Measure Search Relevance for Home Depot Products

Peng Xu
Department of Computing Science
pxu4@ualberta.ca

ABSTRACT

In this project, the task is to measure the relevance between products and search terms. This task can be split into two stages. The first stage is to extract effective features from the available data, namely feature extraction. In this stage, I explore different techniques in both information retrieval and natural language processing to measure relevance from different aspects. The second stage is to learn from the features and predict the relevance between search queries and products. In this stage, the task can be formulated as a regression problem. I try different ensemble methods using the features extracted in the first stage and use cross-validation to choose best parameters for the models. I tried various experiments to improve the performance on the test data, and finally I got the root mean squared error (RMSE) of 0.46248 and ranked in top 10% out of more than two thousand teams.

Keywords

Information Retrieval, Natural Language Processing, Machine Learning

1. INTRODUCTION

1.1 Motivation

Measure relevance between search queries and documents is an important issue in information retrieval. Inspired by the Kaggle comeptition "Home Depot Product Search Revelane"[1], we will focus on measuring relevance between products and real customer search terms from Home Depot's website in this project.

Shoppers rely on Home Depot's product authority to find and buy the latest products and to get timely solutions to their home improvement needs. Speed, accuracy and delivering a frictionless customer experience are essential. In this competition, the goal is to help improve the customers' shopping experience by developing a model that can accurately predict the relevance of search results.

Search relevancy is an implicit measure Home Depot uses to gauge how quickly they can get customers to the right products. Currently, human raters evaluate the impact of potential changes to their search algorithms, which is a slow and subjective process. By removing or minimizing human input in search relevance evaluation, Home Depot hopes to increase the number of iterations their team can perform on the current search algorithms.

1.2 Task Definition

In this competition, the data set contains a number of products and real customer search terms from Home Depot's website. The challenge is to predict a relevance score for the provided combinations of search terms and products. To create the ground truth labels, Home Depot has crowdsourced the search/product pairs to multiple human raters. The task is to predict the relevance for each pair listed in the test set.

The results are evaluated on the root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where y_i denotes the labeled relevance and $\hat{y}_i$ denotes the predicted relevance.

The task can be divided into two stages. The first stage is to extract effective features from the available data, namely feature extraction. In this stage, I explore different techniques in both information retrieval and natural language processing to measure relevance from different aspects. The second stage is to learn from the features and predict the relevance between search queries and products. In this stage, the task can be treated simply as a traditional regression problem. In this case, I try different ensemble methods using the features extracted in the first stage and use cross-validation to choose best parameters for the models.

The outline of this paper is as follows. I introduce some related work in section 2. In section 3, I describe how I preprocess the data. In section 4, the methods of feature extraction are illurstrated. In section 5, the learning models used in this project are introduced. In section 6, it shows the details about experiments and the final results along with the analysis on them. Finally I summarize our work and potential further work in section 7.

[1]https://www.kaggle.com/c/home-depot-product-search-relevance

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

2. RELATED WORK

Latent semantic indexing (LSI), query expansion and statistical language models are all studied a lot in the area of information retrevial. They're used to address different problems or different aspects of an problem. In my project, I combine all these techniques together to achieve better performance to predict the relevance between products and search queries.

2.1 Latent Semantic Indexing

Dumais (1993) [7] and Dumais (1995) [4] describe experiments on TREC benchmarks giving evidence that at least on some benchmarks, LSI can produce better precision and recall than standard vector-space retrieval. Schütze and Silverstein (1997) [32] evaluate LSI and truncated representations of centroids for efficient K-means clustering. Bast and Majumdar (2005) [2] detail the role of the reduced dimension k in LSI and how different pairs of terms get coalesced together at differing values of k. Applications of LSI to cross-language information retrieval (where documents in two or more different languages are indexed, and a query posed in one language is expected to retrieve documents in other languages) are developed in Berry and Young (1995) [5] and Littman et al. (1998) [21]. LSI (referred to as LSA in more gen- eral settings) has been applied to host of other problems in computer science ranging from memory modeling to computer vision.

Hofmann (1999a;b) [13, 12] provides an initial probabilistic extension of the basic latent semantic indexing technique. A more satisfactory formal basis for a probabilistic latent variable model for dimensionality reduction is the Latent Dirichlet Allocation (LDA) model (Blei et al. 2003) [6], which is generative and assigns probabilities to documents outside of the training set.

