conj.	PRONC
Pronoun with prep. & conj.	PRONPC
Any other pronoun	PRON
Auxiliary verb	AUX
Relative verb	VREL
Verb with prep.	VP
Verb with conj.	VC
Verb with prep. & conj.	VPC
Any other verb	V
Adjective with prep.	ADJP
Adjective with conj.	ADJC
Adjective with prep. & conj.	ADJPC
Any other adjective	ADJ
Preposition	PREP
Conjunction	CONJ
Adverbs	ADV
Cardinal number	NUMCR
Ordinal number	NUMOR
Number with prep.	NUMP
Number with conj.	NUMC
Number with prep. & conj.	NUMPC
Interjection	INT
Punctuation	PUNC
Unclassified	UNC

Table 1: *Amharic POS tagset (extracted from [6])*

In this corpus, collocations have been annotated inconsistently. Sometimes a collocation assigned a single POS tag and sometimes each token in a collocation got a separate POS tag. For example, 'tmhrt bEt', which means *school*, has got a single POS tag, N, in some places and a separate POS tags for each of the tokens in some other places. Therefore, unlike [8] who merged a collocation with a single tag, effort has been exerted to annotate collocations consistently by assigning separate POS tags for the individual words in a collocation.

2.3 The software

We used two kinds of software, namely TnT and SVMTool, to train different taggers.

TnT, Trigram'n'Tags, is a Markov model based, efficient, language independent statistical part of speech tagger [5]. It has been applied on many languages including German, English, Slovene, Hungarian and Swedish successfully. [15] showed that TnT is better than maximum entropy, memory- and transformation-based taggers.

SVMTool is support vector machine based part-of-speech tagger generator [10]. As indicated by the developers, it is a simple, flexible, effective and efficient tool. It has been successfully applied to English and Spanish.

2.4 TnT-based tagger

We have developed three TnT-based taggers by taking different amounts of tokens (80%, 90% and 95%) from the corpus as training data and named the taggers as tagger1, tagger2 and tagger3, respectively. Five percent of the corpus (after taking 95% for training) has been reserved as a test set. This test set has also been used to evaluate the SVM-based taggers to make the results comparable.

Table 2 shows the accuracy of each tagger. As it is clear from the table, the maximum accuracy was found when 95% of the data (199,500 words) have been used for training. This tagger has an overall accuracy of 82.99%. The results also show that the training has not yet reached the point of saturation and the overall accuracy increases, although slightly, as the amount of training data increases. This conforms with findings for other languages that "... the larger the corpus and the higher the accuracy of the training corpus, the better the performance of the tagger" [5]. One can also observe that improvement in the overall accuracy is affected with the amount of data added. Higher improvement in accuracy has been obtained when we increase the training data by 10% than increasing by only five percent. Compared to similar experiments done for other languages and the result which has been recently reported for Amharic by [8], our taggers have worse performance. The better result obtained in [8] might be due to the use of cleaned data and a 10-fold cross-validation technique to train and evaluate the taggers. Nevertheless, we still consider the result acceptable for the given purpose.

Taggers	Accuracy in %		
	Known	Unknown	Overall
Tagger1	88.24	48.77	82.70
Tagger2	88.09	48.11	82.94
Tagger3	88.00	47.82	82.99

Table 2: *Accuracy of TnT taggers*

2.5 SVM-based tagger

We trained SVM-based tagger, SVMM0C0, using 90% of the tagged corpus. To train this model, we did not tune the cost parameter (C) that controls the trade off between allowing training errors and forcing rigid margins. We used the default value for other features like the size of the sliding window. The model has been trained in a one pass, left-to-right and right-to-left combined, greedy tagging scheme. The resulting tagger has an overall accuracy of 84.44% (on the test set used to evaluate the TnT-based taggers) as Table 3 shows.

