Journal Of Al Azhar University Engineering Sector

Vol. 13, No. 46, January, 2018, 122-132

PCA REDUCED FOREST FOR LEARNING TO RANK SPOKEN TRANSCRIPTIONS

Farida Sabry, Mayada Hadhoud and Nevin Darwish
Computer Engineering Department, Faculty of Engineering, Cairo University

ABSTRACT

This paper discusses the problem of learning to rank specially for spoken transcriptions. The state-of-art approach for text/web documents is to apply machine learning techniques to learn a ranking model from labeled query-documents pairs with their features. One of the best state-of-art learning algorithms is the Random Forest, however it does not perform very well when features are dependent or are monotonic transformation of other features as this makes the trees of the forest less independent. We propose to use Principal Component Analysis (PCA) to bags of features, in order to reduce them to simplify the model and have a surrogate score for each field's features producing more independent set of features for the Random Forest. Using this technique for a transcriptions dataset, 4.32% improvement in terms of Expected Reciprocal Rank (ERR@10) and 0.4% improvement in terms of Normalized Discounted Cumulative Gain (NDCG@10) for training data are achieved with very comparable results for the testing data. We emphasized the effectiveness of the technique by applying it to the larger and benchmarked web documents dataset; Microsoft LETOR. An improvement of 7.99% and 1.29% for test data are achieved for the two used metrics respectively.

1. INTRODUCTION

Multimedia on the web is increasing in a dramatic way with hundreds of hours of spoken content being shared every minute on websites like YouTube, vimeo, NetFlix, SoundCloud and YouListen. This spoken content contains valuable information, but this information becomes useless unless this huge volume of multimedia can be effectively browsed and searched. Spoken Content Retrieval (SCR) can be defined as the task of returning speech media results that are relevant to an information need expressed in a user query [1]. A general architecture that represents an abstraction for SCR system for the block diagram presented in [1] can be viewed as shown in Figure 1. Spoken content is converted into a lattice representation or best transcription together with timing information using the Automatic Speech Recognition (ASR) component [1]. The indexing module is then used to index the lattice representation or best transcription together with the metadata extracted for the spoken documents to produce a timed index. When a user enters a query through the user interface of the search engine, the query terms are analyzed using the same analyzer used by the indexer, this is represented as a dotted line in Figure 1 between the indexing and retrieval building blocks. The search process is then done on the timed index. Audio segments that match the user's query are retrieved and ranked.

