# INCORPORATING USER FEEDBACK TO RE-RANK KEYWORD SEARCH RESULTS

*Scott Novotney, Kevin Jett and Owen Kimball*

Raytheon BBN Technologies, Cambridge, MA, USA
{snovotne, kjett}@bbn.com

## ABSTRACT

This paper capitalizes on user feedback of a keyword search engine to improve search performance on queries users are actively searching for. We assume users give a binary label as to whether a hypothesized token is correct. This signal is used to train a support vector machine to re-rank lattice posteriors using additional features derived from automatic speech recognition. We simulate user feedback using 1800 hours of English Fisher conversational telephone speech as a search corpus and the Switchboard corpus as our training corpus. Our novel contribution focuses on combining keyword specific and keyword independent models, improving search precision by 5% absolute over using one keyword independent model alone. Clustering keyword training data into multiple models based on their false alarm behavior gives even greater gains, achieving a 9% increase in precision over one keyword independent model.

**Index Terms**: keyword search, re-ranking, CTS, user feedback

## 1. INTRODUCTION

Our users are eager to tell us when our keyword search system is wrong. Their enthusiasm for correcting search errors (false alarms) provides us with valuable annotations that can improve performance on the very words that matter most to users. While transcription of new data is time consuming and expensive [22], these annotations require only the good will of our users. These keyword specific binary labels indicate whether a word occurred in a region of speech. We do *not* have transcription of the surrounding context so cannot simply treat this resource as additional training data for the acoustic and language model. Instead, we seek a method that can capitalize on both positive and negative labels to improve keyword search performance. Relevance feedback from a user is an integral area of research in information retrieval [2, 3]. Many fields, such as web search, must infer a user's implicit feedback [4] to improve search results [5]. Luckily, a keyword search system has a clear relevance task – either the query did or did not occur. This binary definition makes relevance judgements easy to contribute [6]. While not as rich for improved model estimation as full transcription, these binary labels are valuable training data, especially for new queries which do not appear in the training data.

Prior work has explored user verifications and post-recognition re-ranking of keyword search results. One use of verifications is semi-supervised estimation of both the acoustic [7, 8, 9] and language [10, 11, 12] models. The manually verified word can be added in conjunction with high confidence neighbors to both models as well as unsupervised discriminative acoustic modeling. This paper explores post-recognition techniques which more quickly incorporate user feedback into a production environment without requiring speech to test model re-estimation. Additionally, we will explore models that can incorporate features which may be difficult in a decoder.

Other work has considered this task of post-recognition re-ranking of speech output. Prior work used development data as a proxy of manual verification to build either logistic regression or support vector machines to re-rank confusion networks [13, 14, 15, 16, 17] with the goal of minimizing word error rate (WER). Instead of WER, one can optimize for keyword search performance [18]. The posteriors of all words to the left and right of a hypothesized word in a speech lattice were used as features as input to a support vector machine to re-rank search results. The top-most and bottom-most results were assumed to be true and false respectively as manual annotations were not available. This pseudo-relevance feedback led to an 18% relative improvement in mean average precision on a Mandarin read speech task. Most closely related to this work [14], re-ranking of confusion network bins reduced WER by 2.8% absolute using a combination of features derived from the recognition output and word identity. Additionally, re-scoring of search results (as in this work) achieved improvements ranging from 0.45% to 0.9% in MTWV [19].

This work differs from the previous literature in two ways. First, we assume sufficient annotations are available beyond the initial training corpora to build keyword specific models. As explained in Section 2.3, each keyword will have at least ten labeled examples of whether or not it occurred in a time region. This is in contrast to the relative rarity of content words in randomly sampled speech corpora traditionally assumed in prior work. This allows to explore the variance bias trade off between one keyword independent model and multiple keyword models. Second, we focus on methods that reduce annotation costs. By sharing training examples and parameters across subsets of keywords, we achieve greater performance than training either keyword independent or dependent models alone. Section 4 explores various techniques for parameter sharing.

This paper makes the following conclusions: I) User verifications are a useful resource to improve the quality of the top keyword search results. II) System combination of keyword dependent models with a global model reduces annotation needs and outperforms either method. III) Clustering keywords based on similar search behavior provides better gains than one global model.

