Towards Separating Trigram-Generated Sentences and Natural Sentences with Support Vector Machines

Jerry Xiaojin Zhu ZHUXJ@CS.CMU.EDU
CALD, CMU

Abstract

We propose research on a novel method to improve language modeling for automatic speech recognition. The problem of language modeling is to assign probabilities to any sentence, such that well-formed, meaningful sentences get high probabilities, while ungrammatical, nonsense ones get low probabilities. Traditional trigram language models are unable to model sentence level syntactic and semantic correlation, and thus often assign high probabilities to ill-formed sentences which leads to recognition error. This proposal addresses such problems by training a support vector machine (SVM) classifier to learn a decision boundary between well-formed and ill-formed sentences. The SVM feature space is suitably chosen to express sentence level correlation. With the trained SVM, we classify candidate sentences as either well-formed or ill-formed. If a sentence is classified as ill-formed, we can then decrease its language model probability to prevent it from being the speech recognition result. This paper focuses on the classification problem, especially feature representation for sentences. We show that it is very hard to derive good features that captures sentence syntactic structures with part-of-speech tagging. We also show empirically that the margin of SVM classification can be used to derive posterior probabilities of classes.

1. Introduction

Automatic speech recognition (ASR) systems offer an attractive interface between human and machines. State of the art ASR systems have begun to show practical impact in situations where the user is physically challenged to use a keyboard or see a display; in hands-free eye-free applications such as driving assistant; or in telephone based automatic information services where keyboard input is limited. However these applications are usually very limited in acceptable vocabulary, syntax, or even speaking style. We are still unable to talk to a computer as fluent as HAL-9000 in "2001: A Space Odyssey". Computers simply cannot recognize large vocabulary spontaneous speech, e.g. conversations in our everyday life, with enough accuracy, let alone the subsequent much harder natural language understanding, reasoning, and responding problems.

One reason for this lack of accuracy can be ascribed to the deficiency of a major module, the language model, within ASR systems. Intuitively, a language model contains all the knowledge the system knows about language usage. Ideally the language model should assign high probability scores to all well-formed sentences (fluent, meaningful ones), and low probability scores to ill-formed sentences (ungrammatic, nonsense ones). An ASR system uses these probability scores to weigh acoustically similar candidate sentences in order to find the best decoding for a human utterance. For example, two candidate sentences for the utterance "It's hard to recognize speech." might be the correct sentence itself, and "It's hard to wreck a nice beach.". A good language model should give a high probability score to the first candidate sentence, and a much lower probability score to the second one. But to pack the complete language usage knowledge in a computational model is very hard. ASR researchers have to enormously simplify the language model to make it practically viable. Because of this simplification, language models are more lax than we would like it to be. It still gives high probability scores to well-formed sentences, but it might give higher scores to some ill-formed sentences, which leads to recognition errors. Creating better language models is therefore an important problem for ASR.

The long term goal of this research is to improve language modeling to reduce ASR recognition errors. As a first step, the focus of this paper is to classify

well-formed sentences from ill-formed sentences, in a way that we can estimate the posterior probabilities. Specifically, we pose it as a classic binary classification problem. The well-formed and ill-formed sentences as two distinct classes. We collect samples from each class, and then train a support vector machine (SVM) to find a good decision boundary between the two classes. With the SVM we can compute the classification and margin (confidence) for any unseen new sentences.

2. Related Work

Over the last decade, there has been extensive work on language modeling. For example, (Katz, 1987) and (Jelinek & Mercer, 1980) introduced the trigram independence assumption and ways to estimate probabilities from observed frequencies, (Witten & Bell, 1991), (Kneser & Ney, 1995) and (Rosenfeld, 1996) improved the estimation with various statistical methods. A detailed survey of recent literature on language modeling can be found in (Rosenfeld, 2000). Our approach differs from other work in the field in two important technical aspects.

