SEARCH RESULTS BASED N-BEST HYPOTHESIS RESCORING WITH MAXIMUM ENTROPY CLASSIFICATION

Fuchun Peng, Scott Roy, Ben Shahshahani, Françoise Beaufays

Google

ABSTRACT

We propose a simple yet effective method for improving speech recognition by reranking the N-best speech recognition hypotheses using search results. We model N-best reranking as a binary classification problem and select the hypothesis with the highest classification confidence. We use query-specific features extracted from the search results to encode domain knowledge and use it with a maximum entropy classifier to rescore the N-best list. We show that rescoring even only the top 2 hypotheses, we can obtain a significant **3%** absolute sentence accuracy (SACC) improvement over a strong baseline on production traffic from an entertainment domain.

Index Terms— N-best reranking, Voice search, Language modeling, Maximum entropy modeling

1. INTRODUCTION

Nowadays, smart devices such as smart phones and TV are becoming more and more popular as they provide people with convenient ways to find information related to entertainment. However, traditional keyboard typing is not an efficient way for input on such devices. Instead, voice input based on automatic speech recognition (ASR) is a more natural alternative [1, 2, 3]. Thus, the quality of an ASR system is critical to the success of such smart systems.

In ASR systems, the language model (LM) provides a likelihood for each recognized sentence. N-gram language models are by far the most common, as they provide excellent modeling power while being fast to train and easy to compile into finite state transducer (FST) decoders [4]. However, because of the limited language context they model, n-gram language models may output ungrammatical or implausible word sequences [5]. Another drawback of n-gram language models is that they cannot easily capture domain knowledge: at most they can be trained on domain-specific data, but they can't take into consideration query-specific information. For example, a language model that is broad enough to model naturally spoken queries may not be able to disambiguate between acoustically confusable sentences such as "I say" and "Ice Age". However, if it is known that the input is a query meant for an entertainment search system recognizing TV shows, music, apps, etc, it becomes more likely that "Ice Age" is what users wanted. There are many other examples where words are acoustically confusable, and a broad language model is not able to pick the right phrase, such as "Leap frog" versus "sleep frog", "Halo Reach" vs "halo wheat", "Pinterest" versus "interest", or "Siri" versus "series". However, we can accurately detect all of the errors if we send them as queries to a search backend and examine the search results.

Even when the right transcription is not the top recognition hypothesis, it is often in the N-best hypotheses list. In this study, we found that among all the recognition errors we found in our test sets, 30% contain the correct recognition result among the top 20 hypotheses. Furthermore, most of the correct results occur within top five hypotheses. Table 1 shows the error distribution at different N-best positions. Notably, if the correct recognition result is in the N-best list, 44% of the time it occurs in the second position.

Table 1: N-Best error distribution in different position

Rank position	Error percentage
2	44%
3	20%
4	13%
5	9%
6+	14%

These observations motivate us to rerank the speech N-best list using search results as features. A benefit of such an approach is that it is query specific, thus we can improve infrequent queries that are usually not handled well by a generic language model which naturally puts more modeling power on the head of the distribution. As we will see in our experiments, this approach significantly improves speech recognition accuracy, increasing the sentence-level accuracy by **3%** even when we only rerank the top 2 hypotheses.

This work is to our knowledge the first publication that reports successfully exploiting search results directly to improve speech recognition accuracy. We formulate the problem as a binary classification problem and model it with a maximum entropy classifier whose output is used for reranking hypotheses and for rejecting bad utterances.

2. RELATED WORK

There is a lot of research in N-best rescoring in the natural language processing (NLP) and speech domains [6, 7]. Most existing speech reranking approaches focus on reranking lattices with a large language model or with additional features [5, 8, 9, 10, 6, 11]. Reranking with additional features usually works in the scenarios where there is additional knowledge that is not available or too expensive to compute in the first pass, such as syntactic features [12, 13] or morphological features and N-best edit distance features [14]. In this work, we use search results and semantic parsing features.

Maximum entropy models for language modeling have been applied to various problems, such as correcting speech split and merge errors [15] or discriminative reranking [6]. In this work, we choose instead to define the problem as a binary classification problem, where the classifier output can be directly used for reranking or rejecting recognition results.

3. N-BEST RERANKING AS A CLASSIFICATION PROBLEM

In our application, misrecognizing part of a movie name or an app name results in a poor user experience. To emphasize this, we use whole sentence accuracy (SACC) as our recognition metric, as done in other papers such as [16].