## 2. EXPERIMENTAL SETUP

Our experiments simulated the workflow of a user engaged with our keyword search system over time. After deploying an LVCSR system, we index an evolving corpus of audio from one domain. Our simulated user interacts with the system by querying occurrences of a set of keywords and browsing a ranked list over a number of trials (days, weeks, etc. . . ). For instance, an auditor of a call center may have a set of trigger words that they are tasked with monitoring over the course of a week. The user provides us with binary labels as to whether each of the top search results are correct or a false alarm.

For speech recognition and indexing, we used a large vocabulary continuous speech recognizer Byblos [20] with a decoding speed around two times faster than real time. The system indexed whole words and no special effort was made for out of vocabulary since all words evaluated were in vocabulary. Search results returned a ranked list of a word by its posterior score in a confusion network [21].

### 2.1. Re-Ranking Metrics

Keyword search is a ranked retrieval task. A KWS engine returns a set of time regions along with a score, which creates a ranking over the set of results. In this paper, the results are automatically segmented utterances. If the proposed keyword appears anywhere in the utterance, it is correct. The appropriate metric for this ranked retrieval is mean average precision (MAP) defined as

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\text{Precision}(R_{jk}) \qquad (1)$$

where $Q$ is the set of keywords, $U = \{u_1, \ldots, u_{m_j}\}$ is the set of size $m_j$ of relevant utterances for query $q_j$ and $R_{jk}$ is the set of ranked utterances from the top until utterance $u_k$ is reached. Precision$(R_{jk})$ is computed as the percent of relevant utterances in $R_{jk}$ divided by the size of $R_{jk}$. Queries are equally weighted, meaning MAP is the arithmetic average of the per-query average precision. Higher MAP scores are better, with a minimum of zero and maximum of one.

The task of this paper is more constrained than keyword search. Instead of improving overall search performance, we seek to improve the quality of a fixed set of search results. All correct results should be ranked higher than all false alarms. Both recall and the precision of the entire list will be fixed since the number of correct results is unchanging.

Therefore, we will report results using a modestly modified metric called "trimmed MAP" which does not penalize documents not returned in the search results. Of course, for the general task of keyword search, this metric is deficient as it favors short, high confidence lists. But the goal of this paper is to re-rank a *fixed* list, so this is of no concern. We prefer this metric to MAP as it highlights the quality of the "first page" of search results, which we have noticed has a large impact on user perception.

### 2.2. A Simulated User Feedback Corpus

We derived our simulated user verifications from 1800 hours of the Fisher collection of transcribed English conversational telephone speech (CTS) [22]. This corpus was randomly partitioned into nine subsets of 200 hours each, separately indexed by the LVCSR system. Instead of simulating user relevance feedback on the top ten results of a keyword across all 1800 hours, we have nine separate top ten results, each over a separate 200 hours. We constructed the experiment this way in order to simulate a shifting corpus through time, say over nine days, enabling us to measure the benefit of more user annotations. The LVCSR system was trained on the 370 hour Switchboard corpus [23] from which we estimated acoustic and language models and derived the decoding dictionary. Pronunciations were manually generated, both from those in Switchboard and those queries not in Switchboard but in Fisher.

### 2.3. Keywords for Evaluation

We evaluated performance on a subset of words appearing in the Fisher corpus selected to reflect words that users might actually correct. We decoded a subset of Fisher separate from the 1800 hours with the Switchboard LVCSR models using a decoding dictionary that covered all of Fisher and Switchboard. Next, we ranked all words in the held-out set by their precision at ten and selected the worst 100 performing words as our evaluation list. This is a fair set of keywords as they are *not* necessarily the worst performing keywords in the 1800 hours. Some examples are BRITNEY, CRAIGSLIST, SUMMERS and UPENN.

These keywords were then used to simulate user corrections and measure improvements in trimmed MAP. For each keyword, we have nine separate ranked lists. We report results on the cross validation of these nine lists. We also vary the depth of the lists from 10, 25 and 50 – assuming manual verifications for all results.