A slight improvement of the overall accuracy and the accuracy of known words has been achieved setting the cost parameter to 0.1 (see SVMM0C01 in Table 3). The accuracy improvement for unknown words is bigger (from 73.64 to 75.30) compared to the accuracy of known words and the overall accuracy. However, when the cost parameter was increased above 0.1, the accuracy declined. We experimented with cost parameters 0.3 (SVMM0C03) and 0.5 (SVMM0C05) and in both cases no improvement in accuracy has been observed (neither for the overall accuracy nor for the accuracy of known and unknown words).

Taggers	Accuracy in %		
	Known	Unknown	Overall
SVMM0C0	86.03	73.64	84.44
SVMM0C01	86.97	75.30	85.47
SVMM0C03	86.71	73.49	85.01
SVMM0C05	86.48	71.97	84.61

Table 3: *Accuracy of SVM-based taggers*

To determine how the amount of training data affects accuracy, we trained another SVM-based tagger using 95% of the data and the cost parameter of 0.1. Only a slight improvement in the overall accuracy (85.50%) and accuracy for classifying unknown words (from 75.30% to 75.35%) has been achieved compared to the SVMM0C01 tagger which has been trained on 90% of the data. This corresponds to the findings for TnT-based taggers that improved only marginally when a small amount of data (5%) is added. For known words the accuracy declined slightly (from 86.97% to 86.95%). Although this tagger is better (in terms of the overall accuracy) than all the other ones, it performs not better than the one reported by [8] who used a 10-fold cross-validation technique and cleaned data.

Another tagger has been developed using the same data but with a different cost parameter (0.3). However, no improvement in performance has been observed. This model has an overall accuracy of 85.09% and accuracy of 86.76% and 73.40% for known and unknown tokens, respectively.

2.6 Comparison of TnT- and SVM-based taggers

The SVMM0C0 has been trained with the same data that has been used to train the TnT-based tagger, tagger2. The same test set has also been used to test the two types of taggers so that we can directly compare results and decide which algorithm to use for tagging our text for factored language modeling. As it can be seen from Table 3, the SVM-based tagger has an overall accuracy of 84.44%, which is better than the result we found for the TnT-based tagger (82.94%). This finding is in line with what has been reported by [10]. We also noticed that SVM-based taggers have a better capability of classifying unknown words (73.64%) than a TnT-based tagger (48.11%) as it has also been reported in [8].

With regard to speed and memory requirements, TnT-based taggers are more efficient than the SVM-based ones. A SVM-based tagger tags 366.7 tokens per second whereas the TnT-based tagger tags 114083 tokens per second. Moreover, the TnT-based tagger, tagger2, requires less (647.68KB) memory than the SVM-based tagger, SVMM0C0, (169.6MB). However, our concern is on the accuracy of the taggers instead of their speed and memory requirement. Thus, we preferred to use SVM-based taggers to tag our text for the experiment in factored language modeling.

Therefore, we trained a new SVM-based tagger using 100% of the tagged corpus based on the assumption that the increase in the accuracy (from 85.47 to

85.50%) observed when increasing the training data (from 90% to 95%) will continue if more training data are added. Again, the cost parameter has been set to 0.1 which yielded good performance in the previous experiments. It is this tagger that was used to tag the text for training factored language models.

3 Application of the POS information

To determine how the addition of an extra information, namely POS, improves the quality of a language model and consequently the performance of a natural language application that uses the language model, we have developed factored language models that use POS as an additional information. The language models have then been applied to an Amharic speech recognition task in a lattice rescoring framework [12]. Using factored language models in standard word-based decoders is problematic, because they do not predict words but factors.

3.1 Baseline speech recognition system

3.1.1 Speech and text corpus

The speech corpus used to develop the speech recognition system is a read speech corpus developed by [2]. It contains 20 hours of training speech collected from 100 speakers who read a total of 10850 sentences (28666 tokens). Compared to other speech corpora that contain hundreds of hours of speech data for training, for example, British National Corpus (1,500 hours of speech), it is a fairly small one and a model trained on it will suffer from lack of training data.

