Learning to Re-rank for Interactive Problem Resolution and Query Refinement

Rashmi Gangadharaiah
IBM Research,
India Research Lab,
Bangalore, KA, India
rashgang@in.ibm.com

Balakrishnan Narayanaswamy and Charles Elkan
Department of CSE,
University of California, San Diego
La Jolla, CA, USA
{muralib, elkan}@cs.ucsd.edu

Abstract

We study the design of an information retrieval (IR) system that assists customer service agents while they interact with end-users. The type of IR needed is difficult because of the large lexical gap between problems as described by customers, and solutions. We describe an approach that bridges this lexical gap by learning semantic relatedness using tensor representations. Queries that are short and vague, which are common in practice, result in a large number of documents being retrieved, and a high cognitive load for customer service agents. We show how to reduce this burden by providing suggestions that are selected based on the learned measures of semantic relatedness. Experiments show that the approach offers substantial benefit compared to the use of standard lexical similarity.

1 Introduction

Information retrieval systems help businesses and individuals make decisions by automatically extracting actionable intelligence from large (unstructured) data (Musen et al., 2006; Antonio Palma-dos Reis, 1999). This paper focuses on the application of retrieval systems in a contact centers where the system assists agents while they are helping customers with problem resolution.

Currently, most contact center information retrieval use (web based) front-ends to search engines indexed with knowledge sources (Holland, 2005). Agents enter queries to retrieve documents related to the customer's problem. These sources are often incomplete as it is unlikely that all possible customer problems can be identified before product release. This is particularly true for recently released and frequently updated products. One approach, which we build on here, is to mine problems and resolutions from online discussion forums Yahoo! Answers[1] Ubuntu Forums[2] and Apple Support Communities[3]. While these often provide useful solutions within hours or days of a problem surfacing, they are semantically noisy (Gangadharaiah and Narayanaswamy, 2013).

Most contact centers and agents are evaluated based on the number of calls they handle over a period (Pinedo et al., 2000). As a result, queries entered by agents into the search engine are usually underspecified. This, together with noise in the database, results in a large number of documents being retrieved as relevant documents. This in turn, increases the cognitive load on agents, and reduces the effectiveness of the search system and the efficiency of the contact center. Our first task in this paper is to automatically make candidate suggestions that reduce the search space of relevant documents in a contact center application. The agent/user then interacts with the system by selecting one of the suggestions. This is used to expand the original query and the process can be repeated. We show that even one round of interaction, with a small set of suggestions, can lead to high quality solutions to user problems.

In query expansion, the classical approach is to automatically find suggestions either in the form of words, phrases or similar queries (Kelly et al., 2009; Feuer et al., 2007; Leung et al., 2008). These can be obtained either from query logs or based on their representativeness of the initial retrieved documents (Guo et al., 2008; Baeza-yates et al., 2004). The suggestions are then ranked either based on their frequencies or based on their similarity to the original query (Kelly et al., 2009; Leung et al., 2008). For example, if suggestions and queries are represented as term vectors (e.g.

[1] http://answers.yahoo.com/
[2] http://ubuntuforums.org/
[3] https://discussions.apple.com/

Proceedings of the SIGDIAL 2014 Conference, pages 218–227,
Philadelphia, U.S.A., 18-20 June 2014.

term frequency-inverse document frequency or tf-idf) their similarity may be determined using similarity measures such as cosine similarity or inverse of euclidean distance (Salton and McGill, 1983).

However, in question-answering and problem-resolution domains, and in contrast to traditional Information Retrieval, most often the query and the suggestions do not have many overlapping words. This leads to low similarity scores, even when the suggestion is highly relevant. Consider the representative example in Table 1, taken from our crawled dataset. Although the suggestions, "does not support file transfer", "connection not stable", "pairing failed" are highly relevant for the problem of "Bluetooth not working", their lexical similarity score is zero. The second task that this paper addresses is how to bridge this lexical chasm between the query and the suggestions. For this, we learn a measure of semantic-relatedness between the query and the suggestions rather than defining closeness based on lexical similarity.

Query	Bluetooth not working .
Suggestions	devices not discovered, bluetooth greyed out, bluetooth device did not respond, does not support file transfer, connection not stable, pairing failed

Table 1: Suggestions for the Query or customer's problem, "Bluetooth not working".