## 3. LEARNING TO RE-RANK

The task is to order a set of $d = \{10, 25, 50\}$ hypothesized instances of a keyword initially sorted by the posterior probability from the LVCSR system. Each instance has an unknown binary label of correct or false alarm and we aim to order all correct instances before any false alarms. We have available to us a training corpus of $k = 9$ user feedback trials. Each trial consists of a simulated user annotating the top $d$ instances of the keyword that appeared in the $k^{\text{th}}$ subset of the search corpus. We perform cross validation, holding out one trial and training a re-ranker on the remaining eight trials ($8 \cdot d$ training samples total per keyword).

### 3.1. Feature Extraction

Each observation is a pair $(q, u)$ with keyword query $q$ and utterance $u$. The utterance is tokenized by an LVCSR system, producing a confusion network from which to extract features. Due to efficient run time constraints, we limit feature extraction to the confusion network and those efficiently derived from the entire conversation side. Multiple occurrences of a keyword in the same utterance were treated as separate observations. The first class of features were extracted only from the confusion bin the word occurred in.

1. Word posterior and log of posterior.
2. Relative rank of the keyword.
3. Keyword duration relative to mean keyword length.
4. Frequency of the bigrams in the training data.
5. Binary indicator if the silence token is more likely than the keyword.
6. Binary indicator if the keyword was the most likely word in the bin.
7. Expected counts of each word in the same confusion bin as the keyword.
8. Entropy of confusion bin.

We also considered features in the entire utterance.

1. Expected counts of each word in left or right bins.
2. tf-idf score of each words in the entire utterance.
3. Identity of word *pairs* to the left and right, weighted by the product of the word posteriors.
4. Binary indicator if the most likely n-grams including the keyword were seen in the training data.
5. Estimate topic labels of the utterance.

The estimated topic labels were unsupervised [24] and performed nearly as well as the manual topic labels assigned to the Fisher corpus. We found hard assignment of the most likely topic label to result in better performance than a distribution over all 50 topics. We pruned features which had an expected count less than 0.1, leaving on the order of 100,000 features, most consisting of word types.

We make a distinction between the presence of another word in the same bin, in the neighboring bins or anywhere in the utterance, meaning a word type may appear up to three times in a feature vector. We did not add lexical features (such as pronunciation length or occurrences in training) since we are estimating separate models per keyword and these would be constant. Additionally, we did not use direct acoustic features such as signal to noise ratio or prosodic features due to the modest impact in earlier work [14].

### 3.2. Re-ranking Model

We use a support vector machine with a loss function that minimizes average precision of the training labels [25]. Let $R_{ni}(q) = \{(u_1, y_1), (u_2, y_2), \ldots, (u_n, y_n)\}$ be the top $n$ utterances hypothesized for keyword $q$ in the 200 hour search trial $i$ as well as the binary label $y_j$ whether keyword $q$ was correct or false alarmed in utterance $u_j$.

If given a feature function $\Phi(q, u)$ which maps from observed utterance $u$ and keyword $q$ to a $d$-length vector of real valued features, we seek to find a weight vector $w \in \mathcal{R}^d$ which is minimized by

$$\frac{1}{2} w \cdot w + C \sum \xi_{i,j,k} \tag{2}$$

subject to the constraints that for each query $q_i$ and set $R_{ni}$

$$\begin{aligned} \forall (u_j, u_k) \in R_{ni} \text{ s.t. } y_j > y_k : \\ w \cdot \Phi(q_i, u_j) \geq w \cdot \Phi(q_i, u_k) + 1 - \xi_{i,j,k} \end{aligned} \tag{3}$$

where $C$ is a parameter to trade off between margin size and training error and $\xi_{i,j,k}$ are non-negative slack variables. Full details are explained in the description of the package SVM$^{\text{light}}$ [26]. This equation seeks to find the smallest weight vector that ensures all correct instances of a keyword occur before any incorrect instances. Utterance pairs with the same relevance judgement (both correct or false alarmed) are not constrained and ignored for training purposes. We found no gain for using a loss function that ranked results with the same relevance judgement by their posterior.