2.2 Query Expansion

Different approaches have been proposed for selecting expansion terms. Pseudo-relevance feedback (PRF) assumes that the top-ranked documents returned for the initial query are relevant, and uses a sub set of the terms extracted from those documents for expansion. PRF has been proven to be effective in improving retrieval performance [20].

Corpus-specific approaches analyze the content of the whole document collection, and then generate correlation between each pair of terms by co-occurrence [24], mutual information [14], etc. Mutual information (MI) is a good measure to assess how much two terms are related, by analyzing the entire col- lection in order to extract the association between terms. For each query term, every term that has a high mutual information score with it is used to expand the user query.

Many approaches exploit knowledge bases or thesauruses for query expansion, among them: WordNet [44], UMLS Meta thesaurus [45], Wikipedia [40], etc. The nature of these resources varies: linguistic like WordNet, domain specific like UMLS in the medical domain, or knowledge about named entities like Wikipedia.

Other approaches like semantic vectors and neural probabilistic language models, propose a rich term representation in order to capture the similarity between terms. In these approaches, a term is represented by a mathematical object in a high dimensional semantic space which is equipped with a metric. The metric can naturally encode similarities between the corresponding terms. A typical instantiation of these approaches is to represent each term by a vector and use a cosine or distance between term vectors in order to measure term similarity [3, 33, 38].

2.3 Statistical Language Models

Statistical language models have recently been successfully applied to many information retrieval problems [41]. A great deal of recent work has shown that statistical language models not only lead to superior empirical performance, but also facilitate parameter tuning and open up possibilities for modeling nontraditional retrieval problems. In general, statistical language models provide a principled way of modeling various kinds of retrieval problems.

The field has progressed in two different ways. On the one hand, theoretical models have been proposed often to model relevance through inferences; representative models include the logic models [11, 37, 39] and the inference network model [35]. However, these models, while theoretically interesting, have not been able to directly lead to empirically effective models, even though heuristic instantiations of them can be effective. On the other hand, there have been many empirical studies of models, including many variants of the vector space model [29, 30, 31, 34] and probabilistic models [10, 18, 25, 26, 36, 35]. The vector-space model with heuristic TF-IDF weighting and document length normalization has traditionally been one of the most effective retrieval models, and it remains quite competitive as a state of the art retrieval model. The popular BM25 (Okapi) retrieval function is very similar to a TF-IDF vector space retrieval function, but it is motivated and derived from the 2-Poisson probabilistic retrieval model [27, 28] with heuristic approximations. BM25 is one of the most robust and effective retrieval functions. Another effective retrieval model is divergence from randomness which is based on probabilistic justifications for several term weighting components [1].

While both vector space models and BM25 rely on heuristic design of retrieval functions, an interesting class of probabilistic models called language modeling approaches to retrieval have led to effective retrieval functions without much heuristic design. In particular, the query likelihood retrieval function [25] with Dirichlet prior smoothing has comparable performance to the most effective TF-IDF weighting retrieval functions including BM25 [8]. Due to their good empirical performance and great potential of leveraging statistical estimation methods, the language modeling approaches have been attracting much attention since Ponte and Croft?s pioneering paper published in ACM SIGIR 1998 [25]. Many variations of the basic language mod- eling approach have since been proposed and studied, and language models have now been applied to multiple retrieval tasks such as cross-lingual retrieval [22], distributed IR [17, 23], expert finding [9], passage retrieval [22], web search [17, 23], genomics retrieval [42], topic tracking [15, 19, 16], and subtopic retrieval [43].

3. PREPROCESSING

The given information about each product is somewhat dirty and poor-structured. As a result, it's necessary to clean and convert the data into the desired format. In the dataset, each product has the following information:

- **product uid**: Unique identifier of each product.

- **product title**: Title of each product.
- **product description**: A text description of each product.
- **attributes (optional)**: Provide extended information about a subset of the products. Not every product will have attributes.

In order to clean the text information, I conduct a series of basic operations on the data, including parsing, stemming, removing html content and punctuations, correcting common spelling mistakes and so on. Furthermore, I find that most attributes are extracted from the product description except one attribute which denotes brand of the product, after exploring the attributes information. So I extract this attribute and denote it as **product brand**.

Apart from the basic operations described above, I use three different schemes to handle the text information in the data :

- D_1: Just apply basic operations on the text information.
- D_2: Apply basic operations and remove the common stopwords.
- D_3: Apply basic operations and remove the stopwords and numbers.

These three corpora may reflect different aspects of the data, and I will apply feature extraction and learning models to these three corpora simultaneously.