The primary contributions of this paper are that:

- We show how tensor methods can be used to learn measures of question-answer or problem-resolution similarity. In addition, we show that these learned measures can be used directly with well studied classification techniques like Support Vector Machines (SVMs) and Logistic Classifiers to classify whether suggestions are relevant. This results in substantially improved performance over using conventional similarity metrics.

- We show that along with the learned similarity metric, a data dependent Information Gain (which incorporates knowledge about the set of documents in the database) can be used as a feature to further boost accuracy.

- We demonstrate the efficacy of our approach on a complete end-to-end problem-resolution system, which includes crawled data from online forums and gold standard user interaction annotations.

2 System outline

As discussed in the Introduction, online discussion forums form a rich source of problems and their corresponding resolutions. Thread initiators or users of a product facing problems with their product post in these forums. Other users post possible solutions to the problem. At the same time, there is noise due to unstructured content, off-topic replies and other factors. Our interaction system has two phases, as shown in Figure 1. The offline phase attempts to reduce noise in the database, while the online phase assists users deal with the cognitive overload caused by a large set of retrieved documents. In our paper, threads form the documents indexed by the system.

The goals of the **offline phase** are two-fold. First, to reduce the aforementioned noise in the database, we succinctly represent each document (i.e., a thread in online discussion forums) by its *signature*, which is composed of *units* extracted from the first post of the underlying thread that best describe the problem discussed in the thread. Second, the system makes use of click-through data, where users clicked on relevant suggestions for their queries to build a relevancy model. As mentioned before, the primary challenge is to build a model that can identify units that are semantically similar to a given query.

In the **online phase**, the agent who acts as the mediator between the user and the Search Engine enters the user's/customer's query to retrieve relevant documents. From these retrieved documents, the system then obtains candidate suggestions and ranks these suggestions using the relevancy model built in the offline phase to further better understand the query and thereby reduce the space of documents retrieved. The user then selects the suggestion that is most relevant to his query. The retrieved documents are then filtered displaying only those documents that contain the selected suggestion in their signatures. The process continues until the user quits.

2.1 Signatures of documents

In the offline phase, every document (corresponding to a thread in online discussion forums) is represented by units that best describe a problem. We adopt the approach suggested in (Gangadhara-

iah and Narayanaswamy, 2013) to automatically generate these signatures from each discussion thread. We assume that the first post describes the user's problem, something we have found to be true in practice. From the dependency parse trees of the first posts, we extract three types of units (i) phrases (e.g., sync server), (ii) attribute-values (e.g., iOS, 4) and (iii) action-attribute tuples (e.g., sync server: failed). Phrases form good base problem descriptors. Attribute-value pairs provide configurational contexts to the problem. Action-attribute tuples, as suggested in (Gangadharaiah and Narayanaswamy, 2013), capture segments of the first post that indicate user wanting to perform an action ("I cannot hear notifications on bluetooth") or the problems caused by a user's action ("working great before I updated"). These make them particularly valuable features for problem-resolution and question-answering.

2.2 Representation of Queries and Suggestions

Queries are represented as term vectors using the term frequency-inverse document frequency (tf-idf) representation forming the query space. The term frequency is defined as the frequency with which word appears in the query and the inverse document frequency for a word is defined as the frequency of queries in which the word appeared. Similarly, units are represented as tf-idf term vectors from the suggestion space. Term frequency in the unit space is defined as the number of times a word appears in the unit and its inverse document frequency is defined in terms of the number of units in which the word appeared. Since the vocabulary used in the queries and documents are different, the representations for queries and units belong to different spaces of different dimensions.

For every query-unit pair, we learn a measure of similarity as explained in Section 4. Additionally, we use similarity features based on cosine similarity between the query and the unit under consideration. We also consider an additional feature based on information gain (Gangadharaiah and Narayanaswamy, 2013). In particular, if S represents the set all retrieved documents, S_1 is a subset of S ($S_1 \subseteq S$) containing a unit $unit_i$ and S_2 is a subset of S that does not contain $unit_i$, information gain with $unit_i$ is,

$$Gain(S, unit_i) = E(S) - \frac{|S_1|}{|S|}E(S_1) - \frac{|S_2|}{|S|}E(S_2) \quad (1)$$

$$E(S) = \sum_{k=1,\ldots|S|} -p(doc_k)log_2p(doc_k). \quad (2)$$

The probability for each document is based on its rank in the retrieved of results,

$$p(doc_j) = \frac{\frac{1}{rank(doc_j)}}{\sum_{k=1,\ldots|S|}\frac{1}{rank(doc_k)}}. \quad (3)$$

We crawled posts and threads from online forums for the products of interest, as detailed in Section 5.1, and these form the documents. We used trial interactions and retrievals to collect the click-though data, which we used as labeled data for similarity metric learning. In particular, labels indicate which candidate units were selected as relevant suggestions by a human annotator. We now explain our training (offline) and testing (online) phases that use this data in more detail.

