Efficient Language Model Adaptation for Automatic Speech Recognition of Spoken Translations

Joris Pelemans[1], *Tom Vanallemeersch*[2], *Kris Demuynck*[3], *Hugo Van hamme*[1], *Patrick Wambacq*[1]

[1]ESAT, KU Leuven, Belgium
[2]Centre for Computational Linguistics, KU Leuven, Belgium
[3]ELIS, Ghent University, Belgium

{joris.pelemans,hugo.vanhamme,patrick.wambacq}@esat.kuleuven.be
tallem@ccl.kuleuven.be
kris.demuynck@elis.ugent.be

10.21437/Interspeech.2015-497

Abstract

Direct integration of translation model (TM) probabilities into a language model (LM) with the purpose of improving automatic speech recognition (ASR) of spoken translations typically requires a number of complex operations for each sentence. Many if not all of the LM probabilities need to be updated, the model needs to be renormalized and the ASR system needs to load a new, updated LM for each sentence. In computer-aided translation environments the time loss induced by these complex operations seriously reduces the potential of ASR as an efficient input method.

In this paper we present a novel LM adaptation technique that drastically reduces the complexity of each of these operations. The technique consists of LM probability updates using exponential weights based on TM probabilities for each sentence and does not enforce probability renormalization. Instead of storing each resulting language model in its entirety, we only store the update weights which also reduces disk storage and loading time during ASR. Experiments on Dutch read speech translated from English show that both disk storage and recognition time drop dramatically compared to a baseline system that employs a more conventional way of updating the LM.

Index Terms: speech recognition, language models, computer-aided translation, machine translation, efficient adaptation

1. Introduction

Although traditionally computer-aided translation (CAT) is performed with keyboard and mouse, a recent study [1] has shown that in the context of machine translation (MT) the use of automatic speech recognition (ASR) as an input method may constitute a significant speed-up, even with a non-perfect speech transcription that needs additional correction. Furthermore, it has been established that by using the translation model (TM) and more specifically the translation probabilities of the words and/or word groups (phrases) of the source language text, the speech recognition of spoken translations can be improved [2, 3, 4, 5, 6].

There are several different scenarios that can be applied to combining models for decoding the optimal transcription of a spoken translation where each implies different assumptions about system implementation and constraints on computational complexity. All of them are based on a Bayesian extension of the ASR maximum likelihood formula, first proposed by [2]: given a source language text $F = f_1...f_J$ and an acoustic signal $X = x_1...x_T$, which is the spoken version of a target language text $E = e_1...e_I$, the optimal transcription $\hat{E}$ is decoded as follows:

$$\begin{aligned}\hat{E} &= \underset{E}{\operatorname{argmax}}\, P(E|X,F) = \underset{E}{\operatorname{argmax}}\, P(X,F|E)P(E) \\ &= \underset{E}{\operatorname{argmax}}\, P(X|E)P(F|E)P(E) \end{aligned} \quad (1)$$

where the conditional independence assumption of X and F given E is considered to be reasonable and allows a decomposition into three knowledge sources: the acoustic model $P(X|E)$, the translation model $P(F|E)$ and the language model $P(E)$.

One scenario in which this extension can be used is to rescore hypotheses generated by the ASR system, using the TM probabilities as part of a multi-pass approach. This can be done either by re-ranking ASR n-best lists [7, 8, 3] or by rescoring word lattices [4, 5]. One of the main issues with multi-pass approaches however, is that the recognizer does not have access to MT information during the first pass. The recognizer might have already pruned out several interesting hypotheses that no rescoring can ever recover. Another issue is that the output of the recognizer has to be stored and that the second pass can only start when the first pass has finished, which takes up valuable space and time. In human-computer interaction, response times and storage should be minimized to reduce the overhead associated with rescoring.

Only very recently did the multi-pass scenario make room for a new scenario which we believe will be the new paradigm in MT-ASR integration. [6] shows that, for each source language sentence, the language model of the ASR system can be updated using information derived from the text of that individual sentence. This approach is claimed to yield a decrease in computational complexity by efficiently pruning the LM as well as a significant reduction in word error rate (WER).