The regularization weight $C$ was tuned on a small development set from Fisher and left fixed throughout the experiments. Sweeping the regularization weight could improve performance slightly, but in general results were not sensitive to the regularizer. We used a linear kernel as experiments with both polynomial and RBF kernels performed no better. Logistic regression had a strong appeal as it produced calibrated probabilities and was more amenable to system combination and hierarchical modeling. However, performance with an SVM drastically outperformed logistic regression due to the direct optimization of the ranking cost function as well as better robustness to large feature spaces with sparse training data.

## 3.3. High Resource Re-ranking Performance

Re-ranking improved trimmed MAP by an average of 5% absolute across varying depths. We swept the ranked lists from the top 10, 25 or 50 results. A separate SVM was trained for each keyword at each depth. As described in Section 2.2, nine 200 hour subsets of Fisher were used to extract separate keyword results. We simulated user feedback for all nine lists and report 9-fold cross validation, training on eight and testing on the ninth. When using the posterior alone to rank results, the average trimmed MAP of the top ten results was 0.732. Re-ranking the results increased trimmed MAP to 0.786. Similar gains were seen when training and re-ranking the top 25, which increased from 0.669 to 0.721 trimmed MAP. Re-ranking the top 50 hits also increased from 0.629 to 0.679.

Figure 1 plots the per-keyword gain (or loss) for re-ranking the first subset of depth 10. To reiterate, the number of correct results in the list did not change, only the ordering. The lowest performing keywords showed great improvement in trimmed MAP, with some achieving perfect scores and minimal degradation. Unfortunately, keywords which began with perfect lists (all correct instances on top) suffered from re-ranking. Most of these keywords had few total number of correct instances, making them very sensitive to any re-ordering of the search results.

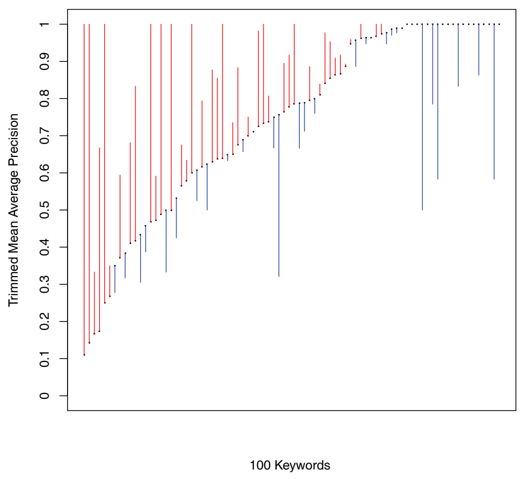

**Fig. 1**. *Per-Keyword Re-ranking Performance* - We show the gain (or loss) for re-ranking the top 10 results of each keyword. They are sorted left to right by their baseline performance from worst to best. When performance helps (trimmed MAP increases), the line is red and improves from bottom to top, otherwise it is blue and trimmed MAP decreases from top to bottom. Re-ranking dramatically improves the precision of low-frequency terms. Unfortunately, high-precision terms have nowhere to go but down and some suffer.

Table 1 details the improvement in trimmed MAP for the most important features. To find out the relative importance of each feature, we greedily added feature classes (described in Section 3.1) to the initial posterior. The most important was expected counts of words in the same bin. Next was the unsupervised topic label, then the indicator if neighboring $n$-grams were seen in the training data. Finally, the log posterior and neighboring words had a small impact, with the remaining features added up to the five point gain. The value of the unsupervised topics increased as the re-ranking depth increased, from a marginal increase in trimmed MAP from 0.02 when re-ranking the top ten to a 0.02 gain when re-ranking the top 50. Overall, these features are efficient to extract from a consensus network or an entire conversation side, allowing for a fast post-recognition process to improve the precision of the top results.

| | Ranked Depth | | |
|---|---|---|---|
| **Feature** | **10** | **25** | **50** |
| Posterior | .732 | .669 | .629 |
| +Words | .762 | .677 | .633 |
| +Topic | .765 | .681 | .651 |
| +LM | .773 | .697 | .658 |
| +Log post. | .777 | .709 | .668 |
| +Neighbors | .780 | .718 | .677 |
| +Remaining | .786 | .721 | .679 |

**Table 1**. *Value of Additional Features* - Each row adds an additional set of features in order of their relative improvement to trimmed MAP. We re-rank either the top 10, 25 or 50 results (cols 2-4) and report trimmed MAP with 9-fold cross validation. Each set of keyword specific results was re-ranked using an SVM estimated on eight times as many labels. In total, re-ranking improved trimmed MAP by around 5% across a variety of depths.