2.3 Training

The labeled (click-through) data for training the relevance model is collected as follows. Annotators were given pairs of queries. Each pair is composed of an *underspecified* query and a *specific* query (Section 5.1 provides more information on the creation of these queries). An underspecified query is a query that reflects what a user/agent typically enters into the system, and the corresponding specific query is full-specified version of the underspecified query. Annotators were first asked to query the search engine with each underspecified query. We use the Lemur search engine (Strohman et al., 2004). From the resulting set of retrieved documents, the system uses the information gain criteria (as given in (1) below) to rank and display to the annotators the candidate suggestions (i.e., the units that appear in the signatures of the retrieved documents). Thus, our system is bootstrapped using the information gain criterion. The annotators then selects the candidate suggestion that is most relevant to the corresponding specific query. The interaction with the system continues until the annotators quit.

We then provide a class label for each unit based on the collected click-through information. In particular, if a unit $s \in \mathcal{S}(x)$ was clicked by a user for his query x, from the list $\mathcal{S}$ we provide a $+$ label to indicate that the unit is relevant suggestion for the query. Similarly, for all other units that are never clicked by users for x are labeled as $-$. This forms the training data for the system. Details on

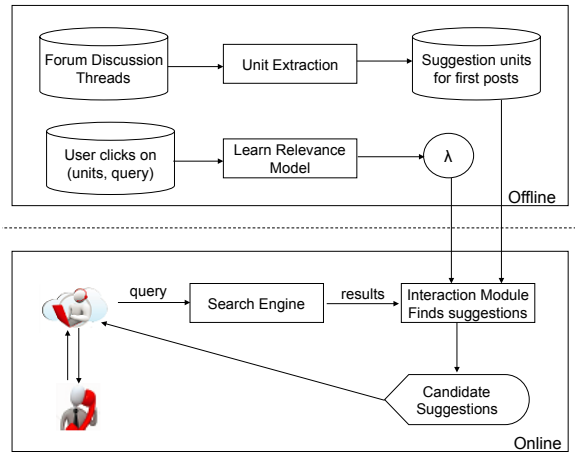

Figure 1: Outline of our interactive query refinement system for problem resolution

the feature extraction and how the model is created is given in Section 3.

2.4 Testing

In the online phase, the search engine retrieves documents for the user's query x'. Signatures for the retrieved documents form the initial space of candidate units. As done in training, for every pair of x' and unit the label is predicted using the model built in the training phase. Units that are predicted as $+$ are shown to the user. When a user clicks on his most relevant suggestion, the retrieved results are filtered to show only those documents that contain the suggestion (i.e., in its signature). This process continues until the user quits.

3 Model

We consider underspecified queries $x \in \mathbb{R}^{x_d}$ and units $y \in \mathbb{R}^{y_d}$. Given an underspecified query x we pass it through a search engine, resulting in a list of results $\mathcal{S}(x)$.

As explained in Section 2.3, our training data consists of labels $r(x, y) \in +1, -1$ for each under-specified query, $y \in \mathcal{S}(x)$. $r(x, y) = +1$ if the unit is labeled a relevant suggestion and $r(x, y) = -1$ if it is not labeled relevant. Units are relevant or not based on the final query, and not just y, a distinction we expand upon below.

At each time step, our system proposes a list $\mathcal{Z}(x)$ of possible query refinement suggestions z to the user. The user can select one or none of these suggestions. If the user selects z, only those documents that contain the suggestion (i.e., in its signature) are shown to the user, resulting in a filtered set of results, $\mathcal{S}(x + z)$.

This process can be repeated until a stopping criterion is reached. Stopping criterion include the size of the returned list is smaller than some number $|\mathcal{S}(x + z)| < N$, in which case all remaining documents are returned. Special cases include when only one document is returned $N = 1$. We will design query suggestions so that $|\mathcal{S}(x+z)| > 0$. Another criterion we use is to return all remaining documents after a certain maximum number of interactions or until the user quits.