4. FEATURE EXTRACTION

Feature extraction may be the most important part in a machine learning contest. In this section, I make use of many techniques in both information retrieval and natural language processing to extract the effective features from the given corpus.

4.1 Basic Features

First, I extract some basic features for each (product, search terms) pair, which descibes the size of the text information and the brand feature:

- **len_of_query**: length of the search term
- **len_of_title**: length of the title of each product
- **len_of_description**: length of the description of each product
- **len_of_brand**: length of the brand of each product
- **brand_feature**: unique identifier for each brand

Then, I calculate the number of common words between search terms and the text information about the product:

- **words_in_title**: common words between search terms and title of the product
- **words_in_description**: common words between search terms and title of the description
- **words_in_brand**: common words between search terms and title of the brand

One thing to be noted is that the last search term usually has much more importance than other terms. So here I add two more features about the last search term:

- **last_word_in_title**: whether last word in search terms is in title of the product
- **last_word_in_description**: whether last word in search terms is in description of the product

Finally, I also add the ratio of common words in search terms:

- **ratio_of_title**: **words_in_title** divided by **len_of_query**
- **ratio_of_description**: **words_in_description** divided by **len_of_query**
- **ratio_of_brand**: **words_in_brand** divided by **len_of_query**

4.2 Correct Search Terms

After looking into the data, I find that there are quite a lot search terms unseen in the vocabulary of the whole corpus. The appearances of those words are mainly due to the spelling mistakes.

In order to reduce the effect of spelling mistakes, I use the following strategy to correct the unseen search terms. For a certain unseen search term, I first calculate its edit distance with the vocubulary of the whole corpus. Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Then, I replace the unseen term with the word with minimal edit distance. If there are multiple words with the same minimal edit distance, I choose the one with the highest frequency in the whole corpus.

I also explore some other metrics, like Jaro-Winkler Distance and Soundex, to correct search terms. But the performance of these metrics are not good.

4.3 Latent Semantic Indexing

Latent semantic indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts.

LSI is actually a transformation which applies truncated SVD to term-document matrices and it transforms such matrices to a "semantic" space of low dimensionality. In particular, LSA is known to combat the effects of synonymy and polysemy (both of which roughly mean there are multiple meanings per word), which cause term-document matrices to be overly sparse and exhibit poor similarity under measures such as cosine similarity.

Mathematically, truncated SVD applied to training samples X produces a low-rank approximation X_k:

$$X \approx X_k = U_k \Sigma_k V_k^\top$$

After this operation, $U_k \Sigma_k^\top$ is the transformed training set with k features.

To also transform a test set X, we multiply it with V_k:

$$X' = XV_k.$$

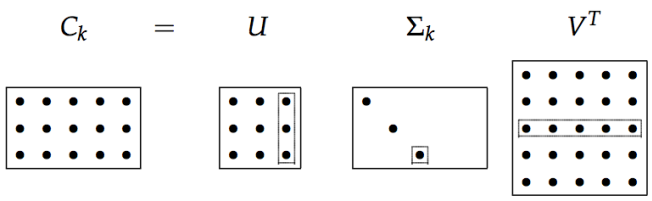

Figure 1: Illustration of low-rank approximation using the singular value decomposition. The dashed boxes indicate the matrix entries affected by "zeroing out" the smallest singular values.

In this project, I extract two kinds of features from the corpus using the LSI. First, I apply the LSI on the term-document matrices of search terms, title, description and brand seperately and treat the low-rank approximations as features directly. Second, I combine all the text information including title and description together for each product as documents and apply LSI on the corresponding term-document matrices. Then, I transfroms the query to the semantic space by multiplying V_k as illurstrated above. Finally, I treat the cosine distance between document vectors and query vectors as features.

Also, I try latent dirichlet allocation (LDA) on the corpus and use topic distributions as features. However, the results show that the features from LDA don't work on this task, compared to LSI.

4.4 Query Expansion by Word2Vec

In this project, the queries are just some keywords and usually too short to describe the information need accurately. Important terms can be missing from the query, leading to a poor coverage of the relevant documents. To solve this problem, automatic query expansion techniques leveraging on serveral data sources and employ different methods for find expansion terms.

Recently, several efficient NLP methods, based on Deep Learning, are proposed to learn high quality vector representations of terms from a large amount of unstructured text data with billions of words. This high quality vector representation captures a large number of term relationships. In my project, I investigate these term vector representations in query expansion.