## 4. REDUCING USER ANNOTATIONS

While the previous section demonstrated meaningful improvements in re-ranking, it assumed eight separate trials (1600 hours of searched audio total) were available to estimate per-keyword SVMs. Each keyword had between 80 and 400 verified search results, depending on the re-ranking depth. Such resource rich training data may require too long a lag in improved performance. Additionally, this requires diligent effort by the users to annotate each word, which may be overly optimistic in a production system.

To lessen this annotation burden, we experimented with system combination of keyword dependent (one model per keyword) and keyword independent (one model for all keyword trained on more data) results. This section considers only re-ranking the top 10 and now, instead of all 80 samples being available, we will report results across a range of training data from 10 to 80 per keyword.

## 4.1. System Combination

For each training condition, we first estimated a separate SVM for each of the 100 queries. We then estimated a keyword independent model trained on all the combined samples (100 times as many observations). Note most prior work *only* estimated a keyword independent model.

As we swept the number of training samples per keyword from 10 to 80, the global re-ranker improves trimmed MAP by 1.5% on average over the baseline of using only the posterior (compare the black straight line to the blue dotted line in Figure 2). However, the global model is quickly beaten out by the keyword dependent model after 30 samples – 3 trials (compare thee black dotted line to the blue dashed line). Furthermore, the dependent model continues to improve as more labels are available while the global model quickly tapers off. The underlying behaviors that cause false alarms are not global, but specific to each keyword.

To achieve the best of both worlds, we combined the keyword dependent and independent model scores. Since an SVM does not produce probabilistic scores, we experimented with a variety of methods to combine these real valued scores. We calibrated the SVM scores by estimating a keyword independent logistic regression model using a 50 hour held-out set to combine the two feature scores. Surprisingly, simply normalizing the raw scores to fall in $[0, 1]$ and interpolating the two fake probabilities gave equally good performance. As the amount of available in-domain data increased, the optimal interpolation weight gave more weight to the dependent scores. With no additional training data, system combination with the background model improved performance as seen in Figure 2. Although the background model by itself is quickly outperformed by the keyword dependent models as the amount of data increases, it provides a complementary gain.

## 4.2. Clustered Background Models

Merging all training samples together to form a global model gave only marginal gains over the simple baseline. Still, interpolation with this broad background model did improve over using the keyword dependent models alone. We can increase this system combination gain by estimating more accurate background models for each keyword. Instead of using all the data, this section will estimate background models on varying subsets of all the training data. This approach draws inspiration from multilevel modeling (also known as hierarchical modeling) [27]. Multilevel modeling exploits natural hierarchies or clustering within data to share parameter estimates, improving inference. Clustered phoneme states [28] for acoustic modeling are a natural example from the speech research community.

We consider two keywords similar if they false alarm or hit in the presence of similar features. Note that this is not a notion of *semantic* similarity, but instead one of *task* similarity. For instance, *CAT* and *DOG* may be semantically similar, or *CAT* and *HAT* may be phonetically similar, but their behavior in false alarming may be unrelated.

Instead, we define similarity for two keywords $q_1$ and $q_2$ as the cosine distance of the unit-length normalized feature vectors estimated from the linear-kernel SVMs

$$\text{sim}(q_1, q_2) = \frac{w_1 \cdot w_2}{||w_1|| \cdot ||w_2||} \tag{4}$$

where $w_1$ and $w_2$ are the parameter weights from two SVMs trained on the ranked lists of $q_1$ and $q_2$ respectively. Projecting the keywords into this space of discriminative feature weights will hopefully result in better parameter sharing than one global model.