4 Our Approach

We specify our algorithm using a tensor notation. We do this since tensors appear to subsume most of the methods applied in practice, where different algorithms use slightly different costs, losses and constraints. These ideas are strongly motivated by, but generalize to some extent, suggestions for this problem presented in (Elkan, 2010).

For our purposes, we consider tensors as multi-dimensional arrays, with the number of dimensions defined as the order of the tensor. An M order tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \ldots I_M}$. As such tensors subsume vectors (1st order tensors) and matrices (2nd order tensors). The vectorization of a tensor of order M is obtained by stacking elements from the M dimensions into a vector of length $I_1 \times I_2 \times \ldots \times I_M$ in the natural way.

The inner product of two tensors is defined as

$$\langle \mathbf{X}, \mathbf{W} \rangle = \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \ldots \sum_{i_M}^{I_M} x_{i_1} w_{i_1} x_{i_2} w_{i_2} \ldots x_{i_M} w_{i_M} \tag{4}$$

Analogous to the definition for vectors, the (Kharti-Rao) outer product $\mathbf{A} = \mathbf{X} \otimes \mathbf{W}$ of two tensors $\mathbf{X}$ and $\mathbf{W}$ has $A_{ij} = X_i W_j$ where i and j run over all elements of $\mathbf{X}$ and W. Thus, if $\mathbf{X}$ is of order M_X and $\mathbf{W}$ of order M_W, $\mathbf{A}$ is of order $M_A = M_X + M_W$.

The particular tensor we are interested in is a 2-D tensor (matrix) $\mathbf{X}$ which is the outer product of query and unit pairs (Feats). In particular, for a query x and unit y, $X_{i,j} = x_i y_j$.

Given this representation, standard classification and regression methods from the machine learning literature can often be extended to deal with tensors. In our work we consider two classifiers that have been successful in many applications, logistic regression and support vector machines (SVMs) (Bishop, 2006).

In the case of logistic regression, the conditional probability of a reward signal $r(\mathbf{X}) = r(x, y)$ is,

$$p(r(\mathbf{X}) = +1) = \frac{1}{1 + exp(-\langle \mathbf{X}, \mathbf{W} \rangle + b)} \quad (5)$$

The parameters $\mathbf{W}$ and b can be obtained by minimizing the log loss $\mathcal{L}_{reg}$ on the training data $\mathcal{D}$

$$\mathcal{L}_{reg}(\mathbf{W}, b) = \quad (6)$$
$$\sum_{(\mathbf{X}, r(\mathbf{X})) \in \mathcal{D}} \log(1 + exp(-r(\mathbf{X})\langle \mathbf{X}, \mathbf{W} \rangle + b)$$

For SVMs with the hinge loss we select parameters to minimize $\mathcal{L}_{hinge}$,

$$\mathcal{L}_{hinge}(\mathbf{W}, b) = ||\mathbf{X}||_F^2 + \quad (7)$$
$$\lambda \sum_{(\mathbf{X}, r(\mathbf{X})) \in \mathcal{D}} \max[0, 1 - (r(\mathbf{X})\langle \mathbf{X}, \mathbf{W} \rangle + b)]$$

where $||\mathbf{X}||_F$ is the Frobenius norm of tensor $\mathbf{X}$.

Given the number of parameters in our system $(\mathbf{W}, b)$ to limit overfitting, we have to regularize these parameters. We use regularizers of the form

$$\Omega(\mathbf{W}, b) = \lambda_W ||\mathbf{W}||_F \quad (8)$$

such regularizes have been successful in many large scale machine learning tasks including learning of high dimensional graphical models (Ravikumar et al., 2010) and link prediction (Menon and Elkan, 2011).

Thus, the final optimization problem we are faced with is of the form

$$\min_{\mathbf{W}, b} \mathcal{L}(\mathbf{W}, b) + \Omega(\mathbf{W}, b) \quad (9)$$

where $\mathcal{L}$ is $\mathcal{L}_{reg}$ or $\mathcal{L}_{hinge}$ as appropriate. Other losses, classifiers and regularizers may be used.

The advantage of tensors over their vectorized counterparts, that may be lost in the notation, is that they do not lose the information that the different dimensions can (and in our case do) lie in different spaces. In particular, in our case we use different features to represent queries and units (as discussed in Section 2.2) which are not of the same length, and as a result trivially do not lie in the same space.

