Maximum Entropy Language Model Adaptation for Mobile Speech Input

Tanel Alumäe[1], *Kaarel Kaljurand*[2]

[1]Institute of Cybernetics at Tallinn University of Technology, Estonia
[2]Institute of Computational Linguistics, University of Zurich, Switzerland
tanel.alumae@phon.ioc.ee, kaljurand@gmail.com

Abstract

This paper describes unsupervised adaptation of language model for many related target domains. In mobile speech input, subject and vocabulary of the language depend highly on the usage context. We use automatically transcribed speech data to select a subset from the language model training data for building a maximum entropy model adapted to speech input. This model is further adapted for most popular mobile applications. When used in interpolation with the background N-gram model, the adapted models give over 10% relative word error rate reduction in Estonian mobile speech input experiments.

Index Terms: language model adaptation, maximum entropy, mobile speech input

1. Introduction

Speech-based text input is currently probably the most widely used type of speech recognition application. Recently, we developed and released an Estonian speech input application for Android that interfaces the Android speech API and makes Estonian speech recognition available in all other applications. For example, speech may be used to dictate SMSs, find locations on the map, dictate notes and reminders, etc. The diversity of the applications makes it difficult to build a single good language model (LM) that handles all usage scenarios of mobile speech input equally well. One option to improve this is to adapt the LM for major use cases.

LM adaptation for mobile speech input has been investigated in several papers. In [1], a large set of hand-transcribed utterances are used to optimize mixture weights for linear combination of component n-gram LMs. Adapted mixture weights are created for thousands of individual text fields, and interpolation is applied on-demand during decoding.

Unsupervised LM adaptation has been studied in numerous publications. In [2], term frequency–inverse document frequency (TF-IDF) based document selection is used to construct an adaptation corpus for the target domain. Unigram scaling and linear interpolation are used to adapt the background model. Adaptation for many related adaptation targets has also been investigated before. In [3], LM is adapted towards individual speakers. Latent topics are extracted from the LM training corpus. Accumulated recognition hypotheses for each speaker are treated as documents and are used for estimating topic interpolation weights for each speaker. In addition, speakers are modeled by a latent speaker model, and the final speaker-specific models are inferred via a linear interpolation of the latent speaker models.

In this paper, we explore unsupervised LM adaptation for mobile speech input based on the usage context. In our case, the context is the name of the mobile application through which the speech recognition service was invoked. This is slightly different from the approach used in [1]: they use the granularity of the text field. Such difference is partly due to technical reasons. However, the main difference with regard to [1] is the adaptation method that we use: instead of optimizing mixture weights of component LMs, we rely on document selection techniques to extract a subset of the LM training corpus for each application context, based on the accumulated recognition hypotheses. Instead of using hand-transcribed development corpus for document selection, we use automatic transcripts generated by a multi-pass recognition system as the sample for document selection. The extracted texts are used for estimating maximum entropy (ME) LMs [4]. In order to to take advantage of the similarity of the modeled target domains, we use a hierarchical method for building the ME models: application-specific models are adapted from the global model which is trained from the union of the selected documents over all actively used applications. This hierarchical approach is found to be crucial for the perplexity and word error rate (WER) improvements.

2. Motivation

Estonian is the official language of Estonia, spoken by about one million people. We implemented an open and extendable ASR architecture that enables one to develop speech-based applications for mobile platforms [5]. The system stack consists of an ASR server and a client service for Android that transparently provides large vocabulary ASR functionality for other applications, similarly to Google's voice input for English and other languages

(but not yet for Estonian). The Android application that provides Estonian speech input features ("*Kõnele*") was publicly launched at the end of 2011, gaining quickly over 4000 installations.

Similarly to Google's speech input module, our system is also based on cloud technology: the speech signal is recorded on the mobile device and sent to the server for decoding. The speech signal is accompanied by various metadata, including the name of the mobile application from where speech recognition was invoked. It is obvious that the vocabulary, style and subject of the speech depends heavily on the used application: e.g., geographic names and names of institutions are often used in a map application, names of popular artists in the YouTube application and names of food and household items in a "grocery list" application. The goal of this study is to improve the speech recognition accuracy for mobile speech input by better modeling such dependencies in the LM.