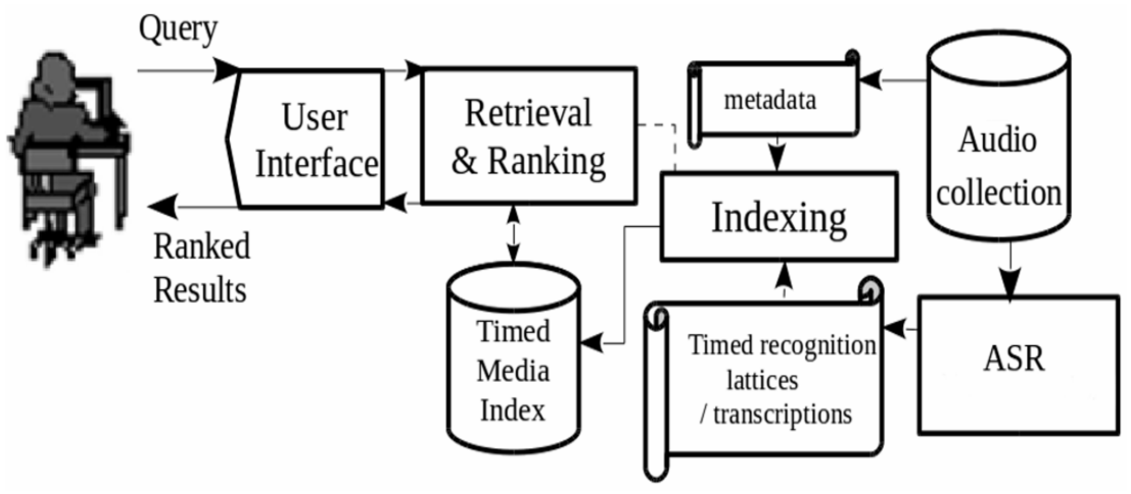


Figure 1: Spoken Document Retrieval Architecture

The retrieved results are ranked mainly based upon the titles of the videos and their meta-data {description, tags, views, ratings, playlist, shares, comments, age of videos, channel views and subscribers inbound links (e.g. links from outside of YouTube pointing to your videos)}. Dishonest Search Engine Optimization (SEO) practices can affect the retrieved results accuracy because the title of the video and its metadata can be faked. Using keywords known for their high hit rate while the actual content of the video may be irrelevant can cause irrelevant documents to be ranked among legitimate ones. In order to overcome this challenge while ranking the retrieved documents, we proposed to extract features, not only based on the aforementioned metadata features, but also on the timed transcriptions of the spoken content. To do ranking of the retrieved documents based on the extracted features, machine learning techniques are applied. A ranking model is learned from labeled query-documents pairs and their corresponding extracted features. Using this learned model, the ranking label of a retrieved document for unseen user query can be predicted. Learning to rank techniques have been successfully adopted in other tasks, like ranking tweets [2], entities search [3] and searching images [4]. Random Forest algorithm [5] is one of the best state-of-art Learning To Rank (L2R) algorithms. Other algorithms include LambdaMART [6], RankNet [7], ListNet [8] and RankSVM [9]. These learning to rank algorithms, in addition to others, are best reviewed in [10]. Random Forest [5], however does not perform well when features are dependent or are monotonic transformation of other features as this makes the trees of the forest less independent from each other. Another drawback of Random Forest is the model size, as having a lot of features will result in a forest that takes a lot of memory and is slow to evaluate. In this work, we propose to use a reduced set of features by using the well known Principal Component Analysis (PCA) algorithm that do dimensionality reduction, dimensionality reduction techniques are reviewed in [11]. We use PCA in the learning of forest by projecting bags of features to a new space to simplify the random forest learned model and reduce the dependency among features. In section 2, ranking spoken content transcriptions is formulated as a learning to rank problem listing the set of extracted features. Section 3 discusses the proposed algorithm for PCA Reduced Forest. The used datasets and results are presented in section 4. Conclusion and future work are explained in section 5.

2. L2R for spoken content transcriptions

In this section, we formulate the problem of SCR as a learning to rank problem. We propose to represent our L2R framework for SCR as in Figure 2. The query set **Q**; $Q = \{q_1, q_2, \cdots, q_m\}$ is the set of queries used for training and testing. The set of all audio transcriptions is **D**; $D = \{D_1, D_2, \dots D_i, \dots, D_m\}$ where $D_i = \left\{d_{i,1}, d_{i,2}, \cdots, d_{i,n_i}\right\}$ is the set of transcription documents of size n_i retrieved for query q_i and $R_i = \{r_{i,1}, r_{i,2}, \cdots, r_{i,n_i}\}$ is the set of labels associated with the query-transcription pairs $(q_i, d_{i,j}), j \in \{1,2, \dots n_i\}$, where labels represent relevance grades. A feature vector $F_{i,j} = \{f_{i,j}^1, f_{i,j}^2, \dots\dots, f_{i,j}^x\}$ of x features is extracted for each query-transcription document pair $(q_i, d_{i,j})$ with some features being query-dependent that they depend on the similarity and matching of a document to the terms of the query. Other features are query-independent and can be extracted for each document ahead of query-time. A training set $QD_t = \{(q_1, D_1, F_{1,j}, R_1), (q_2, D_2, F_{2,j}, R_2), \dots, (q_t, D_t, F_{t,j}, R_t)\}$ is formed from query-documents pairs $(q_i, D_i, F_{i,j}, R_i)$ where $q_i \in Q_t$ and $Q_t = \{q_1, q_2, \cdots, q_t\}$ which is a subset of **Q**; and its corresponding documents D_i with features vectors $F_{i,j}$ and relevance labels R_i. And a testing set QD_T is formed from Q_T; a subset of Q and their corresponding documents features vectors and relevance labels where $Q_T = \{q_1, q_2, \cdots, q_T\}$ and $QD_T = \{(q_1, D_1, F_{1,j}, R_1), (q_2, D_2, F_{2,j}, R_2), \dots, (q_T, D_T, F_{T,j}, R_T)\}$.