Tensor methods also allow us to regularize the components of queries and units separately in different ways. This can be done for example by, i) forcing $\mathbf{W} = \mathbf{Q}_1\mathbf{Q}_2$, where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are constrained to be of fixed rank s ii) using trace or Frobenius norms on Q_1 and Q_2 for separate regularization as proxies for the rank iii) using different sparsity promoting norms on the rows of Q_1 and Q_2 iv) weighing these penalties differently for the two matrices in the final loss function. Note that by analogy to the vector case, we directly obtain generalization error guarantees for our methods.

We also discuss the advantage of the tensor representation above over a natural representation $\mathbf{X} = [x; y]$ i.e. $\mathbf{X}$ is the column vector obtained by stacking the query and unit representations. Note that in this representation, for logistic regression, while a change in the query x can change the probability for a unit $P(r(\mathbf{X}) = 1)$ it cannot change the relative probability of two different units. Thus, the ordering of all unit remains the same for all queries. This flaw has been pointed out in the literature in (Vert and Jacob, 2008) and (Bai et al., 2009), but was brought to our attention by (Elkan, 2010).

Finally, we note that by normalizing the query and unit vectors (x and y), and selecting $W = I$ (the identity matrix) we can recover the cosine similarity metric (Elkan, 2010). Thus, our representation is atleast as accurate and we show that learning the diagonal and off-diagonal components of $\mathbf{W}$ can substantially improve accuracy.

Additionally, for every (query,unit) we also compute information gain (IG) as given in (1), and the lexical similarity (Sim) in terms of cosine similarity between the query and the unit as additional features in the feature vectors.

5 Results and Discussion

To evaluate our system, we built and simulated a contact center information retrieval system for iPhone problem resolution.

5.1 Description of the Dataset

We collected data by crawling forum discussion threads from the Apple Discussion Forum, created during the period 2007-2011, resulting in about 147,000 discussion threads. The underspecified queries and specific queries were created as follows. Discussion threads were first clustered treating each discussion thread as a data point using a tf-idf representation. The thread nearest the centroid of the 60 largest clusters were marked as the 'most common' problems.

The first post is used as a proxy for the problem description. An annotator was asked to then create

Underspecified query "Safari not working"
1. safari:crashes
2. safari:cannot find:server
3. server:stopped responding
4. phone:freezes
5. update:failed

Table 2: Specific Queries generated with the underspecified Query, "Safari not working".

a short query (underspecified) from the first post of each of the 60 selected threads. These queries were given to the Lemur search engine (Strohman et al., 2004) to retrieve the 50 most similar threads from an index built on the entire set of 147,000 threads. The annotator manually analyzed the first posts of the retrieved threads to create contexts, resulting in a total 200 specific queries.

We give an example to illustrate the data creation in Table 2. From an under-specified query "Safari not working", the annotator found 5 specific queries. Two other annotators, were given these specific queries with the search engine's results from the corresponding under-specified query. They were asked to choose the most relevant results for the specific queries. The intersection of the choices of the annotators formed our 'gold standard' of relevant documents.

5.2 Results

We simulated a contact center retrieval systems (as in Figure 1) to evaluate the approach proposed in this paper. To evaluate the generality of our approach we conduct experiments with both SVMs and Logistic Regression. Due to lack of space we illustrate each result for only one kind of classifier.

5.2.1 Evaluating the Relevance Model

To measure the performance of the relevance model for predicting the class labels or for finding the most relevant units towards making the user's underspecified query more specific, we performed the following experiment. 4000 random query-unit pairs were picked from the training data, collected as explained in Section 2. Since most units are not relevant for a query, 90% of the pairs belonged to the $-$ class. On average, every specific query gave rise to 2.4 suggestions. Hence, predicting $-$ for all pairs still achieves an error rate of 10%. This data was then split into varying sizes of training and test sets. The relevancy model was then built on the training half and the classifiers were used to predict labels on the test

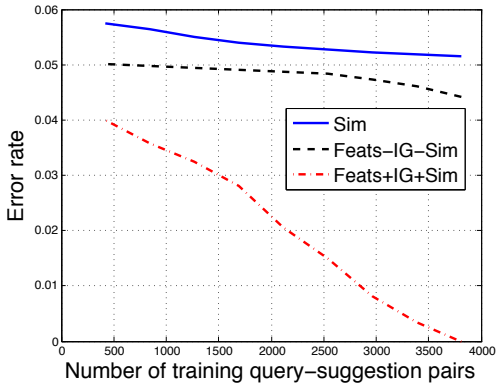

Figure 2: Performance with Logistic Regression using different features and various sizes of Training and Test sets. Feats-IG-Sim does not use cosine similarity (Sim) and information gain (IG). Feats+IG+Sim considers Sim and IG.