3. Methodology

3.1. Document selection

Document selection is a process of extracting a subset of documents (such as newspaper articles, web pages) from the LM training corpus that hopefully represents the target domain more accurately than the whole corpus.

Log-likelihood based criterion for selecting documents from a LM training corpus was suggested in [6]. The proposed method tries to find a subset of documents from the LM training corpus that minimize the perplexity of the held-out target corpus. For each document D_i in the training corpus, the algorithm calculates the change ΔF_i in LM perplexity of the target corpus when D_i is removed from the LM training data. Documents are then sorted according to ΔF_i and top K documents (or documents with F_i over some threshold) are selected as the subcorpus for the given domain.

We adopted a slightly different approach. For each document D_i, we calculated the perplexity difference

$$\Delta F_i' = PPL_{P_{BG}(w)} - PPL_{\lambda P_{BG}(w) \times (1-\lambda) P_{D_i}(w)}$$

where $PPL_{P_{BG}(w)}$ is the perplexity of the background unigram LM over the target data w and $PPL_{\lambda P_{BG}(w) \times (1-\lambda) P_{D_i}(w)}$ is the perplexity of background unigram model interpolated with the unigram LM built only from D_i (using absolute discounting). In other words, we look for such documents in the training corpus that individually produce a LM that improve held-out data perplexity, when applied in interpolation with the background model. We used $\lambda = 0.95$ in all our experiments. For all modeled domains, we selected $K = 5000$ documents with the highest $\Delta F_i'$. These values were chosen heuristically and were not tuned on the development data.

The justification of this document selection approach is the idea that we are not looking for documents that result in a good training corpus for the target domain *per se*. Rather, we acknowledge the intent to interpolate with the background model already during document selection process, and thus look for data that helps to improve of the performance the background model.

Source	Documents	Tokens
Newspapers	655 847	206M
News portals	186 781	40M
Scientific publications	78 709	17M
Parliament transcripts	6024	15M
Magazines	4137	12M
Fiction	202	6.3M
Broadcast conversations	227	0.34M
Blogs	3722	0.17M
Conference transcripts	23	0.06M
Total	935 672	299M

Table 1: Language model training data

3.2. Adapted maximum entropy models

We used maximum entropy language modeling techniques for building domain-specific language models from the selected documents. For each domain (i.e., mobile application), a ME model with trigram features was built. The models were built hierarchically: first, we used the union of the documents selected for each individual domain to build a parent model. The parent model was adapted to each domain as described in [7]: during optimization of the parameters for a certain domain, the parent model was taken as a prior. This method encourages the domain-specific models to have feature weights close to the prior model using ℓ_2^2 regularization, if there is little evidence to change them. The ME models were built using the SRILM extension [4].

4. Experiments

4.1. System description

Acoustic models for the Estonian speech recognition service were trained on various wideband Estonian speech corpora: the BABEL speech database, a corpus of Estonian broadcast news, a corpus of broadcast conversations, a corpus of lecture and conference recordings, and a corpus of spontaneous dialogs, totaling in around 90 hours. The models are triphone HMMs, using MLLT/LDA-transformed MFCC features, with 3000 tied states, each modeled with 32 Gaussians. The model inventory consists of 25 phonemes and 7 silence/filler models. Cepstral mean normalization (CMN) was applied during training.

As Estonian is a heavily inflected and compounding language, we split compound words into separate LM

units using a morphological analyzer [8], and reconstruct compound words from recognition hypotheses using a conditional random field model during postprocessing. The LM vocabulary consists of the most likely 200K tokens after compound splitting. The LM was trained on various text corpora, enumerated in Table 1. Separate models were built from each source and linearly interpolated into the final model. As we did not have any development data before the launch of the Android speech recognition module, the interpolation weights for the initial model were optimized on the development data from the broadcast conversations domain. After the system had been live for about a week, we re-optimized the interpolation weights based on automatically transcribed live usage data. The data was transcribed using a three-pass transcription system using a four-gram LM, CMLLR and MLLR adaptation, implemented using the RWTH ASR software [9]. Due to the limitation of the used live decoder, the ASR server uses trigram LMs. Kneser-Ney smoothing was employed, and entropy pruning was applied to reduce the size of the model to about one third.