With SACC as a metric, we can define the N-best reranking problem as a binary classification problem: a recognition hypothesis is a *positive* example that has all words correctly recognized or a *negative* example that has at least one misrecognition. We compute the probability that each hypothesis for a given input is correct, and we rank the hypotheses using this score. The hypothesis with the highest score is used as the final recognition result. Since the classification output is probabilistic, we can also use the score to reject an utterance, i.e., if the score of the best hypothesis is below some threshold, we can consider this utterance as irrelevant. In production systems, this happens quite often as users speak to other people, or click on the microphone but don't speak, so that the input contains only ambient noise. Rejecting such inputs provides a better user experience than returning random search results for them.

There are many classification solutions such as decision trees, support vector machine (SVM), and maximum entropy models. We choose maximum entropy model as it naturally models the problem we want to solve, and it has the flexibility of incorporating arbitrary features.

Figure 1 and 2 describe the training and testing procedures.

1. Step 1: Extract N-best hypotheses from the speech recognizer.
2. Step 2: For each hypothesis, extract speech recognition features, semantic features, and search result features.
3. Step 3: Label each hypothesis as *positive* or *negative* by comparing it to a human transcription of the input: if all words are correct, it is labeled as *positive*, else it is marked as *negative*.
4. Step 4: Train a maximum entropy classifier from the labeled training data, using the L-BFGS algorithm.

Fig. 1: Training procedure

1. Step 1: Extract N-best hypotheses from the speech recognizer.
2. Step 2: For each hypothesis, extract speech recognition features, semantic features, and search result features.
3. Step 3: Apply the maximum entropy classifier on each hypothesis and compute a classification score.
4. Step 4: Rerank the N-best hypotheses based on their classification scores. Pick the hypothesis with the highest score as the final system output.
5. Step 5: If the ranking score of the best hypothesis is too low, reject the utterance. This step is optional.

Fig. 2: Testing procedure

3.1. Maximum Entropy Classification

Conditional maximum entropy (ME) modeling is a framework that has been used in a wide array of NLP tasks [17, 18]. The principle of maximum entropy is to pick a distribution that has the maximum entropy subject to known constraints expressed as features. Let a real-valued function of the hypothesis h and the class c be a feature, $f_i(c, h)$. Maximum entropy allows us to restrict the model distribution to have the same expected value for this feature as its empirical value from the training data, H. Thus, we stipulate that the learned conditional distribution $P(c|h)$ must have the property:

$$\sum_{h \in H} f_i(c, h) = \sum_{h \in H} \sum_{c} P(c|h) f_i(c, h) \quad (1)$$

It is shown that the maximum entropy distribution has the following form [19]:

$$P(c|h) = \frac{e^{\sum_i \lambda_i f_i(c,h)}}{Z(h)} \quad (2)$$

where $Z(h)$ is a normalization factor,

$$Z(h) = \sum_{c} \sum_{i} \lambda_i f_i(c, h). \quad (3)$$

During ME training, the optimal weights λ_i corresponding to features $f_i(c, d)$ are learned so as to maximize the log-likelihood of the training data. The weights are learned via an improved iterative scaling algorithm, or faster algorithms such as conjugate gradient descent and L-BFGS [20]. We used L-BFGS. For further details on the training algorithm, we refer the reader to [20].

A great advantage of the maximum entropy model compared to other models is that it can take arbitrary constraints as features with no independence assumptions among the features. In our application, we use a maximum entropy model to combine arbitrary contextual features from search results, syntactic features from parsing, and speech recognition features.

3.2. Features

This section describes in more details the constraint features used in our maximum entropy model. The features defined need to capture domain knowledge. Our application is in the entertainment domain, where users can search by voice for songs, albums, artists, TV shows, movies, and apps. We define three sets of features: speech recognition features, semantic features and search result features.

Query features capture knowledge that can be extracted from the speech recognizer. These include:

- **LM score:** This is a real value score from the n-gram language model.
- **Rank position:** Rank position of first pass rank from the speech recognizer N-best list. Position 0 is the best hypothesis from the speech recognizer, and is our baseline.
- **Num tokens**: Number of tokens in the hypothesis.
- **Speech confidence**: Confidence score from the speech recognizer.