Using this similarity metric, we built an agglomerative hierarchical [29] tree of all 100 keywords. The feature vector of each keyword was the feature weights of an SVM trained on 80 labeled examples. This let us sweep the number of clusters from one (one background model) to six (64 background models) by partitioning the tree at different depths. For each cluster, we pooled the component keyword training samples together and estimated one SVM. We then scored the test samples using both the keyword dependent model and the clustered background model and, as before, interpolated the two normalized scores.

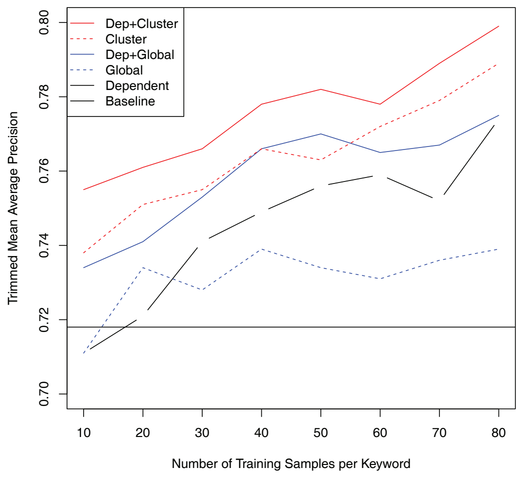

**Fig. 2**. *Benefit of System Combination* - We estimated an SVM to re-rank KWS results using 10 to 80 training samples per keyword and re-scored the top 10 results. Instead of estimating a separate keyword per model (black w/dots) or one global model for all words (dashed blue), we achieved greater success by interpolating the two scores (solid blue), improving over the baseline precision (solid black). Even further success was achieved with clustered background models (dashed red line) which when interpolated with the dependent models provided the best results across a range of training amounts (solid red line).

Clustered background models gave meaningful gains over only one background model, improving precision by 2% absolute across a range of training samples. The optimal number of clusters was sixteen (around seven keywords per cluster) with minimal sensitivity to the number of clusters. Figure 2 shows the gain for clustered system combination. The clustered models (without system combination) outperformed both the keyword dependent and independent models. System combination of the clustered models with the dependent models gave the highest performance. In total, parameter sharing increased trimmed MAP by 3-5% depending on the number of training samples and always outperforming the keyword dependent models. The practical benefit was that parameter sharing cuts the number of required samples in third - a clustered model trained on 10 samples has the same performance as a dependent model trained on 30.

This section used the labeled verifications in two ways: to train a re-ranking SVM and to estimate keyword similarity (through the SVM parameter vector). As the amount of training data increases, the parameter vectors are better estimated, leading to both improved re-ranking *and* cluster assignments. To test the importance of cluster estimation, we used three different methods of assigning similarity. First, we used the fair feature vectors estimated with only the training data available at each operating point (meaning eight different hierarchical trees). Second, we used tf-idf weighted bag of word counts from neighboring words (requiring no verifications). Third, we used the feature weights from the best estimated SVMs trained with 80 samples per keyword.

At each operating point in Figure 3, there are three distinct clusters: the unsupervised, fair and oracle. The unsupervised and oracle cluster remains constant as the number of training samples increase. However, the fair cluster improves as more samples are available to estimate the hierarchical tree.

For each operating point then, the process is as before: estimate keyword dependent SVMs, estimate shared SVMs using one of the three clusters and combine combine the scores from the two models for each keyword. Note that the available training samples are always the same – all that changes is the background model used in system combination. Figure 3 demonstrates three conclusions. First, unsupervised cluster estimation is surprisingly strong, providing meaningful gains over no parameter sharing. Second, clustering based on SVM parameter similarity is better than on feature counts. Third, clusters estimated with little training samples are not degraded much from the optimal cluster.

## 5. CONCLUSIONS

We demonstrated that an SVM re-ranker trained on simple binary annotations of keyword search results improves the precision of the top keyword search results. Instead of one model for all keywords, we interpolated keyword-dependent models with clustered background models. This reduced the amount of annotation required, allowing us to quickly improve search result quality with small amounts of manual annotation. Such reductions are critical in deployment as users have minimal patience for annotation.

