# Using Query Performance Predictors to Improve Spoken Queries

Jaime Arguello[1], Sandeep Avula[1], and Fernando Diaz[2]

[1] University of North Carolina at Chapel Hill
[2] Microsoft Research

**Abstract.** Query performance predictors estimate a query's retrieval effectiveness without user feedback. We evaluate the usefulness of pre- and post-retrieval performance predictors for two tasks associated with speech-enabled search: (1) predicting the most effective query transcription from the recognition system's n-best hypotheses and (2) predicting when to ask the user for a spoken query reformulation. We use machine learning to combine a wide range of query performance predictors as features and evaluate on 5,000 spoken queries collected using a crowd-sourced study. Our results suggest that pre- and post-retrieval features are useful for both tasks, and that post-retrieval features are slightly better.

## 1 Introduction

Speech-enabled search allows users to formulate queries using spoken language. The search engine transcribes the spoken query using an automatic speech recognition (ASR) system and then runs the textual query against the collection. Speech-enabled search is increasingly popular on mobile devices and is an important component in multimodal search interfaces and assistive search tools [12]. While speech is a natural means of communicating an information need, spoken queries pose a challenge for speech-enabled search engines. In a recent study, 55% of all spoken queries had recognition errors that caused a significant drop in retrieval performance [7].

The goal of *query performance prediction* is to estimate a query's effectiveness without feedback. Current approaches are classified into *pre-* and *post-retrieval* measures. Pre-retrieval measures are computed without conducting a full retrieval from the collection and capture evidence such as the query terms' specificity [6, 2, 18] and topical relatedness [5]. Post-retrieval measures are computed from the query's retrieval from the collection and capture evidence such as the topical coherence of the top results [2] and the rank stability [1, 17, 19, 20].

We investigate the usefulness of query performance predictors for two tasks associated with speech-enabled search: (1) re-ranking the ASR system's n-best list and (2) deciding when to ask for a spoken query reformulation. While ASR systems typically output the single most confident transcription of the input speech, internally they construct a ranked *n-best* list of the most confident hypotheses. In our n-best list re-ranking task, the input to the system is a spoken query's n-best list, and the goal of the system is to predict the candidate transcription from the n-best list that maximizes retrieval performance, which may not necessarily be the top candidate.

In certain situations, a spoken query may perform poorly due to ASR error or the user's failure to formulate an effective query. In our second predictive task, the input to the system is a spoken query (specifically, the top candidate from the ASR system's n-best list), and the goal of the system is to predict whether to run the input query or to ask for a reformulation.

For both tasks, we use machine learning to combine a wide range of performance predictors as features. We trained and tested models using a set of 5,000 spoken queries that were collected in a crowdsourced study. Our spoken queries were based on 250 TREC topics and were automatically transcribed using freely available APIs from AT&T and WIT.AI. We evaluate our models based on retrieval performance using the TREC 2004 Robust Track collection.

## 2 Related Work

The goal of *query performance prediction* is to estimate a query's performance without user feedback. Pre-retrieval measures capture evidence such as the query's specificity, topical coherence, and *estimated* rank stability [5]. In terms of query specificity, different measures consider the query terms' inverse document frequency (IDF) and inverse collection term frequency (ICTF) values [2, 6, 18]. Other specificity measures include the *query-scope*—proportional to the number of documents with at least one query term—and the *simplified clarity*—equal to the KL-divergence between the query and collection language models [6]. Topical coherence can be measured using the degree of co-occurrence between query terms [5]. Finally, the rank stability can be estimated using the query terms' variance of TF.IDF weights across documents in the collection [18].

Post-retrieval measures capture evidence such as the topical coherence of the top results, the *actual* rank stability, and the extent to which similar documents obtain a similar retrieval score. The *clarity* score measures the KL-divergence between the language model of the top documents and a background model of the collection [2]. Rank stability approaches perturb the query [17, 20], the documents [19], or the retrieval system [1], and measure the degree of change in the output ranking. The assumption is that more effective queries produce more stable rankings. Finally, the auto-correlation score from Diaz [4] considers the extent to which documents with a high text similarity obtain similar scores. Pre- and post-retrieval performance predictors have been applied to IR tasks such as reducing natural language queries [8, 16] and predicting the effectiveness of different query reformulations [3, 13].