In this paper we show that the efficiency of updating a LM according to the techniques described in [6] can still be greatly improved. We propose a method that allows for LM adaptation without renormalization, but with limited extra storage and fewer and on-the-fly updates.

In the remainder of the paper we discuss related work (Section 2), describe our method of integrating translation and language models (Section 3), and validate this method experimentally (Section 4). We end with conclusions and future work in Section 5.

This research is funded by the Flemish government agency IWT (project 130041, SCATE).

2. Related Work

While there has been a lot of work on rescoring the output of the speech recognizer with a translation model, our work is more comparable to the work of [6]. Instead of applying a multi-pass approach, the authors propose a direct integration of the translation model probabilities into the ASR language model. They do this by approximating $P(F|E)$ in Eq. 1 at the sentence level, only taking into account lexical translation probabilities. That is, each word e_i in the target language sentence E corresponds to exactly one word f_j in the source language sentence F:

$$P(F|E) \approx \prod_{i=1}^{I} \max_{j:f_j \in F} P(f_j|e_i) \tag{2}$$

The multiplication of $P(F|E)$ and $P(E)$ can then be considered as the new language model $P'(E)$ with which speech is decoded.

If we consider an n-gram to be a tuple (h, e) consisting of a history h and a target word e, then adapting the n-gram probability $P(e|h)$ to $P'(e|h)$ boils down to:

$$P'(e|h) = \frac{P(e|h) \max_{j:f_j \in F} P(f_j|e)}{norm(h)} \tag{3}$$

The normalization factor $norm(h)$ is obtained as follows:

$$norm(h) = \frac{\sum_{e':(h,e') \in \mathcal{T}} P(e'|h) \max_{j:f_j \in F} P(f_j|e')}{\sum_{e':(h,e') \in \mathcal{T}} P(e'|h)} \tag{4}$$

where $\mathcal{T}$ corresponds to the training data used to train the original language model probabilities $P(e|h)$.

Once all the relevant probabilities have been updated, the back-off weights $bow(h)$ are also renormalized to obtain a true probability distribution $P'(e|h)$:

$$bow(h) = \frac{1 - \sum_{e':(h,e') \in \mathcal{T}} P'(e'|h)}{1 - \sum_{e'':(h,e'') \notin \mathcal{T}} P'(e''|h)} \tag{5}$$

In addition to these updates, the authors also allow out-of-vocabulary (OOV) named entities to pass through and be included untranslated into the target language LM and the ASR lexicon. The named entities are assigned a unigram probability based on the length of the target sentence.

The authors claim that their suggested approach is efficient, because only the relevant n-grams (h, e), i.e. the n-grams for which the TM contains a probability $P(f|e)$, need to be updated. It is not clear however what happens to the irrelevant n-grams, i.e. the ones for which $P(f|e)$ is absent from the TM. Not updating these n-grams corresponds to multiplying the original probability $P(e|h)$ by 1 which is always larger than or equal to $\max_{j:f_j \in F} P(f_j|e)$, leading to relatively larger probabilities for irrelevant n-grams. It would be more logical to decrease the probabilities of the irrelevant n-grams e.g. by multiplying with a small value $\epsilon \ll 1$, but then each n-gram in the model would need updating, which would have a negative impact on efficiency.

Even if we assume that the authors apply a smart technique to solve this issue, their approach still requires computing the normalization factor and back-off weight for each history h. Moreover, since a new LM is created for each sentence, the approach requires a lot of storage and the speech recognizer needs to switch language models every time. In the next Section, we will describe an alternative to this approach that overcomes all of these issues.

3. Efficient Integration of Translation and Language Models

3.1. Doing Away with Normalization

A first and important observation for our proposed method is that LM normalization is not mandatory in the context of ASR decoding. In general, statistical language models assign probabilities to words and word sequences according to a certain probability distribution. If this were not the case and language model scores would be allowed to have any value, then a measure such as perplexity that compares LMs based on the score given to each observed word would lose its meaning entirely. It is therefore necessary to renormalize a language model whenever some of its probabilities are updated. In back-off n-gram LMs, renormalization not only consists of computing a normalization factor for each history, but also requires recomputing the back-off weight for that history, making it a relatively complex operation, as shown by Eqs. 4 and 5.