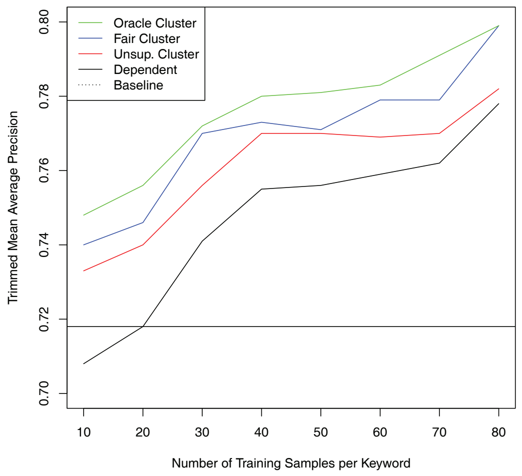

**Fig. 3**. *Impact of Cluster Estimation on Trimmed MAP* - We sweep the amount of training data from 10 to 80 samples. However, the lines now represent different methods of clustering. Interpolating the dependent scores (rising black line) with any cluster provides a gain. Grouping based on unsupervised bag of words (red line) provides a modest gain but begins to taper off once the amount of training data saturates. Fair cluster estimation in the SVM feature space (blue line) provides a bigger gain because keywords are grouped based on their consistency of false alarming rather than semantic relatedness. These fair clusters are estimated only with the available training data at each step. Using all available data to estimate clusters (green) gives a smaller gain, indicating that cluster estimation is robust with small amounts of data.

Further work should explore a more principled method of system combination such as hierarchical modeling [30]. However, the benefit of better parameter sharing must be weighed against the worse empirical performance of logistic regression compared to support vector machines. Structured inference of the entire ranked list could also improve performance [31]. While the SVM was estimated with constraints derived from the entire ranked lists, predictions were independently generated, one result at a time.

Beyond improved model inference, we hope to further integrate verifications into the acoustic and language model. In particular, we are eager to try discriminative techniques for both the acoustic and language model to capitalize on the verified false alarms. Finally, one could explore improved methods of collecting user annotations [32]. Techniques which inspire users to provide feedback such as gamification [33] may help in quickly bootstrapping a keyword search system.

## 6. REFERENCES

[1] David RH Miller, Michael Kleber, Chia-Lin Kao, Owen Kimball, Thomas Colthurst, Stephen A Lowe, Richard M Schwartz, and Herbert Gish, "Rapid and accurate spoken term detection," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[2] J. J. Rocchio, "Relevance feedback in information retrieval," in *The SMART Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed., Prentice-Hall Series in Automatic Computation, chapter 14, pp. 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.

[3] Gerard Salton and Chris Buckley, "Improving retrieval performance by relevance feedback," *Readings in information retrieval*, vol. 24, no. 5, pp. 355–363, 1997.

[4] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay, "Accurately interpreting clickthrough data as implicit feedback," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 154–161.

[5] H.J. Kim, S. Tong, N.M. Shazeer, and M. Diligenti, "Modifying search result ranking based on implicit user feedback," Feb. 25 2014, US Patent 8,661,029.

[6] Ciprian Chelba, Timothy J. Hazen, and Murat Saralar, "Retrieval and browsing of spoken content," *IEEE Signal Processing Mag*, pp. 39–49, 2008.

[7] Frank Wessel and Hermann Ney, "Unsupervised Training Of Acoustic Models For Large Vocabulary Continuous Speech Recognition," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 2001, pp. 23–31.

[8] Lori Lamel, Jean-Luc Gauvain, and Gilles Adda, "Lightly supervised acoustic model training," in *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop*, 2000, pp. 115–129.

[9] Jeff Ma and Spyros Matsoukas, "Unsupervised Training on a Large Amount of Arabic Broadcast News Data," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007, pp. 349–352.

[10] Michiel Bacchiani and Brian Roark, "Unsupervised Language Model Adaptation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003, pp. 220–224.