Learning takes places from the corpus of all the text information including title and description. The resulting vectors carry relationships between terms, such as a city and the country it belongs to, e.g. France is to Pairs what Germany is to Berline. Therefore, each term t is represented by a vector of a predefined dimension v_t. The similarity between two terms t_1 and t_2 is measured with the normalized cosine between their two vectors: v_{t_1} and v_{t_2}.

$$SIM(t_1, t_2) = \widetilde{cos}(v_{t_1}, v_{t_2})$$

where $\widetilde{cos}(v_{t_1}, v_{t_2}) \in [0, 1]$ is the normalized cosine between the two term vectors v_{t_1} and v_{t_2}. Based on this normalized cosine similarity between terms, we now define the function that returns the k-most similar terms to a term t, $top_k(t)$:

$$top_k : V \rightarrow 2^V$$

where V is the set of all terms t.

Let q be a user query represented by a bag of terms, $q = [t_1, t_2, \ldots, t_{|q|}]$. Each term in the query has a frequency $\#(t, q)$. In order to expand a query q, I follow these steps:

- For each $t \in q$, collect the k-most similar terms to t using the function $top_k(t)$. The expanded query q' is defined as follows: $q' = q \bigcup_{t \in q} top_k(t)$.
- The frequency of each $t \in q$ still the same in the expanded query q'.
- The frequency of each expansion term $t' \in top_k(t)$ in the expanded query q' is given as follows:

$$\#(t', q') = \#(t, q') \times \widetilde{cos}(v_t, v_{t'}).$$

After query expansion, we can use the expanded queries to recalculate all the basic features and LSI features, which are very effective as the experiments show.

4.5 Language Models and Ranking

Statistical language models have recently been successfully applied to many information retrieval problems. In my project, it can be used to calculate a score for relevance between queries and documents. In addition, based on the scores, I can also obtain the ranking of a certain document among all documents. And the relevance score and the ranking can be treated as features in the regression model.

In my project, I use the Indri search engine to calculate the scores and obtain the corresponding rankings. The Indri search engine, developed as part of the Lemur Project, is designed to be both efficient and effective over a wide range of collections, especially large, semi-structured text collections, such as the web. Indri makes use of INQUERY's underlying inference network retrieval framework, which allows complex structured queries to be constructed and evaluated. However, Indri makes use of language modeling probabilities instead of INQUERY's tf.idf based probabilities, which provides increased robustness, as reflected in the Indri query language.

5. MODELS

In this project, the task can be formulated as a regression problem essentially. After feature extraction, we need to fit the training data on a learning model and give predictions on the test data. Here I explore three different models:

5.1 Support Vector Machine

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that

same space and predicted to belong to a category based on which side of the gap they fall on.

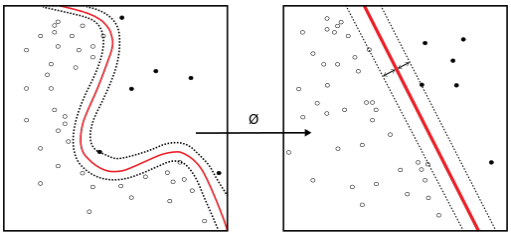

Figure 2: Example of SVM

A version of SVM for regression is called support vector regression (SVR). The model produced by support vector classification as described above depends only on a subset of the training data, because the cost function for building the model does not care about training points that lie beyond the margin. Analogously, the model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

5.2 Random Forests

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

5.2.1 *Tree Bagging*

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, \ldots, x_n$ with responses $Y = y_1, \ldots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

- Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
- Train a decision or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x')$$

or by taking the majority vote in the case of decision trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

5.2.2 *From Bagging to Random Forests*

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by Ho.

5.3 Gradient Boosting Trees

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners, namely gradient boosting trees. In some sense, it's a generalization of the tree ensembles and can prevent overfitting effectively. In practice, it's extremely powerful and usually can outperform many other models if the parameters are selected properly.