Prior work in the speech recognition domain also considered improving spoken query recognition using evidence similar to the query performance predictors mentioned above. Mamou *et al.* [10] focused on re-ranking the n-best list using term co-occurrence statistics in order to favor candidates with semantically related terms. Li *et al.* [9] combined language models generated from different query-click logs to bias the ASR output in favor of previous queries with clicks. Peng *et al.* [11] focused on re-ranking the n-best list using post-retrieval evidence such as the number of search results and the number of exact matches in the top results. We extend this prior work in three ways. First, in addition to re-ranking the n-best list, we consider the task of predicting when to ask for

a spoken query reformulation. Second, we combine a wider range of pre- and post-retrieval performance predictors as features. Finally, we evaluate in terms of retrieval performance instead of recognition error.

## 3 Data Collection

In the next sections, we describe the user study that we ran to collect spoken queries, our search tasks, the ASR systems used, and our spoken queries.

**User Study.** Spoken queries were collected using Amazon's Mechanical Turk (MTurk). Each MTurk Human Intelligence Task (HIT) asked the participant to read a search task description and produce a recording of how they would request the information from a speech-enabled search engine.[3]

The study protocol proceeded as follows. Participants were first given a set of instructions and a link to a video explaining the steps required to complete the HIT. Participants were then asked to click a "start" button to open the main voice recording page in a new browser tab. While loading, the main page asked participants to grant access to their computer's microphone. Participants were required to grant access in order to continue. The main page provided participants with: (1) a button to display the search task description in a pop-up window, (2) Javascript components to record the spoken query and save the recording as a WAV file on the participant's computer, and (3) an HTML form to upload the WAV file to our server.

Within the main voice recording page, participants were first asked to click a "view task" button to display the search task description in a pop-up window. The task was displayed in a pop-up window to prevent participants from reading the task while recording their spoken query.[4] Participants were instructed to read the task carefully and to "imagine that you are looking for information on this specific topic and that you are going to ask a speech-enabled search engine for help in finding this information". Participants were asked to "not try to memorize the task description word-by-word". The instructions explained that our goal was to "learn how someone might formulate the information request as naturally as possible".

After reading the task, participants were asked to click a "record" button to record their spoken query and then a "save" button to save the recording as a WAV file on their computer. Next, participants were instructed to upload the saved WAV file to our server. The default WAV filename included the MTurk assignment ID, which is unique to each accepted MTurk HIT. The assignment ID was used by our server to check the validity of the uploaded WAV file. If the uploaded file was valid, the participant was then given a 10-character completion code. Finally, participants were asked to validate and submit the code to complete the HIT. As described in more detail below, we used a set of 250 search tasks and collected 20 spoken queries per search task, for a total of 5,000 spoken queries. Each HIT was priced at $0.15 USD.

[3] Our source code and search task descriptions are available at: `http://ils.unc.edu/~jarguell/ecir2016/`.

[4] Participants had to close the pop-up window to continue interacting with the page.

Our HITs were restricted to workers with at least a 95% acceptance rate and workers within the U.S. Also, in order to avoid having a few workers complete most of our HITs, workers were not allowed to do more than 100 of our HITs. We collected spoken queries from 167 participants.

**Search Tasks.** Our 250 search tasks were based on the 250 topics from the TREC 2004 Robust Track. We constructed our tasks using the TREC description and narrative as guidelines and situated each task within a simulated scenario that gave rise to the need for information:

> **TREC Topic 390:** You recently read a news article about the Orphan Drug Act, which promotes the development of drugs to treat "orphan" diseases that affect only a small number of people. Now you are curious to learn more. Find information about the Orphan Drug Act and how it is working on behalf of people who suffer from rare diseases.

**ASR Systems.** In this work, we treat the ASR system as a "black box" and used two freely available APIs provided by AT&T and WIT.AI.[5] Both APIs accept a WAV file as input and return one or more candidate transcriptions in JSON format. The AT&T API was configured to return an n-best list in cases where the API was less confident about the input speech. The AT&T API returned an n-best list with at most 10 candidates along with their ranks and confidence values. The WIT.AI API could not be configured to return an n-best list and simply returned the single most confident transcription without a confidence value.

**Spoken Queries and ASR Output.** In this section, we describe our spoken queries and ASR output. To conserve space, we focus on the ASR output from the AT&T API. The AT&T API was able to transcribe 4,905 of our 5,000 spoken queries due to the quality of the recording. Spoken queries had an average length of $5.86 \pm 2.50$ seconds and $10.04 \pm 2.18$ recognized tokens. The AT&T API returned an n-best list with more than one candidate for 70% of the 4,905 transcribed spoken queries.