[11] Yik-Cheung Tam and Paul Vozila, "A Hierarchical Bayesian Approach for Semi-supervised Discriminative Language Modeling," in *Proceedings of Annual Conference of the International Speech Communication Association*, 2011.

[12] Scott Novotney, *Incorporating Weak Statistics to Improve Semi-Supervised Language Modeling*, Ph.D. thesis, Johns Hopkins University, 2014.

[13] Thomas Pellegrini and Isabel Trancoso, "Improving asr error detection with non-decoder based features," in *Proceedings of Annual Conference of the International Speech Communication Association*, 2010, pp. 1950–1953.

[14] Victor Soto, Erica Cooper, Lidia Mangu, Andrew Rosenberg, and Julia Hirschberg, "Rescoring confusion networks for keyword search," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7088–7092.

[15] Alexandre Allauzen, "Error detection in confusion network," in *Proceedings of Annual Conference of the International Speech Communication Association*, 2007.

[16] Takaaki Hori, I Lee Hetherington, Timothy J Hazen, and James R Glass, "Open-vocabulary spoken utterance retrieval using confusion networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007, pp. 73–76.

[17] Wei Chen, Sankaranarayanan Ananthakrishnan, Ravindra Kumar, Ranga Prasad, and Prem Natarajan, "Asr error detection in a conversational spoken language translation system," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7418–7422.

[18] Hung yi Lee, Tsung wei Tu, Chia-Ping Chen, Chao yu Huang, and Lin shan Lee, "Improved spoken term detection using support vector machines based on lattice context consistency," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, May 2011, pp. 5648–5651.

[19] Jonathan G Fiscus, Jerome Ajot, John S Garofolo, and George Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proceedings of Annual Conference of the International Speech Communication Association*, 2007, vol. 7, pp. 51–57.

[20] R. Prasad, S. Matsoukas, CL Kao, J.Z. Ma, DX Xu, T. Colthurst, O. Kimball, R. Schwartz, J.L. Gauvain, L. Lamel, et al., "The 2004 BBN/LIMSI 20xRT English conversational telephone speech recognition system," in *Proceedings of Annual Conference of the Inter-*

*national Speech Communication Association*, 2005, pp. 1645–1648.

[21] Lidia Mangu, Eric Brill, and Andreas Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *CoRR*, vol. cs.CL/0010012, 2000.

[22] Owen Kimball, Chai-Lin Kao, Teodoro Arvizo, John Makhoul, and Rukmini Iyer, "Quick Transcription and Automatic Segmentation of the Fisher Conversational Telephone Speech Corpus," in *Proceedings of 2004 Rich Transcriptions Workshop*, 2004.

[23] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1994, pp. 517–520.

[24] Tim Leek, Richard Schwartz, and Srinivasa Sista, "Probabilistic approaches to topic detection and tracking," in *Topic detection and tracking*, pp. 67–83. Springer, 2002.

[25] Thorsten Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 133–142.

[26] Thorsten Joachims, "Training linear svms in linear time," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2006, KDD '06, pp. 217–226, ACM.

[27] Andrew Gelman and Jennifer Hill, *Data analysis using regression and multilevel/hierarchical models*, Cambridge University Press, 2006.

[28] Steve J Young and Philip C Woodland, "State clustering in hidden markov model-based continuous speech recognition," *Computer Speech & Language*, vol. 8, no. 4, pp. 369–383, 1994.

[29] Abdelmoula El-Hamdouchi and Peter Willett, "Comparison of hierarchic agglomerative clustering methods for document retrieval," *The Computer Journal*, vol. 32, no. 3, pp. 220–227, 1989.

[30] Jenny Rose Finkel and Christopher D Manning, "Hierarchical Bayesian domain adaptation," in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2009, number June, pp. 602–610, Association for Computational Linguistics.

[31] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin, "Learning structured prediction models: A large margin approach," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 896–903.

[32] Chris Callison-Burch, "Fast, Cheap, and Creative : Evaluating Translation Quality Using Amazons Mechanical Turk," in *Language and Speech*. 2009, vol. 1, pp. 286–295, Association for Computational Linguistics.

[33] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. ACM, 2011, pp. 9–15.