Estonian is almost a phonetic language which lets us use a simple rule-based algorithm for building a pronunciation dictionary, with a list of exceptions for common foreign names.

Our ASR server is based on various open-source technologies [5]. The core of the system is the PocketSphinx decoder of the CMU Sphinx speech recognition toolkit. The communication between the server and client is designed so that the server can start decoding an utterance instantly, while the user is still speaking. CMN coefficients are calculated on the fly and cached for individual devices between requests.

4.2. Data

Experiment data consists of about 5000 mobile speech input requests (about 4 hours in total) made during one week to our servers. 500 randomly chosen requests were selected for both development and test set. The development and test data was hand-transcribed and used for tuning the system and measuring recognition quality. Unintelligible utterances were removed. The rest of the requests were used as unsupervised training data. We processed this training data using the more accurate multipass transcription system. Table 2 lists the most popular Android applications through which the Estonian speech input module was invoked. The most popular application by a very large margin is the module's own launcher, which simply recognizes an utterance and directs the result to internet search.

4.3. Results

Based on the training data, we picked 25 most popular mobile applications through which our speech input module was invoked. This set of 25 application contexts covers 91% of the requests in the development set and 85% of the requests in the test set. We used the automatic transcriptions in the training data to create adapted ME LMs for the individual applications, as described in section 3.

Application	Usage
Voice search via native launcher	60%
SMS	5.8%
Google Maps	4.6%
Native demo app	3.9%
Android Market	1.4%
Todo list app from Market	1.3%
Samsung's notes app	1.1%
Dolphin Browser	1.1%
Google Translate	0.8%
Youtube	0.7%

Table 2: 10 most popular Estonian speech input applications and their relative usage.

For all utterances in the development and test set, a 100-best list of recognition hypotheses was generated, using the live recognition setting. For all hypotheses in the N-best lists, application-specific ME LMs were used for adding additional LM scores. For hypotheses entered via applications in the "long tail" (i.e., outside the top 25 list), the parent model was used (i.e., estimated over the pooled documents selected for the top 25 applications). The global score combination weights for the background LM, acoustic model, word insertion penalty and the adapted LM were then optimized based on the reference transcripts of the development set. We used the "Amoeba" search for word error minimization, as implemented in SRILM [11]. In order to have a fair baseline, we also reoptimized the scores for N-best decoding when using only the background LM. The optimized combination weights were then used for rescoring the hypotheses in test set.

We compared our method to two widely-used methods for LM adaptation. The first method (denoted later as *mix-per-app*) simply optimizes linear interpolation weights of LM components based on the application. This approach is very similar to the one described in [1]. However, we use application granularity, as opposed to text field granularity, and we use unsupervised adaptation, i.e., the component LM interpolation weights were optimized based on automatic transcripts. We computed optimized LM interpolation weights for the top 25 applications and used the weights optimized based on all automatic transcripts in the training data for the other applications. During rescoring of the N-best lists, unpruned component LMs were dynamically interpolated.

We also compared our method to unigram scaling [10], also known as minimum discriminant estimation (MDE). The unigram models were estimated from the same sets of selected documents as the ME models, and

LM	Perplexity		Mean WER	
	Dev	Test	Dev	Test
Background (BG) LM	393	460	37.2%	40.2%
Mix-per-app [1]	363 (-8%)	403 (-12%)	36.9 (-0.9%)	36.8 (-8.5%)
Unigram scaling [10] + BG LM	291 (-24%)	340 (-26%)	35.8 (-3.7%)	40.2 (-0.0%)
Simple 3-gram MaxEnt + BG LM	304 (-23%)	362 (-21%)	35.2 (-5.5%)	37.7 (-6.2%)
Pooled 3-gram MaxEnt + BG LM	332 (-16%)	416 (-10%)	35.5 (-4.6%)	37.0 (-8.0%)
Adapted 3-gram MaxEnt + BG LM	**276 (-30%)**	**335 (-27%)**	**33.6 (-9.6%)**	**34.8 (-13%)**

Table 3: Perplexity and word error rate comparison of various LM adaptation techniques, together with relative improvements over the baseline background LM.

a unigram built from the union of the selected documents was applied for the long tail applications. Default hyperparameters (as implemented in SRILM) were used for adapting the marginals of the background LM to the target-specific unigrams during N-best rescoring.