We were interested in measuring the variability between candidates from the same n-best list. To this end, we measured the similarity between candidate-pairs from the same n-best list in terms of their recognized tokens, top-10 documents retrieved, and retrieval performance. In terms of their recognized tokens, after stemming and stopping, candidate-pairs had an average Jaccard correlation of $0.53 \pm 0.23$. In terms of their top-10 documents retrieved, candidate-pairs had an average Jaccard correlation of $0.28 \pm 0.31$. Finally, in terms of retrieval performance, candidate-pairs had an average P@10 difference of $0.11 \pm 0.17$. More importantly, the most confident candidate achieved the best P@10 performance only 82.71% of the time. Together, these results suggest an opportunity to improve retrieval performance by re-ranking the n-best list.

We were also interested in measuring the variability between spoken queries from different study participants for the same TREC topic (using the most confident candidate from the ASR system's n-best list). In terms of their recognized

[5] `http://developer.att.com/apis/speech` and `https://wit.ai/`

tokens, spoken query-pairs had an average Jaccard correlation of $0.21 \pm 0.22$. In terms of their top-10 documents retrieved, the average Jaccard correlation was $0.12 \pm 0.23$. Finally, the average difference in P@10 performance was $0.19 \pm 0.23$. These measures suggest great variability in spoken query performance, either due to ASR error, background noise, or word choice. This helps motivate our second task of predicting when the input query is poor and the system should ask for a new spoken query.

## 4 Predictive Task Definitions

We investigate the effectiveness of existing query performance predictors on two tasks pertaining to speech-enabled search: (1) re-ranking the ASR system's n-best list and (2) predicting when to ask for a spoken query reformulation.

**Re-ranking the N-Best List.** While ASR systems often output the single most confident transcription, internally the system produces an n-best list of the most confident hypotheses. Off-the-shelf ASR systems such as the AT&T API can be configured to output the n-best list in cases where the system is less confident about the input speech.

We define the n-best list re-ranking task as follows. The input to the system is the spoken query's n-best list and the goal of the system is to predict the query transcription from the n-best that yields the greatest retrieval performance, which may not necessarily be the top candidate. The goal of the system is to maximize retrieval performance over a set of input n-best lists.

**Predicting When to Ask for a Spoken Query Reformulation.** In certain cases, a speech-enabled search engine may decide that the input spoken query is poor and may ask the user to reformulate the query. The input spoken query may be poor due to an ASR error or the user's word choice.

We define the spoken query reformulation task as follows. The input to the system is a spoken query (specifically, the top candidate from the ASR system's n-best list) and the goal of the system is to predict whether to show the results for the input query or to ask the user for a new spoken query. We assume that asking the user for a reformulation yields a more effective query, but incurs a cost. More formally, we assume that if the system decides to *not* ask for a reformulation, then the user experiences a gain equal to the retrieval performance of the original spoken query. Otherwise, if the system *does* decide to ask for a reformulation, then the user experiences a gain equal to the retrieval performance of the *new* query *discounted* by a factor denoted by $\alpha$ (in the range [0,1]). The system must decide whether to ask for a new spoken query without knowing the true performance of the original (e.g., using only pre- and post-retrieval performance predictors as evidence).

To illustrate, suppose that given an input spoken query, the system decides to ask for a reformulation. Furthermore, suppose that the original query achieves an average precision (AP) value of 0.15 and that the reformulated query achieves a AP value of 0.20. In this case, the user experiences a discounted gain of AP = $\alpha \times 0.20$. If we set $\alpha = 0.50$, then the discounted gain of the new query ($0.50 \times 0.20 = 0.10$) is less than the original (0.15), and so the system made the incorrect choice. Parameter $\alpha$ can be varied to simulate different costs of asking a user for

a spoken query reformulation. The higher the $\alpha$, the lower the cost. The goal of the system is to maximize the gain over a set of input spoken queries for a given value of $\alpha$.

## 5 Features

For both tasks, we used machine learning to combine different types of evidence as features. We grouped our features into three categories. The numbers in parentheses indicate the number of features in each category.

**N-best List Features (2).** These features were generated from the ASR system's n-best list. We included two n-best list features: the rank of the transcription in the n-best list and its confidence value. These features were only available for the AT&T API and only used in the n-best list re-ranking task.