1. We consider every sentence as an integrated unit. Most language models operate on subsentence levels like trigrams (word sequences of length 3) for pragmatic reasons. By doing so they lost the ability to model phenomena that range over a whole sentence, such as some syntactic and semantic constraints. Being able to model sentence level phenomena is essential to a better language model.
2. We find the deficiency of the existing language model with a machine learning approach.

Recently Rosenfeld (Rosenfeld, 1997) introduced a whole sentence maximum entropy language model which is capable of model sentence level phenomena. Following the work, Chen and Rosenfeld (Chen & Rosenfeld, 1999) suggested a method to systematically explore the difference of any computable aspects (called features) between an existing language model and the empirical distribution of training sentences, and exploits the differences to improve the language model. These features can include sentence level features such as sentence length, syntactic structure of the sentence, etc. Zhu and Rosenfeld (Zhu et al., 1999) applied the method to a set of sentence level linguistic features and showed some improvement in language model likelihood. However their work relied on human judgement to specify a family of computable features, on which the differences are sought. Human judgement, although usually judicious, is often highly focused on a small area of the problem space. Our proposed approach instead would use a machine learning method to automatically find a (possibly) complex feature that exhibits the difference most. We believe this will be a good complement to the previous work.

Cai et al. (Cai et al., 2000) also started from a set of manually picked features which are believed to capture the semantic coherence of a sentence. But instead of using the features as they are for language modeling, they fit a generalized additive model with smoothing splines. This is essentially a step toward letting the data decide how to best use the features. But their initial set of only six features are too limited to allow enough freedom in generalized additive model fitting, while we are going to start with a much broader choices.

Stolcke et al. (Stolcke et al., 2000) trained an 'anti' language model, not from well-formed sentences but from decoded candidate sentences. Since most candidate sentences contain errors originated from the language model, the anti-language model is specialized in finding the deficiency of the original language model. This is an example of applying machine learning approaches. However unlike our approach, their model cannot capture sentence level phenomena.

Perhaps the most similar work is by Eneva et al. (Eneva et al., 2000). They trained a classifier between well-formed and ill-formed sentences using boosted decision stumps, with many different types of semantic features like content word pair likelihood ratio, word repetition etc. We plan to use a different classifier, namely SVM which has been proven to work well in text categorization tasks, and explore syntactic features (and the combination of both). We will also estimate the posterior probabilities from SVM.

3. Approach

3.1 Background

In ASR systems, the final processing step is 'N-best list rescoring' ((Jelinek, 1997), pp 86). Before this step, the recognizer has generated N candidate sentences which are acoustically similar to the input utterance (but may not make sense syntactically or semantically). At this step the recognizer's task is to pick the best sentence in both the acoustics and language sense. The N candidate sentences are called an N-best list, see Figure 1. Formally, each candidate sentence s has an acoustic score $P(u|s)$ which indicates its acoustic proximity to the input utterance u. N-best list rescoring is to find the best sentence s_u^* that

maximize the *a posteriori* probability

$$s_u^* = arg\max_s P(s|u) = arg\max_s P(u|s)P(s)$$

Therefore we need a *language model* score $P(s)$ to multiply to the acoustic score and re-rank the N-best list to get the best sentence. $P(s)$ ideally should assign high scores to all well-formed sentences and low scores to all ill-formed sentences. This is very hard, because we do not yet know how to put human intuition of good and bad sentences in a computational model. In practice, the so called *trigram language model* is widely used because of its simplicity. In trigram language models, the probability of a sentence is the product of the conditional probabilities of its words, conditioned on previous two words respectively:

$$P(s) = P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$$

and each conditional probability is estimated by counting a large text collection, or corpus:

$$P(w_i|w_{i-2}, w_{i-1}) \approx \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} \quad (1)$$

Albeit popular, trigram language models have severe deficiencies, notably its inability to model long range, sentence level phenomena. This problem roots in their short range (two previous words) conditional independence assumption. Intuitively if we chop a sentence into several large chunks and scramble the order of the chunks, the resultant sentence will most likely have comparable trigram language model probability as the original sentence, but will be nonsense to a human reader. In our prior work, we have tried to incorporate sentence level linguistic and semantic information into language models and showed some improvement (Zhu et al., 1999) (Cai et al., 2000). In this work, we will extend our research with machine learning methods to automatically improve upon trigram language models [1]. As a first step towards this end, we focus on finding good features that reveal the deficiency of trigram language models.