Spoken documents transcriptions D, can be structured into different textual fields for which user queries are matched like title, channel name, tags, description and here we added features related to the timed transcription segments. Some features extracted from these textual fields are query-dependent features used for measuring the matching of the query terms with text in these fields. We added a feature that represents the listening time for the first occurrence of a full covered query match. This gives preference for for documents with shorter listening time to first match. This is preferable for low-bandwidth connections. A complete list of features we extracted from transcriptions are listed in Table 1.

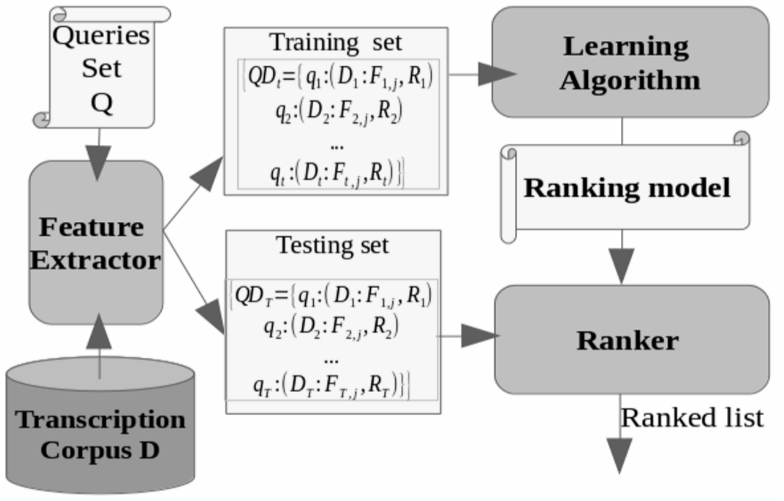

Queries Set Q
Training set
$QD_t = \{q_1 : (D_1 : F_{1,j}, R_1)$
$q_2 : (D_2 : F_{2,j}, R_2)$
...
$q_t : (D_t : F_{t,j}, R_t)\}$
Learning Algorithm
Ranking model
Feature Extractor
Testing set
$QD_T = \{q_1 : (D_1 : F_{1,j}, R_1)$
$q_2 : (D_2 : F_{2,j}, R_2)$
...
$q_T : (D_T : F_{T,j}, R_T)\}$
Ranker
Ranked list
Transcription Corpus D

Figure 2: Learning to rank framework for spoken transcriptions retrieval

The query-dependent features can be further classified into: query-terms-matching features, probabilistic-model features like BM25 and language modeling (LM) scores. Language modeling approaches in information retrieval try to estimate a language model for each document and then rank documents according to the likelihood that the query at hand has been generated from the estimated language model. We also used language modeling smoothing methods which are reviewed in [12]. The groups of sixes (6:11, 12:17 ...and 66:71) in Table 1 correspond to the features for each of the five textual fields in the transcription document mentioned before, in addition to the whole document. The query-independent features are based on the metadata and measures of the importance and freshness of the audio. They include uploaded time, views count, likes count, dislikes count, comments count, the stream length of the fields in the transcription file, the duration of the audio file, the number of segments for the transcription of speech. Query-independent features are formatted in italics in Table 1. Learning to rank techniques developed for ranking text documents can be used to rank spoken documents transcriptions based on the extracted features ($x = 75$). Graded relevance evaluation metrics like Expected Reciprocal Rank (ERR) [13] and Normalized Discounted Cumulative Gain (NDCG) [14] are used for evaluating the output of the L2R model.