**Pre-retrieval Features (27).** Prior work shows that a query is more likely to perform well if it contains discriminative terms that appear in only a few documents. We included five types of features aimed to capture this type of evidence. Our inverse document frequency (IDF) and inverse collection term frequency (ICTF) features measure the IDF and ICTF values across query terms [6, 2, 18]. We included the min, max, sum, average, and standard deviation of IDF and ICTF values across query terms. The *query-collection similarity* (QCS) score measures the extent to which the query terms appear many times in only a few documents [18]. We included the min, max, sum, average, and standard deviation of QCS values across query terms. The *query scope* score is inversely proportional to the number of documents with at least one query term [6]. Finally, the *simplified clarity* score measures the KL-divergence between the query and collection language models [6].

Prior work also shows that a query is more likely to perform well if the query terms describe a coherent topic. We included one type of feature to capture this type of evidence. Our point-wise mutual information (PMI) features measure the degree of co-occurrence between query terms [5]. We included the min, max, sum, average, and standard deviation of PMI values across query-term pairs.

Finally, a query is more likely to perform well if it produces a stable ranking. We included one type of feature to capture this type of evidence. We estimate the pre-retrieval rank stability by considering the query terms' variance of TF.IDF weights across the documents in the collection [18]. We included the min, max, average, sum, and standard deviation of the variance across query terms.

**Post-Retrieval Features (5).** A query is more likely to perform well if the top-ranked documents describe a coherent topic. We included three types of features to model this type of evidence. The *clarity score* measures that KL-divergence between the language model of the top documents and a background model of the collection [2]. The *query feedback* score measures the degree of overlap between the top-ranked documents before and after query-expansion [20]. A greater overlap suggests that the original query is on-topic. Finally, we consider the *normalized query commitment* (NQC) score, which measures the standard deviation of the top document scores. Following Shtok *et al.* [14], we included three NQC scores: the standard deviation of the top document scores, the standard deviation of the scores *above* the mean top-document score, and the standard deviation of the scores *below* the mean top-document score.

## 6 Evaluation Methodology

Retrieval performance was measured by issuing spoken query transcriptions against the TREC 2004 Robust Track collection. In all experiments, we used Lucene's implementation of the query-likelihood model with Dirichlet smoothing. Queries and documents were stemmed using the Krovetz stemmer and stopped using the SMART stopword list. We evaluated in terms of average precision (AP), NDCG@30, and P@10.

**Re-ranking the N-Best List.** We cast this as a learning-to-rank (LTR) task, and trained models to re-rank an n-best list in descending order of retrieval performance. At test time, we re-rank the input n-best list and select the top query transcription as the one to run against the collection. We used the linear RankSVM implementation in the Sophia-ML toolkit and trained separate models for each retrieval performance metric.

Models were evaluated using 20-fold cross-validation. Recall that each TREC topic had 20 spoken queries from different study participants. To avoid training and testing on n-best lists for the same TREC topic (potentially inflating performance), we assigned all n-best lists for the same topic to the same fold. We report average performance across held-out folds and measure statistical significance using the approximation of Fisher's randomization test described in Smucker *et al.* [15]. We used the same cross-validation folds in all our experiments. Thus, when testing significance, the randomization was applied to the 20 pairs of performance values for the two models being compared. We normalized feature values to zero-min and unit-max for each spoken query (i.e., using the min/max values from the same n-best list).

We compare against two baseline approaches: (1) selecting the best-performing candidate from the n-best list (oracle) and (2) selecting the top candidate with the highest recognition confidence (top).

**Predicting Spoken Query Reformulations.** We cast this as a binary classification task. The input to the system is a spoken query's most confident transcription, and the goal of the system is to predict whether to run the input query or to ask for a spoken query reformulation. If the system decides to run the input query, then the user experiences a gain equal to the retrieval performance of the original query. Otherwise, if the system decides to ask for a reformulation, then the user experiences a gain equal to the retrieval performance of the new spoken query *discounted* by $\alpha$.

We simulated the spoken query reformulation task as follows. Recall that each TREC topic had 20 spoken queries. For each topic, we used the top-performing spoken query to simulate the "reformulated" query and the remaining 19 spoken queries to simulate different inputs to the system. This produced $250 \times 19 = 4,750$ instances for training and testing.