We also create a semantic feature based on pattern matching. Many voice queries are commands like "show me movies by Jim Carrey", "open Facebook". The intuition is that if a hypothesis matches a popular voice action pattern, it has a better chance to be a correct recognition. We parse each hypothesis with a finite state machine based pattern matcher, and create semantic parsing features for popular voice actions such as Open app, Play movie. If a hypothesis matches such a pattern, the **Semantic** feature is set to 1, otherwise it is set to 0.

The third category are features extracted from search results. The intention of search result features is to capture the degree to which a hypothesis is navigational, i.e., this hypothesis as search query often leads users to click a particular search result. the rationale being that if a hypothesis is navigational in this domain, it is likely to be a correct recognition. For example, typically if a user has spoken the title of a movie, and the candidate transcription is correct, the highest-ranking movie result has a feature score indicating a high degree of match with the candidate transcription. The second-ranked movie result would tend to have a score indicating a lower degree of match, since the title is different from the title spoken by the user. By contrast, when feature scores indicate that the highest-ranking and second-ranking results are both poor matches for the candidate transcription, the candidate transcription is less likely to be correct. Accordingly, we design the following features to capture navigational intent.

- **Number of results**: The feature value is the number of search results returned for a given query. If a query returns more search results it has a better chance to be correct. In the extreme case, if a query does not return any results, it is most likely a bad transcription.
- **Top result title match**: This feature computes the degree to which a hypothesis has a complete title match as top result.
- **Top category probability**: This is the probability score of the first result category. For example, if the first result is a movie result, this is the probability of being in the Movie category.
- **Next category probability**: This is the probability score for the second result category. Together with feature *top category probability*, it tells us how different the first result and second result are. The larger the gap is, the more chance a query is navigational.
- **Score for TV show**: A score that indicates how well the best TV show result matches the candidate transcription. For example, if the top-most item is a TV show, then this feature is the raw item score for the top-most item, otherwise it is 0.
- **Next score for TV show**: A score that indicates how well the second TV show result matches the candidate transcription.
- **Score for Album**: A score that indicates how well the best album result matches the candidate transcription.
- **Next Score for Album**: A score that indicates how well the second album result matches the candidate transcription;
- **Score for App**: A score that indicates how well the best APP result matches the candidate transcription;

- **Next score for App**: A score that indicates how well the second APP result matches the candidate transcription;
- **Score for Artist**: A score that indicates how well the best artist result matches the candidate transcription;
- **Next score for Artist**: A score that indicates how well the second artist result matches the candidate transcription;
- **Score for Movie**: A score that indicates how well the best movie result matches the candidate transcription;
- **Next score for Movie**: A score that indicates how well the second movie result matches the candidate transcription;
- **Score for Song**: A score that indicates how well the best song result matches the candidate transcription;
- **Next score for Song**: A score that indicates how well the second song result matches the candidate transcription;

4. EXPERIMENTS

4.1. Data Sets

We randomly sampled 22k utterances from an entertainment service. We had these utterances human-transcribed. A significant portion of these utterances (about 10%) contain background music or noise but no speech. These utterances are labeled with a special transcription symbol and are referred to as "*junk*" in our experiments. Sentence accuracy (SACC) [16] is computed by comparing speech recognition outputs with human transcriptions. A simple normalization is applied during comparison to ignore case and punctuation. We used 10-fold cross-validation to avoid overfitting of the classifier.

The speech recognition engine used is a standard large-vocabulary recognizer using a 5-gram language model trained from many sources, including search queries from the entertainment domain. The language model thus already has strong aggregated domain knowledge. The improvement we see below is mostly from adding query-level search result features.

4.2. Results

We first report results on all utterances, including utterances that should be rejected as irrelevant to the application domain.

Table 2 shows 10-fold cross-validation reranking results of the 2-best hypotheses on all utterances including these "junk" utterances. The first column is the fold number. The second column shows the baseline performance testing on this fold after training on the other nine, and the fourth column contains reranking results. The third column is the oracle reranking results, that is, if the correct hypothesis is in the list, it is artificially placed at the top. We can see that there is about a 9.2% gap between the baseline and the oracle ranking. Our reranking obtains stable improvement on each fold of the 10-folds tests, with an average of **3%**, improving SACC from **64.1%** to **67.1%**. A paired t-test [1] shows all results are statistically significant with p-value smaller than *0.001*.