Table 1: Sets of extracted features from the spoken transcriptions and their descriptions

ID	Feature	Description
1	*age of the document*	$\frac{currentTime - uploadedTime}{currentTime}$
2	*comments count*	Number of comments
3	*views count*	Number of views
4	*likes count*	Number of likes
5	*dislikes count*	Number of dislikes
6:11	covered query term number	Number of matching query terms in a field
12:17	covered query term ratio	$\frac{numberofmatchingqueryterms}{totalnumberofqueryterms}$
18:23	*fieldLength*	Length of each of the six field in terms of number of tokens
24:29	IDF (Inverse Document Frequency)	$\sum_{q_i \in q \cap d} \frac{\log N}{n_i}$
30:35	TF (Sum of Term Frequency)	$\sum_{q_i \in q \cap d} TF(q_i, d)$
36:41	TF-IDF	$\sum TF.IDF$
42:47	Boolean Model	whether query terms exists in the transcription field or not
48:53	BM25	$\sum_{i=1}^{n} IDF(q_i). \frac{f(q_i, D).(k_1 + 1)}{f(q_{iD}, D) + k_1.\left(1 - b + b.\frac{\lvert D \rvert}{avgdl}\right)}$
54:59	LM with Jelinek Mercer Smoothing	$P_\lambda(w \vee d) = (1 - \lambda)P_{ml}(w \vee d) + \lambda P(w \vee C)$
60: 65	LM with Absolute Discounting	$P_s(w \vee d) = \frac{max(c(w; d) - \delta, 0)}{\sum_{w' \in V} c(w'; d)} + \sigma P(w \vee C)$
66:71	LM with Dirichlet	$P_\mu(w \vee d) = \frac{c(w; d) + \mu P(w \vee C)}{\sum_{w' \in V} c(w'; d) + \mu}$
72	*nsegments*	Number of segments
73	match_start	Starting time for the first matching segment
74	*duration*	duration of the audio file
75	relevant_segment_duration	$\frac{\sum_{i=1}^{S_d} rel(s_i) * duration(s_i)}{\sum_{i=1}^{S_d} duration(s_i)}$

Tree-based L2R algorithms and specifically Random Forest algorithm [5] show the best accuracy

in terms of ERR and NDCG compared to other L2R algorithms [15][16]. However, Random Forest is greatly affected by the correlation between the features used. Breiman presented in [5] the properties of strength and correlation of trees that can control the accuracy of Random Forest algorithm. Correlation property is calculated based on a raw-margin function of the prediction accuracy of the Random Forest over all pairs of features sets selected in the “Random Feature Selection” step in Random Forest algorithm [5]. Breiman discussed that the less correlation between the trees, the less its error rate. Analysis and discussion of the two properties was also discussed in [17].
The main idea of this work is to decrease the correlation between the trees of the forest with the aim to enhance the accuracy of the learned Random Forest model. We thought of having a surrogate score for each field's features {title, channel name, tags, description, timed transcription segments and whole document} producing more independent set of features for the Random Forest and thus simplify the forest ranking model while at the same time decreasing the correlation between trees which will decrease the error rate. Principal Component Analysis (PCA) [11], one of the well-known dimensionality reduction techniques, is used in our algorithm to convert the set of possibly correlated features into a single feature score per field along the principal component. Steps of the proposed algorithm are discussed in the next section.