While we cast this as a binary classification task, we decided to train a regression model to predict the difference between the performance of the input query and the discounted performance of the reformulated query. Our motivation was to place more emphasis on lower-performing training instances. At test time, we simply use the sign of the real-valued output to make a binary prediction. We used the linear SVM regression implementation in the LibLinear toolkit. We

trained different models for different evaluation metrics (AP, NDCG@30, P@10) and different values of $\alpha$. As in the n-best list re-ranking task, we evaluated using 20-fold cross-validation and assigned all spoken queries for the same TREC topic to the same fold. Similarly, we report average performance across held-out folds and measured statistical significance using Fisher's randomization test [15].

We compare against four different baselines: (1) always making the optimal choice between the input query and asking for a reformulation (oracle), (2) *always* asking for a reformulation (always), (3) *never* asking for a reformulation (never), and (4) asking for a reformulation randomly based on the training data probability that it is the optimal choice (random). The second and third baselines are expected to perform well for high values of $\alpha$ (low cost) and low values of $\alpha$ (high cost), respectively.

## 7 Results

Results for the n-best list re-ranking task are presented in Table 1. For this task, we used the n-best lists produced by the AT&T API. Furthermore, we focus on the subset of 3,414 (out of 5,000) spoken queries for which the AT&T API returned an n-best list with more than one transcription. The first and last rows in Table 1 correspond to our two baseline approaches: selecting the top-ranked candidate from the n-best list (top) and selecting the best-performing candidate for the corresponding metric (oracle). The middle rows correspond to the LTR model using all features (all), all features except for those in group x (no.x), and only those features in group x (only.x).

**Table 1.** Results for the n-best list re-ranking task. The percentages indicate percent improvement over top. A $^{\blacktriangle}$ denotes a significant improvement compared to top, and for no.x and only.x, a $^{\triangledown}$ denotes a significant performance drop compared to all. We used Bonferroni correction for multiple comparisons ($p < .05$).

| | AP | NDCG@30 | P@10 |
|---|---|---|---|
| top | 0.081 | 0.148 | 0.159 |
| all | 0.091 (13.75%)$^{\blacktriangle}$ | 0.162 (12.50%)$^{\blacktriangle}$ | 0.174 (12.26%)$^{\blacktriangle}$ |
| no.nbest | 0.090 (12.50%)$^{\blacktriangle\triangledown}$ | 0.162 (12.50%)$^{\blacktriangle}$ | 0.173 (11.61%)$^{\blacktriangle\triangledown}$ |
| no.pre | 0.089 (11.25%)$^{\blacktriangle\triangledown}$ | 0.155 (7.64%)$^{\blacktriangle\triangledown}$ | 0.166 (7.10%)$^{\blacktriangle\triangledown}$ |
| no.post | 0.085 (6.25%)$^{\blacktriangle\triangledown}$ | 0.154 (6.94%)$^{\blacktriangle\triangledown}$ | 0.168 (8.39%)$^{\blacktriangle\triangledown}$ |
| only.nbest | 0.080 (0.00%)$^{\triangledown}$ | 0.144 (0.00%)$^{\triangledown}$ | 0.155 (0.00%)$^{\triangledown}$ |
| only.pre | 0.084 (5.00%)$^{\blacktriangle\triangledown}$ | 0.154 (6.94%)$^{\blacktriangle\triangledown}$ | 0.167 (7.74%)$^{\blacktriangle\triangledown}$ |
| only.post | 0.089 (11.25%)$^{\blacktriangle\triangledown}$ | 0.155 (7.64%)$^{\blacktriangle\triangledown}$ | 0.166 (7.10%)$^{\blacktriangle\triangledown}$ |
| oracle | 0.102 (27.50%)$^{\blacktriangle}$ | 0.186 (29.17%)$^{\blacktriangle}$ | 0.205 (32.26%)$^{\blacktriangle}$ |

The results from Table 1 suggest five important trends. First, the LTR model using all features (all) significantly outperformed the baseline approach of always selecting the top-ranked transcription from the n-best list (top). The LTR model using all features had a greater than 10% improvement across all metrics.

Second, our results suggest that both pre- and post-retrieval query performance predictors contribute useful evidence for this task. The LTR model using only pre-retrieval features (only.pre) and only post-retrieval features (only.post) significantly outperformed the top baseline across all metrics. Furthermore, in

all cases, individually ignoring pre-retrieval features (no.pre) and post-retrieval features (no.post) resulted in a significant drop in performance compared to the LTR model using all features (all).

Third, there is some evidence that post-retrieval features were more predictive than pre-retrieval features. In terms of AP, ignoring post-retrieval features (no.post) and using *only* pre-retrieval features (only.pre) had the greatest performance drop compared to the model using all features (all). In terms of AP, post-retrieval features were more predictive in spite of having only 5 post-retrieval features versus 27 pre-retrieval features.