Table 2: 10-fold cross validation results on all utterances, including "junk" utterances

Fold	Baseline	Optimal	Rerank
1	0.657	0.745	0.685
2	0.643	0.738	0.671
3	0.636	0.732	0.672
4	0.640	0.729	0.670
5	0.656	0.741	0.681
6	0.645	0.735	0.671
7	0.631	0.724	0.658
8	0.636	0.733	0.671
9	0.621	0.718	0.653
10	0.641	0.734	0.666
Average	0.641	0.733	0.671

In order to isolate the impact of "junk" utterances, since ideally these utterances should be rejected, we repeated the experiments without "junk" utterances. Results are summarized in Table 3. Note the absolute accuracy in this table is higher than that in Table 2 because the computation here ignores the "junk" utterances. We see a similar 3% improvement.

Table 3: 10-fold cross validation results on utterances excluding "junk" utterances

Fold	Baseline	Optimal	Rerank
1	0.694	0.796	0.723
2	0.672	0.774	0.710
3	0.678	0.773	0.709
4	0.694	0.784	0.719
5	0.678	0.773	0.706
6	0.674	0.774	0.703
7	0.677	0.781	0.714
8	0.665	0.769	0.700
9	0.682	0.781	0.708
10	0.700	0.793	0.729
Average	0.680	0.780	0.712

There are many utterances for which the correct recognition result is not among the N-best. These are the 22% of the utterances where oracle reranking still fails. For these utterances, reranking makes no difference. Table 4 reports performance on reranking two hypothesis limited to utterances

[1] https://en.wikipedia.org/wiki/Student's_t-test

that have the correct hypothesis among the top *20*. We can see even though the recognition accuracy on such utterances is fairly high, we can still achieve an additional **4%** SACC improvement, from **87.4%** to **91.4%**.

Table 4: 10-fold cross validation results on utterances that has the correct transcription in the N-best list

Fold	**Baseline**	**Rerank**
1	0.87	0.909
2	0.868	0.917
3	0.878	0.917
4	0.885	0.916
5	0.877	0.911
6	0.872	0.908
7	0.867	0.914
8	0.865	0.908
9	0.873	0.904
10	0.883	0.919
Average	0.874	0.914

4.3. Discussions

Since we have 3 categories of features, it is natural to ask which features are the most important. Table 5 shows the results of removing each feature category. We can see that removing any category of features causes losses, which means all features are useful. In particular, removing speech recognition features decreases the performance the most. This is not surprising since recognition features come from the speech recognizer, which is accurate most of the time. The semantic feature only provides small impact because the triggering ratio of this feature is very low. Removing all the search result features erases almost all gains, indicating their importance. However, result set features alone are not enough. They have to be provided as additional knowledge on top of the speech recognition features.

Table 5: Results with a subset of features

	SACC
All Features	0.914
Remove speech recognition features	0.615
Remove search result features	0.873
Remove semantic feature	0.910

We have reported results on reranking only 2 hypothesis. Table 1 shows that there are still more than half of the errors that can be potentially recovered by going beyond the top 2. A natural question then is how many hypotheses should we rerank? We varied N from 2 to 5, and Table 6 shows that reranking 2 or 3 hypotheses is best. When there are more hypotheses, the quality of hypotheses generally goes down and creates more noise for reranking.

Table 6: Results with ranking different N

N	SACC
1	0.874
2	0.914
3	0.910
4	0.905
5	0.904

Search result features can sometimes cause problems as well. If an utterance is asking for an App that is less navigational, but sounds close to a navigational App, the more navigational intent App may be picked as transcription. For example, an utterance "facial" is asking for some facial Apps, but the reranking picked "Facebook" as the top hypothesis.

Another issue is that we currently send two hypotheses for every utterance to the search backend. This creates a significant backend cost if the traffic is heavy. This could be optimized by selectively sending hypotheses to extract search features. One possible straightforward solution is to use speech recognition confidence as a filter to select only low-confidence utterances for rescoring. However, this approach does not work very well. We calculate the linear correlation coefficient [2] between the speech recognition confidence and class label. The coefficient is **0.43**, which indicates speech recognition confidence and class label are positively correlated but the correlation is not very strong. On the other hand, the coefficient between classification score and class label is **0.62**. This also indicates that the classification output is much more accurate than the first-pass speech recognition output. We can therefore use class confidence as a metric to reject utterances, which is more accurate than speech confidence.