Although the corpus includes four different test sets (5k and 20k both for development and evaluation), for the purpose of the current investigation we have generated the lattices only for the 5k development test set, which includes 360 sentences read by 20 speakers.

The text corpus used to train the baseline backoff bigram language model consists of 77,844 sentences (868929 tokens or 108523 types).

3.1.2 Acoustic and language models

The acoustic model is a set of intra-word triphone HMM models with 3 emitting states and 12 Gaussian mixtures that resulted in a total of 33,702 physically saved Gaussian mixtures. The states of these models are tied, using decision-tree based state-clustering that reduced the number of triphone models from 5,092 logical models to 4,099 physical ones.

The baseline language model is a closed vocabulary (for 5k) backoff bigram model developed using the HTK toolkit. The absolute discounting method has been used to reserve some probabilities for unseen bigrams and the discounting factor, D, has been set to 0.5, which is the default value in the HLStats module. The perplexity of this language model on a test set that consists of 727 sentences (8337 tokens) is 91.28.

3.1.3 Performance of the baseline system

We generated lattices from the 100 best alternatives for each test sentence of the 5k development test set using the HTK tool and decoded the best path transcriptions for each sentence using the lattice processing tool of SRILM [16]. Word recognition accuracy of the baseline system was 91.67% with a language model scale of 15.0 and a word insertion penalty of 6.0.

3.2 Lattice rescoring with FLM

We substituted each word in a lattice and in the training sentences with its factored representation. A word bigram model that is equivalent to the baseline word bigram language model has been trained using the factored version of the data[1]. This language model is used as a baseline for factored representations and has a perplexity of 58.41 (see Table 4). The best path transcription decoded using this language model has a word recognition accuracy of 91.60%, which is slightly lower than the performance of the normal baseline speech recognition system (91.67%). This might be due to the smoothing technique applied in the development of the language models. Although absolute discounting with the same discounting factor has been applied to both bigram models, the unigram models have been discounted differently. While in the baseline word based language model the unigram models have not been discounted at all, in the equivalent factored model the unigrams have been discounted using Good-Turing discounting technique which is the default discounting technique in SRILM.

In addition to the baseline, we have trained models with two $(w_n|w_{n-1}pos_{n-1})$ and four parents $(w_n|w_{n-1}pos_{n-1}w_{n-2}pos_{n-2})$ for which the estimation of the probability of each word depends on the previous word/s and its/their POS. A fixed backoff strategy has been applied during backoff, dropping the most distant factor first and so on. The perplexity of the language models is indicated in Table 4.

Language models	Perplexity
Baseline word bigram (FBL)	58.41
FLM with two parents	115.89
FLM with four parents	17.03

Table 4: *Perplexity of factored language models*

The factored language models have then been used to rescore the lattices and an improvement of the word recognition accuracy was observed. As it can be seen from Table 5, the addition of the POS information makes language models more robust and consequently the word recognition accuracy improved from 91.60 to 92.92. Although normally the use of higher order ngram models also improves the word recognition accuracy, this is not the case for our factored language models.

[1] A data in which each word is considered as a bundle of features including the word itself, POS tag of the word, prefix, root, pattern and suffix.

Language models used	Word accuracy
Baseline word bigram (FBL)	91.60%
FBL + FLM with two parents	92.92%
FBL + FLM with four parents	92.75%

Table 5: *Word recognition accuracy improvement with factored language models*

4 Conclusion

This paper describes a series of POS tagging experiments aimed at providing a factored language model with an additional information source. For the POS tagger development, we used a manually tagged corpus which consist of 210,000 tokens. Two software tools, TnT and SVMTool, have been applied to train different taggers. As SVM-based taggers outperformed the probabilistic ones, we decided to use them to tag the text for our factored language modeling experiment.

We have developed factored language models (with two and four parents) which estimate the probability of each word depending on the previous one or two words and their POS. Using these language models in an Amharic speech recognition task in a lattice rescoring framework, we obtained improvement of word recognition accuracy (1.32% absolute).