3.2 Technical Approach

Our approach takes a whole sentence as a unit and investigates the difference between well-formed and ill-formed sentences. To do so we need samples from both classes. The probability distribution of well-formed sentences is unknown. But we can easily acquire samples (here and in the sequel, a sample is a complete sentence) from that distribution. Indeed, every sentence published in books, newspapers, on the web, or spoken on radio and TV broadcast, is a sample from the distribution. In the following we call well-formed sentences *natural sentences* to indicate their origin. In this experiment we will use as natural sentences a 1 million-word broadcast news transcription collected from ABC, CBS, CNN, NPR between 1992 and 1996.

For ill-formed sentences, we focus on a subset of them, namely those generated by a trigram language model with typical probabilities, since they are closely related to the candidate sentences in the N-best lists. In the following we call them *trigram-generated sentences*. It is easy to use the trigram language model as a generative model and generate sentences as follows: select the first word randomly according to word frequency; select next word randomly according to the trigram probability (1) and so on. We use a separate (but from the same source and same period) 100 million-word text collection to train the trigram language model. Figure 2 shows a few sentences generated in this fashion. They (especially the longer ones) are very distinct to a human reader.

We pose the problem of automatically finding the difference between natural and trigram-generated sentences as a binary classification problem. (Note in fact the two classes completely overlap since both is capable of generating any sentence. But their probability distributions are very different.) The samples mentioned above serves as positive and negative examples.

There are many methods for binary classification, and we choose support vector machines (SVM). A SVM finds a linear decision boundary that separates the positive and negative examples in an augmented feature space, such that the distance (the so called margin) from the decision boundary to the closest positive or negative example is kept as large as possible. This maximization of margin gives SVM good generalization performance. In addition, SVM has a computationally efficient way to augment the feature space via so called kernel functions. For details see e.g. (Burges, 1998) (Cristianini & Shawe-Taylor, 2000). We choose SVM because

- It is the state-of-the-art classification method in text categorization tasks (Joachims, 1998), which is closely related to our problem (though we note in our problem the order of words in a sentence is important, and we will take that into account; while in text categorization people often use the bag-of-word model which ignores order);
- Its kernel function gives us potential ways to ex-

[1] In theory our method can be used to improve any kind of language model.

$\log P(u\|s)$	candidate sentence s
-5522539	it *sites* class *eyes* quality of life as problems
-5556088	it *sites* class size quality of life *has* problems
-5556088	it cites class size quality of life *has* problems
-5622228	it *sites klas eyes* quality of life *has* problems
-5653812	it *sites* class size quality of life as problems
-5653812	it cites class size quality of life as problems
.....	*(many, many other candidate sentences)*

Figure 1. The N-best list for utterance "It cites class size quality of life as problems." Errors are highlighted

how many different people once if you're going to interrupt you right
i just want approval ratings are drawn to the rescue operations of the films that were going too fast
and we hope to move because of nafta culture has reached out to walk down the authority to stop the deity republican conventions with a star power
paranoia about each other for each of you
yes hello
believers say who needs a lawyer from a unk very impressed with such and such is not a time magazine that we just don't believe
he took over or is it by all human beings
danny rolling
and we can put on trial
expect the fiction within the white house and the drag racing school steered a costly and time is up to eight bucks or part of the abortion plank

Figure 2. Sample sentences generated from a trigram language model

press word interactions on a sentence level;

- It produces a single number (the margin) when classifying a new sentence. The number can easily be converted to a posterior probability or used in the subsequent discriminative training.

The performance of SVM is influenced by our design of two things: the feature representation of the sentences, and the kernel function. Traditionally people have used 'bag-of-words' feature presentation (McCallum & Nigam, 1998). But it might not give us the best performance since this representation ignores word order information completely, which intuitively is important in our domain.