In ASR decoding however, a score is not attributed to a single word sequence, but rather to many competing word sequence hypotheses. Therefore, there is no strict constraint that the score that a LM attributes to each hypothesis should obey a true probability distribution, as long as a more likely hypothesis receives a higher score. Moreover, competing word sequence hypotheses E are not simply given a score according to their acoustic likelihood $P(X|E)$ and language model prior $P(E)$, but typically also include a LM scaling factor a and word insertion cost c:

$$score(E) = P(X|E)P(E)^a c^I \tag{6}$$

where I denotes the length of word sequence E. The LM scaling factor compensates for the acoustic model underestimation due to the overlarge dimensionality that acoustic features typically exhibit compared to the true dimensionality of speech [9], whereas the word insertion cost is used to overcome the LM bias towards less, hence longer words. The inclusion of these two parameters makes the final score that is attributed to a word sequence E unnormalized, even if $P(X|E)$ and $P(E)$ are. Note however that, although this means that the LM score does not have to be normalized, it is still important that the three factors in Eq. 6 operate on a similar scale if we want them to have a comparable impact on the final score.

3.2. Probability Inflation

As was explained in Section 2, updating LM probabilities by multiplying with TM probabilities, requires updating all n-gram probabilities $P(e|h)$, even those for which $P(f|e)$ is absent from the TM. This is due to the deflating nature of the update: if we instead were to multiply the relevant n-gram probabilities by values larger than 1, the irrelevant ones could remain untouched, thus significantly reducing the number of updates. We therefore propose an alternative update rule that essentially inflates rather than deflates the n-gram probabilities:

$$score_{LM}(e|h) = P(e|h) \max_{j:f_j \in F} g(f_j, e) \tag{7}$$

where g is a weighting function that maps TM probabilities $P(f|e)$ onto values larger than 1. To have some control over the shape and maximum of this function, yet minimize the introduction of new parameters, we propose the following exponential function:

$$g(f, e) = 1 + \alpha\beta^{1-P(f|e)} \tag{8}$$

where $\alpha \in \mathbb{R}_0^+$ controls the maximum value of the update weight and $\beta \in]0,1[$ determines the relative weight that is given to $P(f|e)$: a smaller value of β will give a relatively higher weight to high probabilities than to low probabilities. It is important that these parameters are carefully optimized, because if we inflate the LM probabilities too much we run the risk of overshadowing the acoustic score or the word insertion cost and ending up with acoustically implausible hypotheses or hypotheses consisting only of short words.

3.3. Applying Inflation Weights

In [6] a new LM is created for each sentence, which means that for each audio fragment the new LM has to be stored and loaded. For tasks that involve a lot of sentences and large LMs, this puts a serious load on the system that runs the software to assist humans in translation and may result in a translator being less rather than more productive.

To prevent such an unwanted scenario, we do not store each updated LM in its entirety, but instead store only the inflation weights. This seriously reduces the storage cost and it allows the speech recognizer to limit memory loading to a small set of inflation weights for each sentence.

4. Experiments

4.1. Task and Setup

The ASR experiments were performed on a test corpus of audio fragments from the Flemish part of the Corpus Spoken Dutch (CGN) [10], component o. The chosen fragments correspond to 167 Dutch utterances that are read translations from English books.

Using a vocabulary of 100k words, an initial 3-gram LM with modified Kneser-Ney smoothing [11, 12] was trained by running the SRILM toolkit [13] on a collection of normalized newspaper texts from the Flemish digital press database Mediargus which contains 1104M word instances (tokens) and 5M unique words (types). The vocabulary was converted into a phonemic lexicon using an updated version of [14] and integrated into the recognizer described in [15], which was built with our in-house speech recognition system SPRAAK [16].