5. CONCLUSION AND FUTURE WORK

This paper describes some early research on using search results as features to rescore speech N-best hypotheses. We formulate the problem as a binary classification problem and model it with a conditional maximum entropy model. We can achieve significant recognition accuracy improvements over a production-quality recognition baseline. There are many problems we are considering for future work. First issue is how to effectively select which utterances to rerank to avoid backend costs. Second, we can model the problem as a regression problem and use word error rate sensitive metrics as cost function. As found in [9], ranking approaches may yield more gain than classification approaches. Third, better features can likely improve accuracy even more.

6. REFERENCES

[1] Alex Franz and Brian Milch, "Searching the Web

[2] http://mathbits.com/MathBits/TISection/Statistics2/correlation.htm

by Voice," in *Proceedings 19th International Conference on Computational Linguistics (COLING)*, 2002, pp. 1213–1217.

[2] Johan Schalkwyk and Doug Beeferman and François Beaufays and Bill Byrne and Ciprian Chelba and Mike Cohen and Maryam Kamvar and and Brian Strope, "Google Search by Voice: A case study," *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*, 2010.

[3] Young-In Song and Ye-Yi Wang and Yun-Cheng Ju and Michael L. Seltzer and Ivan Tashev and Alex Acero, "Voice Search of Structured Media Data," in *Proceedings of ICASSP*, 2009.

[4] Mehryar Mohri and Fernando C. N. Pereira and Michael Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech and Language*, vol. 16(1), pp. 69–88, 2002.

[5] Preethi Jyothi and Leif Johnson and Ciprian Chelba and Brian Strope, "Large-scale Discriminative Language Model Reranking for Voice-search," in *Proceedings of NAACL-HLT*, 2012.

[6] Eugene Charniak and Mark Johnson, "Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking," in *Proceedings of ACL*, 2005.

[7] Michael Collins and Terry Koo, "Discriminative Reranking for Natural Language Parsing," *Computational Linguistics*, vol. 31(1), pp. 25–70, 2005.

[8] Joseph Keshet and Samy Bengio and Brian Roark, "A Survey of Discriminative Language Modeling Approaches for Large Vocabulary Continuous Speech Recognition," *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, 2009.

[9] Erinc Dikici and Murat Semerci and Murat Saraclar and Ethem Alpaydn, "Classification and Ranking Approaches to Discriminative Language Modeling for ASR," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21(2), pp. 291–300, 2012.

[10] Jianfeng Gao and Hisami Suzuki and Wei Yuan, "An Empirical Study on Language Model Adaptation," *ACM Trans on Asian Language Information Processing*, 2006.

[11] Hasim Sak and Murat Saraclar and Tunga Gung, "On-the-fly Lattice Rescoring for Real-time Automatic Speech Recognition," in *Proceedings of InterSpeech*, 2010.

[12] Tobias Kaufmann and Thomas Ewender and Beat Pfister, "Improving Broadcast News Transcription with a Precision Grammar and Discriminative Reranking," in *Proceedings of ISCA*, 2009.

[13] Michael Collins, "Discriminative Syntactic Language Modeling for Speech Recognition," in *Proceedings of ACL*, 2005.

[14] Hasim Sak and Murat Saraclar and Tunga Gungor, "Discriminative Reranking of ASR Hypotheses with Morpholexical and N-best-list Features," in *Proceedings of ASRU*, 2011.

[15] Minwoo Jeong and Sangkeun Jung and Gary Geunbae Lee, "Speech Recognition Error Correction Using Maximum Entropy Language Model," in *Proceedings of InterSpeech*, 2004.

[16] Hung-An Chang and Yun-Hsuan Sung and Brian Strope and Françoise Beaufays, "Recognizing English Queries in Mandarin Voice Search," in *Proceedings of ICASSP*, 2011.

[17] Kamal Nigam and John Lafferty and Andrew McCallum, "Using Maximum Eentropy for Text Classification," in *Proceedings of IJCAI Workshop on Machine Learning for Information Filtering*, 1999.

[18] Adwait Ratnaparkhi, "A Simple Introduction to Maximum Entropy Models for Natural Language Processing," Tech. Rep., Institute for Research in Cognitive Science, University of Pennsylvania, 1997.

[19] Stephen Della Pietra and Vincent Della Pietra and John Lafferty, "Inducing Features of Random Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19(4), pp. 380–393, 1997.

[20] Robert Malouf, "A Comparison of Algorithms for Maximum Entropy Parameter Estimation," in *Proceedings of CONLL*, 2002.