The fourth trend worth noting is that n-best list features contributed little useful evidence. In most cases, ignoring n-best list features (no.nbest) resulted in only a small drop in performance compared to the LTR model using all features (all). Furthermore, the LTR model using only n-best list features (only.nbest) was the worst-performing LTR model across all metrics and performed at the same level as the top baseline.

The final important trend is that there is still room for improvement. Across all metrics, the oracle performance was at least 25% greater than the top baseline. While not shown in Table 1, the oracle outperformed all the LTR models and the top baseline across all metrics by a statistically significant margin ($p < .05$).

Results for the task of predicting when to ask for a spoken query reformulation are shown in Tables 2 and 3. To conserve space, we only show results in terms of AP. However, the results in terms of NDCG@30 and P@10 had the same trends. We show results using the most confident transcriptions from the AT&T API (Table 2) and the WIT.AI API (Table 3). Because the WIT.AI API only returned the most confident transcription without a confidence value, we ignore n-best lists features in this analysis. Results are presented for different values of $\alpha$, with higher values indicating a higher cost of asking for a reformulation and therefore fewer cases where it was the correct choice. We show results for our four baselines (oracle, always, never, and random), as well as the regression model using all features (all), ignoring pre-retrieval features (no.pre), and ignoring post-retrieval features (no.post). The performance of never asking for a reformulation (never) is constant because it is independent of $\alpha$. The performance of always asking for a reformulation (always) increases with $\alpha$ (lower cost).

The results in Tables 2 and 3 suggest three important trends. First, the model using all features (all) performed equal to or better than always, never, and random for both APIs and all values of $\alpha$. The model performed at the same level as never for low values of $\alpha$ (high cost) and at the same level as always for high values of $\alpha$ (low cost). The model outpormed these three baselines for values of $\alpha$ in the mid-range ($0.4 \leq \alpha \leq 0.6$). For these values of $\alpha$, the system had to be more selective about when to ask for a reformulation. These results show that pre- and post-retrieval performance predictors provide useful evidence for predicting when the input spoken query is relatively poor.

Second, post-retrieval features were more predictive than pre-retrieval features. This is consistent with the AP results from Table 1. For values of $\alpha$ in the mid-range, ignoring post-retrieval (no.post) features resulted in a greater drop

in performance than ignoring pre-retrieval features (no.pre). The drop in performance was statistically significant for two values of $\alpha$ for the AT&T results and one value of $\alpha$ for the WIT.AI results. Again, we observed this trend in spite of having fewer post-retrieval than pre-retrieval features.

Finally, we note that there is room for improvement. For both APIs, the oracle baseline (oracle) outperformed the model using all features (all) across all values of $\alpha$. While not shown in Tables 2 and 3, all differences between oracle and all were statistically significant ($p < .05$).

**Table 2.** Results for predicting when to ask for a spoken query reformulation: AT&T API, Average Precision. A $^{\blacktriangle}$ denotes a significant improvement compared to always, never, and random. A $^{\triangledown}$ denotes a significant performance drop in for no.pre and no.post compared to all. We report significance for $p < .05$ using Bonferroni correction.

| discount ($\alpha$) | oracle | always | never | random | all | no.pre | no.post |
|---|---|---|---|---|---|---|---|
| 0.1 | 0.113 | 0.027 | 0.102 | 0.069 | 0.103 | 0.102 (-0.97%) | 0.102 (-0.97%) |
| 0.2 | 0.125 | 0.054 | 0.102 | 0.076 | 0.108$^{\blacktriangle}$ | 0.106 (-1.85%)$^{\blacktriangle\triangledown}$ | 0.105 (-2.78%)$^{\triangledown}$ |
| 0.3 | 0.138 | 0.082 | 0.102 | 0.092 | 0.122$^{\blacktriangle}$ | 0.119 (-2.46%)$^{\blacktriangle}$ | 0.113 (-7.38%)$^{\blacktriangle\triangledown}$ |
| 0.4 | 0.153 | 0.109 | 0.102 | 0.106 | 0.137$^{\blacktriangle}$ | 0.135 (-1.46%)$^{\blacktriangle}$ | 0.130 (-5.11%)$^{\blacktriangle}$ |
| 0.5 | 0.168 | 0.136 | 0.102 | 0.126 | 0.153$^{\blacktriangle}$ | 0.152 (-0.65%)$^{\blacktriangle}$ | 0.149 (-2.61%)$^{\blacktriangle}$ |
| 0.6 | 0.185 | 0.163 | 0.102 | 0.146 | 0.172$^{\blacktriangle}$ | 0.171 (-0.58%)$^{\blacktriangle}$ | 0.167 (-2.91%)$^{\blacktriangle}$ |
| 0.7 | 0.204 | 0.191 | 0.102 | 0.170 | 0.193 | 0.193 (0.00%) | 0.191 (-1.04%) |
| 0.8 | 0.224 | 0.218 | 0.102 | 0.198 | 0.218 | 0.218 (0.00%) | 0.218 (0.00%) |
| 0.9 | 0.247 | 0.245 | 0.102 | 0.231 | 0.245 | 0.245 (0.00%) | 0.245 (0.00%) |