Acknowledgments

We would like to thank the people who developed and made freely available the Amharic manually tagged corpus as well as TnT and SVMTool software tools. Thanks are due to the reviewers who provided constructive comments.

References

[1] S. T. Abate. *Automatic Speech Recognition for Amharic*. PhD thesis, Univ. of Hamburg, 2006.

[2] S. T. Abate, W. Menzel, and B. Tafila. An Amharic speech corpus for large vocabulary continuous speech recognition. In *Proceedings of 9th. European Conffference on Speech Communication and Technology, Interspeech-2005*, 2005.

[3] S. F. Adafre. Part of speech tagging for Amharic using conditional random fields. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 47–54, 2005.

[4] M. Bender, J. Bowen, R. Cooper, and C. Ferguson. *Languages in Ethiopia*. Oxford Univ. Press, London, 1976.

[5] T. Brants. TnT — a statistical part-of-speech tagger. In *Proceedings of the 6th ANLP*, 2000.

[6] G. A. Demeke and M. Getachew. Manual annotation of Amharic news items with part-of-speech tags and its challenges. *ELRC Working Papers*, II(1), 2006.

[7] K. Duh and K. Kirchhoff. Automatic learning of language model structure. In *Proceeding of International Conference on Computational Linguistics*, 2004.

[8] B. Gambäck, F. Olsson, A. A. Argaw, and L. Asker. Methods for Amharic part-of-speech tagging. In *Proceedings of the EACL Workshop on Language Technologies for African Languages - AfLaT 2009*, pages 104–111, March 2009.

[9] M. Getachew. Automatic part of speech tagging for Amharic language: An experiment using stochastic hmm. Master's thesis, Addis Ababa University, 2000.

[10] J. Giménez and L. Màrquez. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004.

[11] D. S. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, New Jersey, 2nd. ed. edition, 2008.

[12] K. Kirchhoff, J. Bilmes, S. Das, N. Duta, M. Egan, G. Ji, F. He, J. Henderson, D. Liu, M. Noamany, P. Schone, R. Schwartz, and D. Vergyri. Novel approaches to Arabic speech recognition: Report from the 2002 johns-hopkins summer workshop. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 1–344 – 1–347, 2003.

[13] K. Kirchhoff, J. Bilmes, J. Henderson, R. Schwartz, M. Noamany, P. Schone, G. Ji, S. Das, M. Egan, F. He, D. Vergyri, D. Liu, and N. Duta. Novel speech recognition models for arabic. Technical report, Johns-Hopkins University Summer Research Workshop, 2002.

[14] K. Kirchhoff, J. Bilmes, and kevin Duh. Factored language models - a tutorial. Technical report, Dept. of Electrical Eng., Univ. of Washington, 2008.

[15] B. Megyesi. Comparing data-driven learning algorithms for pos tagging of Swedish. In *Proceedings of the 2001 Conference on Emperical Methods in Natural Language Processing*, pages 151–158, 2001.

[16] A. Stolcke. SRILM — an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume II, pages 901–904, 2002.

[17] M. Y. Tachbelie and W. Menzel. Sub-word based language modeling for Amharic. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, pages 564–571, September 2007.

[18] M. Y. Tachbelie and W. Menzel. *Morpheme-based Language Modeling for Inflectional Language – Amharic*. John Benjamin's Publishing, Amsterdam and Philadelphia, forthcoming.

[19] D. Vergyri, K. Kirchhoff, K. Duh, and A. Stolcke. Morphology-based language modeling for Arabic speech recognition. In *Proceedings of International Conference on Spoken Language Processing*, pages 2245–2248, 2004.

[20] B. Yemam. *yäamarIña säwasäw*. EMPDE, Addis Ababa, 2nd. ed. edition, 2000 EC.

[21] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book*. Cambridge University Engineering Department, 2006.