The third comparison ("simple maximum entropy") was made with ME LMs estimated directly from the selected documents for each target, without using the parent model as a prior. The parent model was still used for applications in the long tail.

As the fourth comparison ("pooled maximum entropy"), we rescored all N-best lists with the parent ME model built from the pool of all selected documents that was in the main experiment applied only for the long tail applications.

Table 3 lists perplexity and WER results of the compared methods. Hierarchically adapted ME models produced the best perplexity and WER numbers, outperforming the alternative adaptation methods by a noticeable margin in the development and test sets. We used the MAPSSWE test to evaluate the statistical significance of the WER differences in the test set. All adaptation methods were found to be significantly better than the baseline. The adapted ME models were also significantly better than unigram scaling, the "simple" ME models and the "pooled" setting, while there was no significant difference found between adapted ME and "mix-per-app".

To assert that the improvement in WER is not dominated only by the most popular applications, we investigated the average WER for the top 25 applications and for the "long tail" applications. The relative improvement for the popular applications was 12.6% and for the "long tail" applications 17% (when using adapted ME over the baseline LM).

In the presented experiments, the hierarchically adapted ME models were applied offline. However, this approach can be painlessly ported to online setting, since the models used for rescoring are estimated from a small subset of the LM training data and fit easily into RAM.

5. Conclusion

This paper described unsupervised adaptation of LM to many closely-related target domains. We used a log-likelihood based criteria for selecting such documents from the background training corpus that improve the perplexity of automatically transcribed speech from the target domains. The pooled documents from all domains were used to build a parent maximum entropy LM that was then used as a prior for training target-specific models. Based on an Estonian mobile speech input task, this approach resulted in 27% LM perplexity reduction and over 10% relative word error rate reduction compared to using a single LM for all utterances.

6. Acknowledgments

This research was partly funded by the Estonian Ministry of Education and Research target-financed research theme no. 0140007s12.

7. References

[1] B. Ballinger, C. Allauzen, A. Gruenstein, and J. Schalkwyk, "On-demand language model interpolation for mobile speech input," in *Interspeech 2010*, Chiba, Japan, 2010.

[2] T. Niesler and D. Willett, "Unsupervised language model adaptation for lecture speech transcription," in *ICSLP 2002*, Denver, Colorado, 2000.

[3] Y.-C. Tam and P. Vozila, "Unsupervised latent speaker language modeling." in *Interspeech 2011*, Florence, Italy, 2011.

[4] T. Alumäe and M. Kurimo, "Efficient estimation of maximum entropy language models with N-gram features: an SRILM extension," in *Interspeech 2010*, Chiba, Japan, 2010.

[5] T. Alumäe and K. Kaljurand, "Open and extendable speech recognition application architecture for mobile environments," in *SLTU 2012*, Cape Town, South Africa, 2012.

[6] D. Klakow, "Selecting articles from the language model training corpus," in *ICASSP 2000*, vol. 3, Istanbul, Turkey, 2000.

[7] C. Chelba and A. Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot," *Computer Speech and Language*, vol. 20, no. 4, 2006.

[8] H.-J. Kaalep and T. Vaino, "Complete morphological analysis in the linguist's toolbox," in *Congressus Nonus Internationalis Fenno-Ugristarum Pars V*, Tartu, Estonia, 2001.

[9] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney, "The RWTH Aachen University open source speech recognition system," in *Interspeech 2009*, Brighton, U.K., 2009.

[10] R. Kneser, J. Peters, and D. Klakow, "Language model adaptation using dynamic marginals," in *Proceedings of Eurospeech*, vol. 4, Rhodes, Greece, 1997.

[11] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of ICSLP*, vol. 2, Denver, USA, 2002.