**Table 3.** Results for predicting when to ask for a spoken query reformulation: WIT.AI API, Average Precision. Symbols $^{\blacktriangle}$ and $^{\triangledown}$ denote statistically significant differences as described in Table 2.

| discount ($\alpha$) | oracle | always | never | random | all | no.pre | no.post |
|---|---|---|---|---|---|---|---|
| 0.1 | 0.180 | 0.031 | 0.176 | 0.149 | 0.176 | 0.176 (0.00%) | 0.176 (0.00%) |
| 0.2 | 0.186 | 0.062 | 0.176 | 0.142 | 0.176 | 0.176 (0.00%) | 0.176 (0.00%) |
| 0.3 | 0.194 | 0.093 | 0.176 | 0.149 | 0.179 | 0.178 (-0.56%)$^{\blacktriangle}$ | 0.177 (-1.12%) |
| 0.4 | 0.203 | 0.124 | 0.176 | 0.154 | 0.185$^{\blacktriangle}$ | 0.182 (-1.62%) | 0.180 (-2.70%) |
| 0.5 | 0.214 | 0.156 | 0.176 | 0.165 | 0.196$^{\blacktriangle}$ | 0.192 (-2.04%)$^{\blacktriangle}$ | 0.188 (-4.08%)$^{\blacktriangle\triangledown}$ |
| 0.6 | 0.227 | 0.187 | 0.176 | 0.182 | 0.208$^{\blacktriangle}$ | 0.207 (-0.48%)$^{\blacktriangle}$ | 0.201 (-3.37%) |
| 0.7 | 0.243 | 0.218 | 0.176 | 0.203 | 0.225 | 0.224 (-0.44%) | 0.219 (-2.67%) |
| 0.8 | 0.261 | 0.249 | 0.176 | 0.229 | 0.250 | 0.250 (0.00%) | 0.247 (-1.20%) |
| 0.9 | 0.284 | 0.280 | 0.176 | 0.263 | 0.280 | 0.280 (0.00%) | 0.280 (0.00%) |

## 8 Discussion

Our results from Section 7 show that the top candidate from an ASR system's n-best list is not necessarily the best-performing query and that we can use query performance predictors to find a lower-ranked candidate that performs better. A reasonable question is: Why is the most confident candidate not always the best query? We examined n-best lists where a lower-ranked candidate outperformed the most confident, and encountered cases belonging to three categories.

In the first category, the lower-ranked candidate was a more accurate transcription of the input speech. For example, the lower-ranked candidate 'pro-

tect children poison paint' (AP = 0.467) outperformed the top candidate 'protect children poison pain' (AP = 0.055). Similarly, the lower-ranked candidate 'prostate cancer detect treat' (AP = 0.301) outperformed the top candidate 'press cancer detect treat' (AP = 0.014). Finally, the lower-ranked candidate 'drug treat alzheimer successful' (AP = 0.379) outperformed the top candidate 'drug treat timer successful' (AP = 0.001). We do not know why the ASR system assigned the correct transcription a lower probability. It may be that the correct query terms ('paint', 'prostate', 'alzheimer') had a lower probability in the ASR system's language model than those in the top candidates ('pain', 'press', 'timer'). Such errors might be reduced by using a language model from the target collection. However, this may not be possible with an off-the-shelf ASR system.