We plan to investigate a variety of feature representations, including semantic features used by Eneva et al. (Eneva et al., 2000), and syntactic feature. To augment the feature space, we will investigate the effect of nonlinear kernels.

Once we have trained a SVM, we can classify a new sentence as either a natural sentence or a trigram-generated one. We also get the raw (unthresholded) output $f(s)$ of the SVM

$$f(s) = \sum_i y_i \alpha_i k(s_i, s) + b$$

which is related to the confidence or the margin. We will use $f(s)$ to estimate the posterior probabilities.

3.3 Exploring Features

Intuitively, trigram sentences in Figure 2 are 'bad' because of two reasons: first they don't following the correct English grammar, and second the meaning of words don't go with each other. This prompts us to look for features that represent the syntactic and semantic aspects of language.

We experiment with the following syntactic features:

- Bag-of-word

  This is the familiar feature representation used in document classification. A sentence is represented as a V dimensional vector, where V is the vocabulary size (typically more than 10,000). The value of a dimension can be binary, i.e. '1' if the corresponding word occurs in the sentence, '0' otherwise. The value can be the raw count of the word in the sentence as well. It can also be the empirical frequency, which is the raw count normalized by the length of the sentence.

  The bag-of-word representation enjoyed enormous success in document classification. However we

expect this representation perform poorly for our task for two reasons: first a sentence is much shorter than a document which leads to serious sparseness problem; second bag-of-word ignores word order completely, which is important for grammar. Therefore we include this representation here for comparison purpose only.

- Bag-of-POS

  A natural way to handle sparseness while looking for syntactic features is to back off from words to their part-of-speech (POS) tags. There are far less POS tags than the number of words. The system we used, the Brill tagger, employs 40 different POS tags. For example, it tags the trigram-generated sentence

  ```
  when the director of the many of your father
  ```

  into a POS sequence

  ```
  WRB DT NN IN DT NN IN PRP$ NN
  ```

  which, we hope, still retains enough syntactic information.

  A natural feature would then be bag-of-POS, which is exactly bag-of-word except the vocabulary consists of POS tags instead of words.

- Sequence-of-POS

  So far we ignored the order of words/POS, which we think are important for syntactic features. Therefore we consider all sequences of POS of length k. Each such sequence will be a feature, and therefore the dimension of the feature vector is 40^k. Inspired by (Lodhi et al., 2000) which uses letter sequences for document classification, we consider not only sequences consisting of adjacent POS tags, but also sequences that are far apart. For instance, let $k = 3$ and 'WRB IN DT' is a non-adjacent sequence for the example sentence above. The value of a sequence is weighted by λ^s, where $\lambda \in (0, 1)$ is a tuning parameter, and s is the span of the sequence in words. As a result the importance of a sequence decays exponentially as the span gets longer. In the above example the sequence 'WRB IN DT' has a weight λ^5, because the sequence spans the five words 'when the director of the'.

- Sequence-of-STOPOS

  One concern about the part-of-speech tagging is it might be too aggressive in blurring the syntactic function of words. One alternative is to keep some stop words (common words like 'the, of, a, and'), while reducing all other words to their part-of-speech tags. We use a stop word list of 70 most frequent words. For the example sentence above, the result is

  ```
  WRB the NN of the NN of PRP$ NN
  ```

  This hybrid tagging system of stop words and POS has 110 symbols, we call it STOPOS. As above, we seek all sequences of length k and use them as features.

- Statistics of STOPOS n-gram counts

  Now we consider a very different family of features. Let the STOPOS tagging of a k word sentence S be $t_1, t_2, \cdots, t_k$. Consider all the STOPOS n-grams (an n-gram is a sequence of n consecutive tags) in this sentence: there will be $k-n+1$ n-grams: from $t_1, \cdots, t_n$ to $t_{k-n+1}, \cdots, t_k$.