The TM was created by applying the GIZA++ toolkit [17] to a set of 1M English-Dutch parallel sentence pairs extracted from the Europarl corpus [18], which contains the written version of speeches of members of the European Parliament. GIZA++ adopts an EM approach to learning lexical probabilities $P(f|e)$ and $P(e|f)$ from a parallel corpus. The approach initializes the lexical probabilities using a uniform translation distribution for words. Based on this initialization, the probabilities of possible word alignments of sentence pairs are calculated. The new probabilities then allow to recalculate the lexical probabilities across the set of sentence pairs. This process continues until convergence.

To build the updated LMs, we excluded all updates for source language words shorter than 4 letters, as we found that these tend to be unreliable. Words that have at least 2 letters and start with a capital letter are considered named entities and are never excluded. Named entities that are not in the TM, but are in the ASR lexicon, were given a maximal $P(f|e) = 1$; no new words were added to the lexicon. The inflation weights were generated with optimal values of $\alpha = 9$ and $\beta = 0.005$. Further optimization of the LM scaling factor a and word insertion cost c proved to be unnecessary.

As far as we know there is no available implementation for the normalized models presented in Section 2, so we had to revert to own implementation for the LM updates according to Eqs. 3 and 4. We implemented a model that uses a small, optimized value of $\epsilon = 0.001$ for the irrelevant n-gram probabilities, as discussed in Section 2. We also optimized the LM scaling factor a and word insertion cost c, which in contrast to the unnormalized model lead to further improvement. Renormalizing the back-off weights was done using the SRILM toolkit.

4.2. Disk Storage

Using SPRAAK, the initial 3-gram LM was converted into a compressed binary format reducing its size from 2.9GB to ca. 500MB. Since we do not add any words or n-grams to the model during the update, the normalized models are the same size for each sentence. Our test set contains 167 sentences, yielding a total disk storage of 484GB for the uncompressed models or 84GB after compression. For LMs where $n = 4$ or $n = 5$, this goes up to compressed models of 1GB and 1.5GB, respectively.

By contrast, the size of the file containing the update weights for a single sentence never exceeds 250KB, yielding a total disk storage of 42MB. Moreover, as the update weights do not depend on the size of the original n-gram LM, the disk storage does not increase for higher order models. That means that for 5-gram LMs, the disk storage is reduced to 2.8% compared to a normalized model (42MB vs 1.5GB).

4.3. Execution Speed

Both the normalized and unnormalized models must undergo several steps before they can be applied during ASR. For the normalized model we need to compute and normalize the updated probabilities, normalize the back-off weights, and convert the resulting language model to the compressed, binary format that we mentioned in Section 4.2. For the unnormalized model, we only need to compute the update weights. Once the models are created, the ASR system needs to switch LMs for every sentence. In Table 1 we give an overview of the time it takes to perform each of these tasks for 3-gram LMs, where the reported times were acquired by averaging the times of hundreds of iterations for each step on a dedicated machine with an Intel Core i5-2400 processor, using only a single core. To be fair, we do not report individual times for both normalization steps. An efficient implementation would combine the two steps, requiring only a single pass through the data as well as reducing the I/O time significantly. We therefore report a total normalization time that is limited to the time needed by the SRILM toolkit to recompute the back-off weights. We illustrate both the execution time per sentence and the total time to reflect the fact that some of the tasks need to be performed multiple times whereas others only happen once. Note however that for a real-time application, where a translator typically operates sentence by sentence, not all of the resources have to be available at the same time. As such, the most correct way to compare the two models is by looking at the execution time per sentence.

When considering the time needed for LM preprocessing, i.e. creating and converting a new LM to binary format, Table 1 shows that our unnormalized model is almost 3m faster than the normalized model. This is due to the fact that normalizing a LM is a lot more complex than computing update weights. Moreover, in the case of the unnormalized model, the conversion only concerns the initial, unadapted model, which can be done offline and effectively reduces its online LM preprocessing time to 0.2s. The normalized model on the other hand needs to normalize and convert a LM for each of the 167 sentences. With