set. Figure 2 shows error rate obtained with logistic regression (a similar trend was observed with SVMs) on various sizes of the training data and test data. The plot shows that the model (Feats-IG-Sim and Feats+IG+Sim) performs significantly better at predicting the relevancy of units for underspecified queries when compared to just using cosine similarity (Sim) as a feature. Feats-IG-Sim does not make use of cosine similarity as a feature or the information gain feature while Feats+IG+Sim uses both these features for training the relevancy model and for predicting the relevancy of units. As expected the performance of the classifier improves as the size of the training data is increased.

5.2.2 Evaluating the Interaction Engine

We evaluate a complete system with both the user (the agent) and the search engine in the loop. We measure the value of the interactions by an analysis of which results 'rise to the top'. Users were given a specific query and its underspecified query along with the results obtained when the underspecified query was input to the search engine. They were presented with suggestions that were predicted $+$ for the underspecified query using SVMs. The user was asked to select the most appropriate suggestion that made the underspecified query more specific. This process continues until the user quits either because he is satisfied with the retrieved results or does not obtain relevant suggestions from the system. For example, for the underspecified query in Table 2, one of the predicted suggestions was, "server:stopped respond-

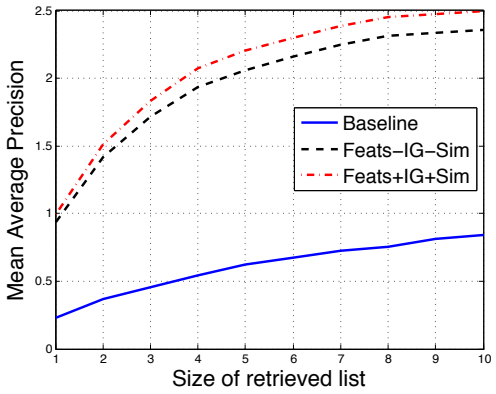

Figure 3: Comparison of the proposed approach with respect to the Baseline that does not involve interaction in terms of MAP at N.

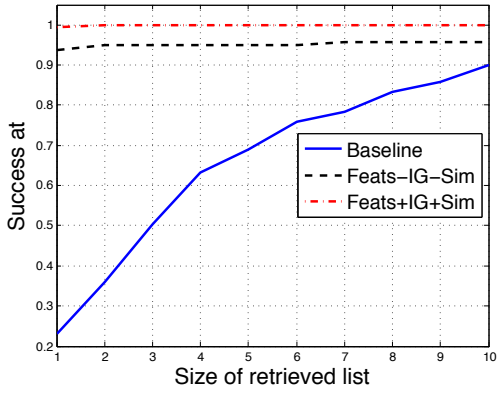

Figure 4: Comparison of the proposed approach with respect to the Baseline that does not involve interaction in terms of Success at N.

ing". If the user finds the suggestion relevant, he clicks on it. The selected suggestion then reduces the number of retrieved results. We then measured the relevance of the reduced result, with respect to the gold standard for that specific query, using metrics used in IR - MRR, Mean Average Precision (MAP) and Success at rank N.

Figures 3, 4 and Table 3 evaluate the results obtained with the interaction engine using Feats-IG-Sim and Feats+IG+Sim. We compared the performance of our algorithms with a Baseline that does not perform any interaction and is evaluated based on the retrieved results obtained with the underspecified queries. The models for each of the systems were trained using query-suggestion pairs collected from 100 specific queries (data collected as explained in Section 2). The remaining 100 specific queries were used for testing. We see that the suggestions predicted by the classifiers using the relevancy model indeed improves the performance of the baseline. Also, adding the IG and Sim feature further boosts the performance of the system.

Systems	MRR
Baseline	0.4218
Feats-IG-Sim	0.9449
Feats+IG+Sim	0.9968

Table 3: Comparison of the proposed approach with respect to the Baseline that does not involve interaction in terms of MRR.