  For each STOPOS n-gram, we find its count c in a separate 100 million word natural sentence corpus. That is, c is the number of times the n-gram appear in the separate corpus (note c is not the count in the sentence S). The corpus is used solely for getting these counts, and not used anywhere else.

  Thus the sentence S can now be represented as n counts $c_1, \cdots, c_n$. Our intuition is that trigram-generated sentences tend to include 'unfamiliar' n-grams, which correspond to lower c counts. We derive several features from counts $c_1, \cdots, c_n$ for sentence S as follows:

  - The min, max, mean, median of counts $c_1, \cdots, c_n$.
  - The histogram of the counts. We use a histogram with 195 bins $h_1, \cdots, h_{195}$. The value of each bin is how many c_i's fall in that bin.
  - Likelihood ratio. Using yet another separate corpus of 2 million word trigram-generated and 2 million word natural sentences, we build two pooled histograms h^t and h^n, one for each class. We interpret them as two multinomial distributions over counts:

    $$\begin{aligned} P(S|natural) &= P(c_1, \cdots, c_n|natural) \\ &= \prod_{i=1}^{n} P(c_i|h^n) \end{aligned}$$

    and similarly $P(S|trigram - generated)$. Then the likelihood ratio is defined as

    $$\frac{P(S|trigram - generated)}{P(S|natural)}$$

feature	test set accuracy
bag-of-word	58.1%
bag-of-POS	55.8%
sequence-of-POS	58.4%
sequence-of-STOPOS	56.9%
STOPOS 4-gram stat.	-
semantic	77.1%
(4-gram stat.)+(semantic)	75.5%

Table 1. Comparison of various features

For semantic features, we follow the work in (Eneva et al., 2000) and use the same 70 features. Most of the features compute the strength of content word co-occurrence in a sentence; some detect content word repetition.

4. Experiment Results

Through out the experiment, we only include sentences that have at least 7 words, since shorter trigram-generated sentences are very similar to natural sentences. We use natural sentences from broadcast news transcript. A separate broadcast news transcript corpus is used to generate a typical trigram language model, from which we sample trigram-generated sentences. We use svm-light (Joachims, 1999) and mysvm (Ruping, 2000) to train SVM used in this experiment. Due to SVM convergence problems, we use a small training set. Our training data consists of 4,800 sentences, half natural and half trigram-generated. Our test data consists of 12,000 sentences.

We experimented with the features proposed above, and list the results in Table 1. The reported accuracy is the best among several parameter settings within the same feature family. Details are:

- bag-of-word: binary values, second order polynomial kernel $(x.y)^2$.
- bag-of-POS: binary values.
- sequence-of-POS: $k = 3$, any sequence's span is limited to be within 7 words, $\lambda = 0.8$.
- sequence-of-STOPOS: $k = 3$, any sequence's span is limited to be within 7 words, $\lambda = 0.5$.
- STOPOS 4-gram statistics: SVM training didn't converge on this feature set alone. The feature set includes normalized min, max, mean, median counts; normalized log likelihood ratio, the first 100 bins of the normalized histogram.

The semantic features worked comparably to those reported by (Eneva et al., 2000) (note we used far less training data, which may explain the few percentage off). Interestingly none of the syntactic features worked as expected. This hints that we are looking at possibly the wrong representation, i.e. part-of-speech.

With the features that did work, i.e. semantic features, we plot the distribution of SVM classification margin on the test set as shown in Figure 3. The blue curve is trigram-generated and the red one natural sentences. The x-axis is margin and the y-axis is frequency. We can fit some distribution on them as in (Platt, 1999), and get class posterior probabilities, which can be used later on to improve N-best list rescoring.