3. PCA Reduced Forest Algorithm

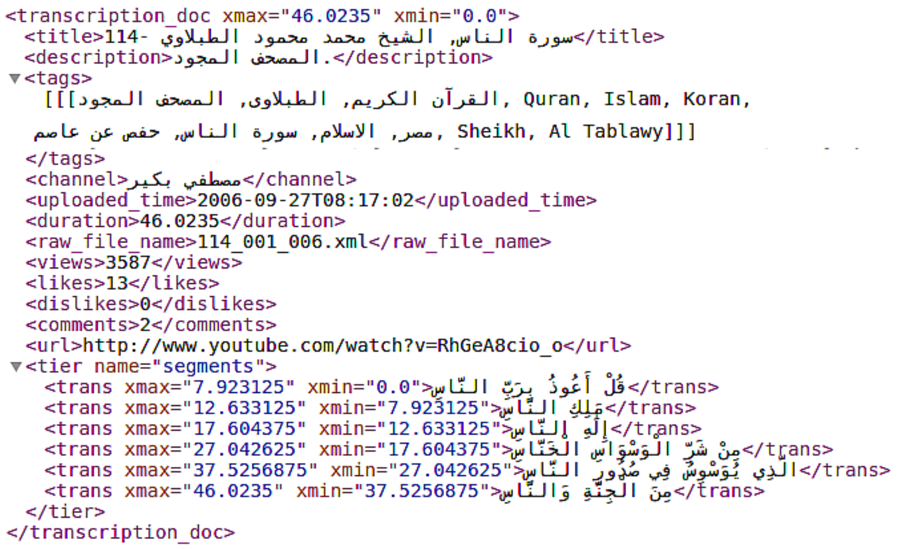
Learning to rank features extracted from the semi structured spoken transcriptions in this research or generally from web document as in Microsoft LETOR dataset [18] are grouped in terms of document fields which we name the bags of features X . For transcriptions documents for example, the bags of features X corresponds to groups of features related to the fields {F_1: title, description, F_2: channel, F_3: tags, F_4: transcription segments and F_5: the whole document}.
Pseudocode for the algorithm steps is presented in Figure 3. After features are distributed into bags based on the documents fields and other query-independent features, PCA reduction step is done for each bag separately. Each bag of features is thus reduced to a single score for each query-document pair along the principal component of the bag of features. This score represents a surrogate score for how the query matches a field based on the extracted features. A random forest is then built of B trees based on this new set of features $F_{reduced}$.
PCA reduces the amount of variables but still describes the same data. The PCA reduction step aims at enhancing the Random Forest performance by reducing the number of features and replacing related and collinear features with their single principal component.

```
**Input:** A training set $QD_t$, features $F$ , features' bags $X$ and number of forests $B$
**Output:** A forest ranking model $H$
**function** PCAReducedForest($QD_t$ , $F$ ):
        $H \leftarrow \varphi$
        $QD_{F_{reduced}}$ = PCA Reduction ($QD_t$, $F$ , $X$ )
        **for** $i \in 1,\ldots,B$ do
            $S^{(i)}$ ← A bootstrap sample from $QD_{F_{reduced}}$
            $\boldsymbol{h}_i$ ← ReducedTreeLearn($S^{(i)}$, $F_{reduced}$ )
            $H$ ← $H \cup \boldsymbol{h}_i$
        **end**
        **return** $H$
**function** PCA Reduction ($QD_t$, $F$ , $X$ ) :
        $F_{reduced} \leftarrow$ query-dependent features
        for each feature bag   x in X, where $F_x \in F$ :
                $F'_x$ ← adjust the features features $F_x$ from the training  dataset $QD_t$ to be of zero
                mean
                $C$ ← compute covariance matrix for $F'_x$
              compute eigen values and eigen vectors for $C$
            $f_x$ ← project $F'_x$ along the principal component
             $F_{reduced}$ ←$F_{reduced} \cup f_x$
     return $QD_{F_{reduced}}$
function ReducedTreeLearn($QD_s$ , $F_{reduced}$ ):
        At each node:
          begin
              Split on best feature in $F_{reduced}$
          end
        return the learned tree
```