	Normalized LM		Unnormalized LM	
	per sentence	**total**	**per sentence**	**total**
normalize LM probabilities	2m46s	7h42m2s	-	-
compute update weights	-	-	0.2s	34s
convert LM to binary format	1m42s	4h43m54s	1m42s	1m42s
total LM preprocessing	4m28s	12h25m56s	1m42.2s	2m16s
load acoustic model and lexicon	0.4s	0.4s	0.4s	0.4s
load LM	5s	13m55s	5s	5s
load update weights	-	-	0.01s	1.7s
total ASR	5.4s	13m55.4s	5.41s	7.1s
total	4m33.4s	12h39m51.4s	1m47.6s	2m23.1s

Table 1: Execution times for the different steps involved in recognizing 167 spoken, translated sentences, using a normalized vs. unnormalized 3-gram LM. Times were measured on a dedicated machine with an Intel Core i5-2400 processor, using a single core.

an execution time of 4m28s per sentence, it is clear that the normalized model can only be used in a scenario where the source language text is available beforehand. Note also that, analogous to Section 4.2, things get even worse when we consider higher order LMs. This is not the case for the unnormalized model: the computation of the update weights is independent of the order of the LM, making it ideally suited for a real-time application.

For the ASR there is a dismissible 10ms difference per sentence due to the loading of the update weights. Notice that, because of the conversion to the SPRAAK binary format, loading a LM can be done in only 5s. Again, with the unnormalized approach this only needs to be done once for the initial LM, whereas the normalized approach requires loading a new LM for every sentence. Nevertheless, with an execution time of 5s per sentence, the normalized model can also be used in a real-time application, provided that the LM is loaded while the translator is working on a previous sentence and that the LM preprocessing has been done offline.

We are aware that execution speeds depend heavily on implementation and although we have attempted to use efficient implementations for each task, the reported times can differ significantly with different hardware and/or software. Nevertheless we are confident that our unnormalized model is vastly more efficient, because several tasks have been made superfluous.

Finally, it is worth noting that the overhead of our technique compared to a setup that does not perform LM adaptation is limited to the computation and loading of the update weights. With a combined execution time of 210ms per sentence, we can safely say that the impact on efficiency is negligible.

4.4. Recognition Accuracy

Although the focus of this paper is on efficiency and not on accuracy, it is crucial to verify that our proposed model does not come at the cost of a significantly increased WER. Table 2 shows that the weighted updates of our model reduce the WER by more than 5% absolute and 20% relative, compared to the initial LM that does not have any MT information at its disposal. The proposed model also outperforms our implementation of the normalized model, but we would like to stress that this implementation does not correspond exactly to the one in [6]. The modest 5% WER reduction by our implementation compared to the 30% WER reduction that was reported in [6] can be partly explained by the different language pairs (English-Dutch vs French-English) and test sets. However, we suspect that the main reason is that we did not add any new, untranslated named entities to the ASR lexicon. From the analysis of our results, we observed that this could have a substantial impact on the WER.

	WER	Reduction
Initial LM (no TM updates)	25.96	-
Normalized LM	24.61	5.2%
Unnormalized LM	20.68	20.3%

Table 2: WERs (in %) and relative reductions of a normalized and unnormalized 3-gram LM, compared to an initial model that does not use TM probabilities. The models are evaluated on 167 utterances from the Dutch CGN corpus (component o), corresponding to translations from English.

We can therefore not claim that our proposed technique is more accurate than [6], but instead conclude that it is competitive and more efficient.

5. Conclusions And Future Work

We have presented an efficient language model adaptation technique for automatic speech recognition of spoken translations. The technique consists of n-gram probability inflation using exponential weights based on translation model probabilities which reduces the number of updates. It does not enforce probability renormalization and reduces data storage and memory load by storing only the update weights.

We have experimentally validated that this technique is capable of reducing word error rates by 5% absolute and 20% relative on spoken Dutch translations from English, while having little to no negative effect on recognition time. Compared to a normalized model, the model takes up only 2.8% of disk space for 5-grams and dramatically reduces the execution time.

Future work will mostly focus on improving recognition accuracy. Because the efficiency of the unnormalized models is only dependent on computing the weights based on TM probabilities, which does not take up a significant amount of time, our method allows us to make use of more complex translation models with a reduced risk of slowing down the recognition.