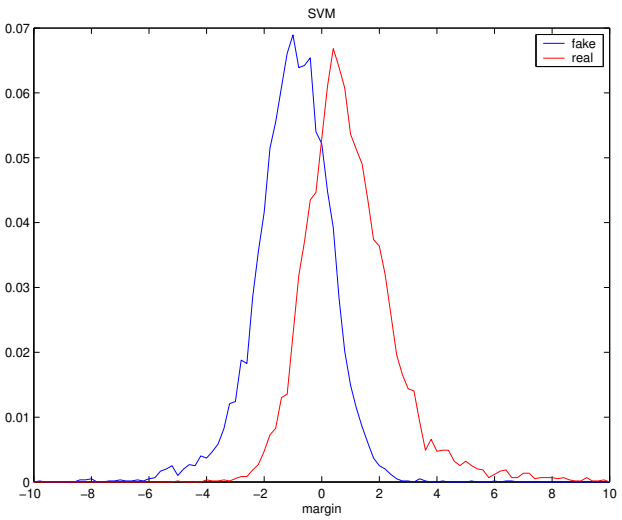

Figure 3. Distribution of SVM margin on test data

5. Discussions

As we have seen, the proposed syntactic features involving POS didn't work. A possible reason is that part-of-speech tagging doesn't really reveal the syntactic structure of sentences. To this end, we can proceed to parsing, which may provide more sophisticated syntactic information.

In our experiment, the convergence of SVM is a major problem. Our empirical experience seems to indicate that when the training set is large and noisy, convergence becomes slow. In the future we plan to experiment with logistic regression instead. The output of logistic regression is posterior probabilities by definition, without the need of another empirical estimation.

Acknowledgements

The author would like to thank Rose Hoberman for the semantic features, Benjamin Han for the enhanced Brill tagger, Thorsten Joachims for svm-light, Stefan Ruping for mySVM, Roni Rosenfeld and Sebastian Thrun for useful discussions, and all the anonymous reviewers for valuable feedbacks.

References

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 955–974.

Cai, C., Wasserman, L., & Rosenfeld, R. (2000). Exponential language models, logistic regression, and semantic coherence. *Proceedings of the NIST/DARPA Speech Transcription Workshop.*

Chen, S. F., & Rosenfeld, R. (1999). Efficient sampling and feature selection in whole sentence maximum entropy language models. *ICASSP-99.* Phoenix, Arizona.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press.

Eneva, E., Hoberman, R., & Lita, L. V. (2000). Semantic coherence. Course Project Report for 15781 Machine Learning Fall 2000, Carnegie Mellon University.

Jelinek, F. (1997). *Statistical methods for speech recognition.* Cambridge, Massachusetts: MIT Press.

Jelinek, F., & Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. *Proceedings of the Workshop on Pattern Recognition in Practice* (pp. 381–397). Amsterdam, The Netherlands: North-Holland.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the European Conference on Machine Learning.*

Joachims, T. (1999). Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning.* MIT-Press.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *35*, 400–401.

Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 181–184). Detroit, Michigan.

Lodhi, H., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2000). *Text classification using string kernels* (Technical Report NC-TR-00-079). NeuroCOLT.

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. *AAAI-98 Workshop on Learning for Text Categorization.*

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf and D. Schuurmans (Eds.), *Advances in large margin classifiers.* MIT Press.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, *10*, 187–228. longer version published as "Adaptive Statistical Language Modeling: A Maximum Entropy Approach," Ph.D. thesis, Computer Science Department, Carnegie Mellon University, TR CMU-CS-94-138, April 1994.

Rosenfeld, R. (1997). A whole sentence maximum entropy language model. *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding.*

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, *88.*

Ruping, S. (2000). mySVM. http://http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

Stolcke, A., Bratt, H., Butzberger, J., Franco, H., Gadde, V. R. R., Plauche, M., Richey, C., Shriberg, E., Sonmez, K., Weng, F., & Zheng, J. (2000). The SRI march 2000 Hub-5 conversational speech transcription system. *Proceedings of the NIST Speech Transcription Workshop.* College Park, MD.

Witten, I. H., & Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, *37*, 1085–1094.

Zhu, X., Chen, S. F., & Rosenfeld, R. (1999). Linguistic features for whole sentence maximum entropy language models. *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech).* Budapest, Hungary.