Figure 3 : The proposed PCA Reduced Forest Algorithm

4. Dataset and Experimental Results

The dataset used in this study is based upon the verse by verse Quran dataset [19]. This dataset was chosen for two reasons; the first is the availability of timing information for verse segments in all chapters of the Quran. The second reason is the availability of text transcription for Quran verses. The corpus has been used to get the timings for each verse in each chapter in the Holy Quran. The corpus was then augmented to create more documents by forming documents of three consecutive verses, five consecutive verses and all verses in each chapter by combining verses' textgrids; the format used by Praat [20]. We then used YouTube API [21] with queries for each chapter to get realistic metadata for the documents. A timed transcription document in XML format is then generated as shown in Figure 4.



```
<transcription_doc xmax="46.0235" xmin="0.0">
  <title>114- سورة الناس, الشيخ محمد محمود الطبلاوي</title>
  <description>المصحف المجود.</description>
 ▼<tags>
    [[[المصحف المجود, الطبلاوى, القرآن الكريم, Quran, Islam, Koran,
   مصر, الاسلام, سورة الناس, حفص عن عاصم, Sheikh, Al Tablawy]]]
  </tags>
  <channel>مصطفي بكير</channel>
  <uploaded_time>2006-09-27T08:17:02</uploaded_time>
  <duration>46.0235</duration>
  <raw_file_name>114_001_006.xml</raw_file_name>
  <views>3587</views>
  <likes>13</likes>
  <dislikes>0</dislikes>
  <comments>2</comments>
  <url>http://www.youtube.com/watch?v=RhGeA8cio_o</url>
 ▼<tier name="segments">
    <trans xmax="7.923125" xmin="0.0">قُلْ أَعُوذُ بِرَبِّ النَّاسِ</trans>
    <trans xmax="12.633125" xmin="7.923125">مَلِكِ النَّاسِ</trans>
    <trans xmax="17.604375" xmin="12.633125">إِلَهِ النَّاسِ</trans>
    <trans xmax="27.042625" xmin="17.604375">مِنْ شَرِّ الْوَسْوَاسِ الْخَنَّاسِ</trans>
    <trans xmax="37.5256875" xmin="27.042625">الَّذِي يُوَسْوِسُ فِي صُدُورِ النَّاسِ</trans>
    <trans xmax="46.0235" xmin="37.5256875">مِنَ الْجِنَّةِ وَالنَّاسِ</trans>
  </tier>
</transcription_doc>
```

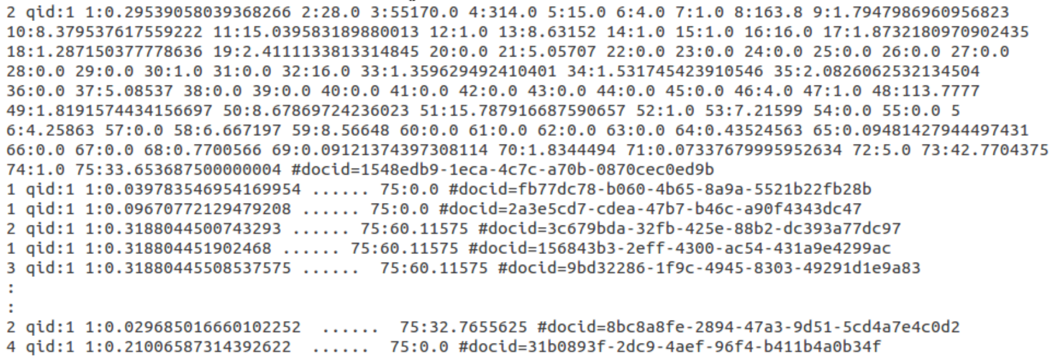
Figure 4 : Sample transcription XML file