In its current setup the technique uses a relatively simple translation model with only lexical probabilities. One extension could be to use phrase-based translations models [19], which are the current state of the art in machine translation. Such models link spans of words (phrases) p_i in sentence pairs using a word alignment derived with GIZA++ probabilities. From phrase pair counts across the parallel corpus, the models create probabilities $P(p_1|p_2)$ and $P(p_2|p_1)$.

Other extensions include tailored models for Dutch compounds and ASR lexicon expansion with out-of-vocabulary named entities.

6. References

[1] B. Dragsted, I. Mees, and I. Gorm Hansen, "Speaking your Translation : Students' First Encounter with Speech Recognition Technology," *Translation & Interpreting*, vol. 3, no. 1, pp. 10–43, 2011.

[2] P. Brown, S. Chen, S. Della Pietra, V. Della Pietra, S. Kehler, and R. Mercer, "Automatic Speech Recognition in Machine Aided Translation," in *Computer Speech and Language*, vol. 8, 1994, pp. 177–187.

[3] M. Paulik, C. Fügen, S. Stüker, T. Schultz, T. Schaaf, and A. Waibel, "Document Driven Machine Translation Enhanced ASR," in *Proc. Interspeech*, 2005, pp. 2261–2264.

[4] S. Khadivi and H. Ney, "Integration of Automatic Speech Recognition and Machine Translation in Computer-assisted translation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 8, pp. 1551–1564, 2008.

[5] A. M. Reddy and R. C. Rose, "Integration of Statistical Models for Dictation of Document Translations in a Machine-Aided Human Translation Task," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 8, pp. 2015–2027, 2010.

[6] L. Rodríguez, A. M. Reddy, and R. C. Rose, "Efficient Integration of Translation and Speech Models in Dictation Based Machine Aided Human Translation," in *Proc. ICASSP*, 2012, pp. 4949–4952.

[7] J. Brousseau, G. Foster, P. Isabelle, R. Kuhn, Y. Normandin, and P. Plamondon, "French Speech Recognition in an Automatic Dictation System for Translators: the TransTalk Project," in *Proc. Eurospeech*, 1995, pp. 193–196.

[8] S. Khadivi, A. Zolnay, and H. Ney, "Automatic Text Dictation in Computer-Assisted Translation," in *Proc. Interspeech*, 2005, pp. 2265–2268.

[9] M. Alder, R. Togneri, and Y. Attikiouzel, "Dimension of the Speech Space," *IEE Proc. Communications, Speech and Vision*, vol. 138, no. 3, pp. 207–214.

[10] N. Oostdijk, "The Spoken Dutch Corpus. Overview and first evaluation," in *Proc. LREC*, 2000.

[11] R. Kneser and H. Ney, "Improved Backing-off for M-gram Language Modeling," in *Proc. ICASSP*, vol. I, pp. 181–184.

[12] S. F. Chen and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling," 1998.

[13] A. Stolcke, "SRILM - An Extensible Language Modeling Toolkit," in *Proc. ICSLP*, 2002, pp. 257–286.

[14] K. Demuynck, T. Laureys, and S. Gillis, "Automatic Generation of Phonetic Transcriptions for Large Speech Corpora," in *Proc. ICSLP*, pp. 333–336.

[15] K. Demuynck, A. Puurula, D. Van Compernolle, and P. Wambacq, "The ESAT 2008 System for N-Best Dutch Speech Recognition Benchmark," in *Proc. ASRU*, 2009, pp. 339–343.

[16] K. Demuynck, "Extracting, Modelling and Combining Information in Speech Recognition," Ph.D. dissertation, K.U.Leuven ESAT, 2001.

[17] F. J. Och and H. Ney, "A Systematic Comparison of Various Statistical Alignment Models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[18] P. Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation," in *Proc. Machine Translation Summit*, 2005, pp. 79–86.

[19] P. Koehn, F. J. Och, and D. Marcu, "Statistical Phrase-Based Translation," in *Proc. HLT-NAACL*, 2003, pp. 48–54.