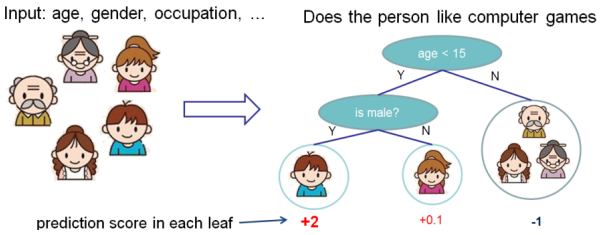

Figure 3: Example of One CART Tree

In my project, I use the *XGBoost* package in python to train the gradient boosting trees. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting(also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

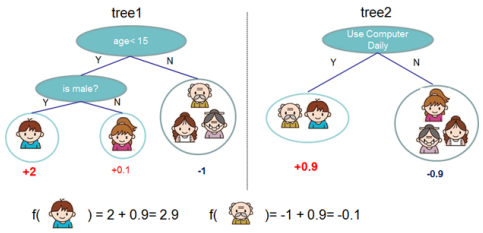

Figure 4: Example of Tree Ensemble

6. EVALUATION

6.1 Resources

For the implementation, I use python as the programming language. And the two packages *scikit-learn* and *XGBoost* are heavily used in my project. Support vector regression, random forest, gradient boosting trees, latent semantic indexing, grid search and cross-validation are all convenient to implement with the help of these two packages.

6.2 CV Results on Different Feature Sets

In the experiments, I first use cross-validation to test each features effectiveness on the preprocessed training data D_1. Then, I combine all effective features together and use three different models to get the final results on test data. Of course, I also use grid search to select the optimal parameters for each model.

Table 1 shows the CV results measured by RMSE on different feature sets learned by gradient boosting trees.

Table 1: CV results on different feature sets

Feature Set	Baseline	F1	F2	F3	F4
RMSE	0.4844	0.4830	0.4799	0.4797	0.4702

How these feature sets are constructed is explained as follows:

- **Baseline**: Basic features only.
- **F1**: Basic features + Basic features based on corrected search terms.
- **F2**: Basic features + Basic features based on expanded search terms.
- **F3**: Basic features + Language model based relevance scores & rankings.
- **F4**: Basic features + Latent semantic indexing features.

As the results show, all these extracted features can have modest improvements on the CV results, but not that significant. Among them, the LSI features bring the model the largest improvement. In order to improve the performance of the model further, I combine all the feature extraction methods together and I can get more features. For example, I can apply LSI on the corrected search terms and the expanded search terms. Also, I can also apply language models to the corrected search terms and the expanded search terms.

Table 2 shows the CV results of combined features measured by RMSE on training data preprocessed by different strategies by gradient boosting trees.

Table 2: CV results of combined features with different preprocessing strategies

Dataset	D1	D2	D3
RMSE	0.4563	0.4589	0.4659

Compared to the results in Table 1, the improvement is significant over the baseline when we combine all the features together. In addition, D_1 performs better than D_2 and D_3. It shows that the stopwords and numbers can also be meaningful from some aspects and just omit them is not a wise approach.

Table 3 shows the CV results of combined features measured by RMSE on training data D_1 using different models.

Table 3: CV results of combined features with different preprocessing strategies

Models	GBT	RF	SVR
RMSE	0.4563	0.4640	0.4653

Generally, gradient boosting trees achieve the best performance in the three models. Actually, gradient boosting trees is the main regression model in my project and I spend quite a lot time to tune the hyper-parameters of the gradient boosting trees.

6.3 Final Results on Test Data

I've submitted 71 entries on the Kaggle website in total. The best result is 0.46248 by gradient boosting trees model with combined features on D_1. And I'm ranked in top 10% out of more than two thousand teams in this competition. However, there is still a gap between my best results and the leaders in this competition. The team ranked first has the results of 0.43083. It's obvious that more work can be done to improve my final results.

My origin plan is to ensemble all the models' results on data preprocessed by different strategies as the final step of the project. However, after some attempts, the ensemble results are not that good. I think the reason may be that the features I used in the models are too similar and the emsemble just doesn't work. I may explore it further after the project.

7. CONCLUSIONS AND FURTHER WORK

In this project, I explore different techniques in both information retrieval and natural language processing to extract effective features from the corpus, including lantent semantic indexing, word2vec, statistical language models. Also, I try three different learning models: support vector machine,

random forest and gradient boosting trees and use cross-validation to select the optimal parameters for each model. As a result, I get best root mean squared error (RMSE) of 0.46248 and ranked in top 10% out of more than two thousand teams in this competition.

In the future, more teniques can be used to extract more features. For example, I can try other approaches to expand the query. Also, I may explore further how to ensemble different results together to get more improvements on the final results.

8. ACKNOWLEDGEMENTS

I would like to express our appreciation to Dr. Denilson Barbosa. Thanks for his time and efforts on guiding the whole process of my project.

9. REFERENCES

[1] G. Amati and C. J. V. Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans- actions on Information System*, 20(357–389), 2002.