5.3 Related Work

Learning affinities between queries an documents is a well studied area. (Liu, 2009) provides an excellent survey of these approaches. In these methods, there is a fixed feature function $\Phi(x, y)$ defined between any query-document pair. These features are then used, along with labeled training data, to learn the parameters of a model that can then be used to predict the relevance $r(x, y)$ of a new query-document pair. The output of the model can also be used to re-rank the results of a search engine. In contrast to this class of methods, we define and parameterize the Φ function and jointly optimize the parameters of the feature mapping and the machine learning re-ranking model.

Latent tensor methods for regression and classification have recently become popular in the image and signal processing domain. Most of these methods solve an optimization problem similar to our own (9), but add additional constraints limiting the rank of the learned matrix W either explicitly or implicit by defining $W = Q_1 Q_2^T$, and defining $Q_1 \in \mathbb{R}^{d_x \times d}$ and $Q_2 \in \mathbb{R}^{d_y \times d}$. This approach is used for example in (Pirsiavash et al., 2009) and more recently in (Tan et al., 2013) (Guo et al., 2012). While this reduces the number of parameters to be learned from $d_x d_y$ to $d(d_x + d_y)$ it makes the problem non-convex and introduces an additional parameter d that must be selected.

This approach of restricting the rank was recently suggested for information retrieval in (Wu et al., 2013). They look at a regression problem, using click-through rates as the reward function $r(x, y)$. In addition, (Wu et al., 2013) does not use an initial search engine and hence must learn an affinity function between all query-document pairs. In contrast to this, we learn a classification function that discriminates between the true and false positive documents that are deemed similar

by the search engine. This has three beneficial effects : (i) it reduces the amount of labeled training data required and the imbalance between the positive and negative classes which can make learning difficult (He and Garcia, 2009) and (ii) allows us to build on the strengths of fast and strong existing search engines increasing accuracy and decreasing retrieval time and (iii) allows the learnt model to focus learning on the query-document pairs that are most problematic for the search engine.

Bilinear forms of tensor models without the rank restriction have recently been studied for link prediction (Menon and Elkan, 2011) and image processing (Kobayashi and Otsu, 2012). Since the applications are different, there is no preliminary search engine which retrieves results, making them ranking methods and ours a re-ranking approach. Related work in text IR includes (Beeferman and Berger, 2000), where two queries are considered semantically similar if their clicks lead to the same page. However, the probability that different queries lead to common clicks of the same URLs is very small, again increasing the training data required. Approaches in the past have also proposed techniques to automatically find suggestions either in the form of words, phrases (Kelly et al., 2009; Feuer et al., 2007; Baeza-yates et al., 2004) or similar queries (Leung et al., 2008) from query logs (Guo et al., 2008; Baeza-yates et al., 2004) or based on their probability of representing the initial retrieved documents (Kelly et al., 2009; Feuer et al., 2007). These suggestions are then ranked either based on their frequencies or based on their closeness to the query. Closeness is defined in terms of lexical similarity to the query. However, most often the query and the suggestions do not have any co-occurring words leading to low similarity scores, even when the suggestion is relevant.

(Gangadharaiah and Narayanaswamy, 2013) use information gain to rank candidate suggestions. However, the relevancy of the suggestions highly depends on the relevancy of the initial retrieved documents. Our work here addresses the question of how to bridge this lexical chasm between the query and the suggestions. For this, we use semantic-relatedness between the query and the suggestions as a measure of closeness rather than defining closeness based on lexical similarity. A related approach to handle this lexical gap by applying alignment techniques from Statistical Machine translation (Brown et al., 1993), in particular by building translation models for information retrieval (Berger and Lafferty, 1999; Riezler et al., 2007). These approaches require training data in the form of question-answer pairs, are again limited to words or phrases and are not intended for understanding the user's problem better through interaction, which is our focus.

6 Conclusions, Discussions and Future Work

We studied the problem of designing Information Retrieval systems for interactive problem resolution. We developed a system for bridging the large lexical gap between short, incomplete problem queries and documents in a database of resolutions. We showed that tensor representations are a useful tool to learn measures of semantic relatedness, beyond the cosine similarity metric. Our results show that with interaction, suggestions can be effective in pruning large sets of retrieved documents. We showed that our approach offers substantial improvement over systems that only use lexical similarities for retrieval and re-ranking, in an end-to-end problem-resolution domain.