In the second category, the user mispronounced an important word associated with the task. In these cases, the top candidate was a better match with the input speech, but a lower-ranked candidate had the correct spelling of the word. For example, the lower-ranked candidate 'articles lives nobel prize winner' (AP = 0.289) outperformed the top candidate 'articles lives noble prize winner' (AP = 0.007). Here, the participant mispronounced 'nobel' as 'noble'. Similarly, the lower-ranked candidate 'welsh devolution movement' (AP = 0.460) outperformed the top candidate 'welsh deevolution movement' (AP = 0.050). In this case, the participant mispronounced 'devolution' as 'de-evolution'.

In the third category, none of the candidates were a perfect transcription of the input speech, but a lower-ranked candidate had an ASR error that was less important for the search task. For example, the lower-ranked candidate 'resend relations britain argentina' (AP = 0.443) outperformed the top candidate 'recent relations brandon argentina' (AP = 0.065). The top candidate had 'brandon' versus 'britain', while the lower-ranked candidate had 'resend' versus 'recent'. While both candidates had exactly one ASR error and similar confidence values, the error in the lower-ranked candidate yielded a more effective query for this task. In fact, one might argue that the lower-ranked candidate describes a more coherent topic, as indicated by its higher clarity score (3.180 versus 2.612). Cases in this category possibly arise when the ASR system is tuned to minimize word error rate [12], without explicitly favoring candidates that describe a coherent topic with respect to the target collection.

## 9 Conclusion

We developed and evaluated models for two tasks associated with speech-enabled search: (1) re-ranking the ASR system's n-best hypotheses and (2) predicting when to ask for a spoken query reformulation. Our results show that pre- and post-retrieval performance predictors contribute useful evidence for both tasks. With respect to the first task, our analysis shows that lower-ranked candidates in the n-best list may perform better due to mispronunciation errors in the input speech or because the ASR system may not explicitly favor candidates that describe a coherent topic with respect to the target collection.

There are several directions for future work. In this work, we improved the input query by exploring candidates in the same n-best list. Future work might consider exploring a larger space, including reformations of the top candidate that are specifically designed for the speech domain (e.g., term substitutions

with similar Soundex codes). Additionally, in this work, we predicted when to ask for a new spoken query. Future work might consider learning to ask more targeted clarification or disambiguation questions about the input spoken query.

**Acknowledgments.** This work was supported in part by NSF grant IIS-1451668. Any opinions, findings, conclusions, and recommendations expressed in this paper are the authors and do not necessarily reflect those of the sponsors.

## References

1. J. A. Aslam and V. Pavlu. Query hardness estimation using jensen-shannon divergence among multiple scoring functions. In *ECIR*, 2007.
2. S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR*, 2002.
3. V. Dang, M. Bendersky, and W. B. Croft. Learning to rank query reformulations. In *SIGIR*, 2010.
4. F. Diaz. Performance prediction using spatial autocorrelation. In *SIGIR*, 2007.
5. C. Hauff. *Predicting the Effectiveness of Queries and Retrieval Systems.* dissertation, Univeristy of Twente, 2010.
6. B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *SPIRE*, 2004.
7. J. Jiang, W. Jeng, and D. He. How do users respond to voice input errors?: Lexical and phonetic query reformulation in voice search. In *SIGIR*, 2013.
8. G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *SIGIR*, 2009.
9. X. Li, P. Nguyen, G. Zweig, and D. Bohus. Leveraging multiple query logs to improve language models for spoken query recognition. In *ICASSP*, 2009.
10. J. Mamou, A. Sethy, B. Ramabhadran, R. Hoory, and P. Vozila. Improved spoken query transcription using co-occurrence information. In *INTERSPEECH*, 2011.
11. F. Peng, S. Roy, B. Shahshahani, and F. Beaufays. Search results based n-best hypothesis rescoring with maximum entropy classification. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.
12. J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope. Your word is my command: Google search by voice: A case study. In *Advances in Speech Recognition*. 2010.
13. D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. Lambdamerge: Merging the results of query reformulations. In *WSDM*, 2011.
14. A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits. Predicting query performance by query-drift estimation. *TOIS*, 30(2), 2012.
15. M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, 2007.
16. X. Xue, S. Huston, and W. B. Croft. Improving verbose queries using subset distribution. In *CIKM*, 2010.
17. E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: Including applications to missing content detection and distributed information retrieval. In *SIGIR*, 2005.
18. Y. Zhao, F. Scholer, and Y. Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *ECIR*, 2008.
19. Y. Zhou and W. B. Croft. Ranking robustness: A novel framework to predict query performance. In *CIKM*, 2006.
20. Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *SIGIR*, 2007.