[2] H. Bast and D. Majumdar. Why spectral retrieval works. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, 2005.

[3] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. *Innovations in Machine Learning*, pages 137–186, 2006.

[4] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595, 1995.

[5] M. W. Berry and P. G. Young. Using latent semantic indexing for multilanguage information retrieval. *Computers and the Humanities*, 29(6):413–429, 1995.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[7] S. Dumais. Lsi meets trec: A status report. *Proceedings of the first Text REtrieval Conference, TREC1*, pages 137–152, 1993.

[8] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56, 2004.

[9] H. Fang and C. Zhai. Probabilistic models for expert finding. *Proceedings of ECIR*, 2007.

[10] N. Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[11] N. Fuhr. Language models and uncertain inference in information retrieval. *Proceedings of the Language Modeling and IR workshop*, pages 6–11, 2001.

[12] T. Hofmann. Probabilistic latent semantic analysis. *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296, 1999.

[13] T. Hofmann. Probabilistic latent semantic indexing. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[14] J. Hu, W. Deng, and J. Guo. Improving retrieval performance by global analysis. *Pattern Recognition, 2006. ICPR 2006.*, 2:703–706, 2006.

[15] H. Jin, R. Schwartz, S. Sista, and F. Walls. Topic tracking for radio, tv broadcast and newswire. *Proceedings of DARPA Broadcast News Workshop*, pages 199–204, 1999.

[16] W. Kraaij and M. Spitters. Language models for topic tracking. *Language Modeling for Information Retrieval*, pages 95–123, 2003.

[17] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 27–34, 2002.

[18] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. *Language modeling for information retrieval*, pages 1–10, 2003.

[19] V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard, and S. Thomas. Relevance models for topic detection and tracking. *Proceedings of the second international conference on Human Language Technology Research*, pages 115–121, 2002.

[20] V. Lavrenko and W. B. Croft. Relevance based language models. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127, 2001.

[21] M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. *Cross-language information retrieval*, pages 51–62, 1998.

[22] X. Liu and W. B. Croft. Passage retrieval based on language models. *Proceedings of the eleventh international conference on Information and knowledge management*, pages 375–382, 2002.

[23] P. Ogilvie and J. P. Callan. Experiments using the lemur toolkit. *TREC*, 10:103–108, 2001.

[24] H. J. Peat and P. Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the american society for information science*, 42(5):378, 1991.

[25] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, 1998.

[26] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[27] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, 1994.

[28] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, 109:109, 1995.

[29] G. Salton. Automatic text processing: The transformation, analysis, and retrieval of information by computer. *Reading: Addison-Wesley*, 1989.

[30] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, Inc., 1986.

[31] G. Salton, C.-S. Yang, and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of the American society for Information Science*, 26(1):33–44, 1975.

[32] H. Schütze and C. Silverstein. Projections for efficient document clustering. *ACM SIGIR Forum*, 31(SI):74–81, 1997.

[33] M. Serizawa and I. Kobayashi. A study on query expansion based on topic distributions of retrieved documents. *Computational Linguistics and Intelligent Text Processing*, pages 369–379, 2013.

[34] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, 1996.

[35] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems (TOIS)*, 9(3):187–222, 1991.

[36] C. J. van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of documentation*, 33(2):106–119, 1977.

[37] C. J. Van Rijsbergen. A non-classical logic for information retrieval. *The computer journal*, 29(6):481–485, 1986.

[38] D. Widdows and T. Cohen. The semantic vectors package: New algorithms and public tools for distributional semantics. *Semantic Computing (ICSC)*, (9–15), 2010.

[39] S. K. M. Wong and Y. Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems (TOIS)*, 13(1):38–68, 1995.

[40] Y. Xu, G. J. Jones, and B. Wang. Query dependent pseudo-relevance feedback based on wikipedia. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 59–66, 2009.

[41] C. Zhai. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141, 2008.

[42] C. Zhai, T. Tao, H. Fang, and Z. Shang. Improving the robustness of language models-uiuc trec 2003 robust and genomics experiments. *TREC*, pages 667–672, 2003.

[43] C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10–17, 2003.

[44] J. Zhang, B. Deng, and X. Li. Concept based query expansion using wordnet. *Proceedings of the 2009 international e-conference on advanced science and technology*, pages 52–55, 2009.

[45] W. Zhu, X. Xu, X. Hu, I.-Y. Song, and R. B. Allen. Using umls-based re-weighting terms as a query expansion strategy. *GrC*, pages 217–222, 2006.