In addition to the classification losses considered in this paper, we can also use another loss term based on ideas from recommender systems, in particular (Menon and Elkan, 2011). Consider the matrix $\mathbf{T}$ with all training queries as rows and all units as the columns. If we view the query refinement problem as a matrix completion problem, it is natural to assume that this matrix has low rank, so that $\mathbf{T}$ can be written as $\mathbf{T} = \mathbf{U}\Lambda\mathbf{V}^T$, where Λ is a diagonal matrix and parameter of our optimization. These can then be incorporated into the training process by appropriate changes to the cost and regularization terms.

Another benefit of the tensor representation is that it can easily be extended to incorporate other meta-information that may be available. For example, if context sensitive features, like the identity of the agent, are available these can be incorporated as another dimension in the tensor. While optimization over these higher dimensional tensors may be more computationally complex, the problems are still convex and can be solved efficiently. This is a direction of future research we are pursuing. Finally, exploring the power of information gain type features in larger database systems is of interest.

References

Fatemeh Zahedi Antonio Palma-dos Reis. 1999. Designing personalized intelligent financial decision support systems.

Ricardo Baeza-yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *In International Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT), Creete*, pages 588–596. Springer.

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Corinna Cortes, and Mehryar Mohri. 2009. Polynomial semantic indexing. In *NIPS*, pages 64–72.

Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 407–416, New York, NY, USA. ACM.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 222–229, New York, NY, USA. ACM.

Christopher M Bishop. 2006. Pattern recognition and machine learning.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Charles Elkan. 2010. Learning affinity with biliear models. *Unpublished Notes*.

Alan Feuer, Stefan Savev, and Javed A. Aslam. 2007. Evaluation of phrasal query suggestions. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 841–848, New York, NY, USA. ACM.

Rashmi Gangadharaiah and Balakrishnan Narayanaswamy. 2013. Natural language query refinement for problem resolution from crowd-sourced semi-structured data. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 243–251, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. 2008. A unified and discriminative model for query refinement. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *SIGIR*, pages 379–386. ACM.

Weiwei Guo, Irene Kotsia, and Ioannis Patras. 2012. Tensor learning for regression. *Image Processing, IEEE Transactions on*, 21(2):816–827.

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

Alexander Holland. 2005. Modeling uncertainty in decision support systems for customer call center. In *Computational Intelligence, Theory and Applications*, pages 763–770. Springer.

Diane Kelly, Karl Gyllstrom, and Earl W. Bailey. 2009. A comparison of query and term suggestion features for interactive searching. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 371–378, New York, NY, USA. ACM.

Takumi Kobayashi and Nobuyuki Otsu. 2012. Efficient optimization for low-rank integrated bilinear classifiers. In *Computer Vision–ECCV 2012*, pages 474–487. Springer.

Kenneth Wai-Ting Leung, Wilfred Ng, and Dik Lun Lee. 2008. Personalized concept-based clustering of search engine queries. *IEEE Trans. on Knowl. and Data Eng.*, 20(11):1505–1518, November.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Aditya Krishna Menon and Charles Elkan. 2011. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer.

Mark A Musen, Yuval Shahar, and Edward H Shortliffe. 2006. Clinical decision-support systems.

Michael Pinedo, Sridhar Seshadri, and J George Shanthikumar. 2000. Call centers in financial services: strategies, technologies, and operations. In *Creating Value in Financial Services*, pages 357–388. Springer.

Hamed Pirsiavash, Deva Ramanan, and Charless Fowlkes. 2009. Bilinear classifiers for visual recognition. In *NIPS*, pages 1482–1490.

Pradeep Ravikumar, Martin J Wainwright, and John D Lafferty. 2010. High-dimensional ising model selection using 1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471, Prague, Czech Republic, June. Association for Computational Linguistics.

Gerard Salton and Michael J McGill. 1983. Introduction to modern information retrieval.

T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. 2004. Indri: A language model-based search engine for complex queries. *Proceedings of the International Conference on Intelligence Analysis*.

Xu Tan, Yin Zhang, Siliang Tang, Jian Shao, Fei Wu, and Yueting Zhuang. 2013. Logistic tensor regression for classification. In *Intelligent Science and Intelligent Data Engineering*, pages 573–581. Springer.

Jean-Philippe Vert and Laurent Jacob. 2008. Machine learning for in silico virtual screening and chemical genomics: new strategies. *Combinatorial chemistry & high throughput screening*, 11(8):677.

Wei Wu, Zhengdong Lu, and Hang Li. 2013. Learning bilinear model for matching queries and documents. *The Journal of Machine Learning Research*, 14(1):2519–2548.