A total of 30,544 transcription documents were generated of different lengths with consecutive verses and corresponding youtube fetched metadata for each set of documents, with about 4% spam documents having fake titles, descriptions and tags that are relevant to some queries while their actual content is irrelevant. The transcription documents were then indexed by the open source Apache Solr [22]. A set of 340 queries were built based on the knowledge of the domain under test “Quran files”. A pooling strategy as adopted in TREC (Text Retrieval Evaluation Conference) has been applied using Solr with applying different similarity measures. Queries vary in their number of terms and generality. We supplied the set of retrieved documents to our feature extractor to extract the 75 features listed in Table 1. The relevance of the retrieved documents to the query has been judged by giving one of 5 relevance labels from 0 to 4 as used in LETOR [18]. The LETOR formatted query-transcription pairs with their corresponding extracted features were then used to train learning to rank model using the proposed PCA Reduced Forest algorithm and compared to Random Forest algorithm for our dataset. To gain emphasis in the result and see how the algorithm will perform with larger datasets, we did a second experiment on the benchmarked Microsoft LETOR dataset [18] as well. Moreover, five-fold cross validation was used to overcome overfitting and to assess how the model will generalize to an independent unknown data set.

The query set in each dataset was divided into five folds. Each fold was further distributed; 70% of query-documents labeled pairs for training and validation and 30% for testing. For the case of transcriptions dataset, samples for the query-documents labeled pairs feature vectors of length 75 are shown in Figure 5. Five bags of features, each bag has 11 features, corresponding to textual fields features are input to the PCA Reduction step as discussed in section 3. The output of the PCA step is the projection of the features in the 5 bags in a new space to get the first principal component as a score for each field summarizing the 11 features. These new set of features are used to learn the forest model. Testing the model is done using 30% of each fold in the same format shown in Figure 5 as input, which is first fed to the PCA Reduction step as done with the training data. Then the new set of features is applied to the learned model to predict the ranks of each document and obtain a ranked list. The predicted ranks are used to calculate the evaluation metrics as in Equations (1) and (2) for the first ten results in the ranked list and is averaged over all the ranked lists for all queries in the test set (Equation 2 already includes the averaging over a set of queries Q).

$$ERR = \sum_{r=1}^{n} \frac{1}{r} \prod_{1}^{r-1} (1 - R_i) R_r \qquad (1)\ [13]$$

$$NDCG(Q,k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{k,j} \cdot \sum_{m=1}^{k} \frac{2^{R(j,m)} - 1}{\log_2(1+m)} \qquad (2)\ [14]$$



```
2 qid:1 1:0.29539058039368266 2:28.0 3:55170.0 4:314.0 5:15.0 6:4.0 7:1.0 8:163.8 9:1.7947986960956823
10:8.379537617559222 11:15.039583189880013 12:1.0 13:8.63152 14:1.0 15:1.0 16:16.0 17:1.8732180970902435
18:1.287150377778636 19:2.4111133813314845 20:0.0 21:5.05707 22:0.0 23:0.0 24:0.0 25:0.0 26:0.0 27:0.0
28:0.0 29:0.0 30:1.0 31:0.0 32:16.0 33:1.359629492410401 34:1.531745423910546 35:2.0826062532134504
36:0.0 37:5.08537 38:0.0 39:0.0 40:0.0 41:0.0 42:0.0 43:0.0 44:0.0 45:0.0 46:4.0 47:1.0 48:113.7777
49:1.8191574434156697 50:8.67869724236023 51:15.787916687590657 52:1.0 53:7.21599 54:0.0 55:0.0 5
6:4.25863 57:0.0 58:6.667197 59:8.56648 60:0.0 61:0.0 62:0.0 63:0.0 64:0.43524563 65:0.09481427944497431
66:0.0 67:0.0 68:0.7700566 69:0.09121374397308114 70:1.8344494 71:0.07337679995952634 72:5.0 73:42.7704375
74:1.0 75:33.653687500000004 #docid=1548edb9-1eca-4c7c-a70b-0870cec0ed9b
1 qid:1 1:0.039783546954169954 ...... 75:0.0 #docid=fb77dc78-b060-4b65-8a9a-5521b22fb28b
1 qid:1 1:0.09670772129479208 ...... 75:0.0 #docid=2a3e5cd7-cdea-47b7-b46c-a90f4343dc47
2 qid:1 1:0.3188044500743293 ...... 75:60.11575 #docid=3c679bda-32fb-425e-88b2-dc393a77dc97
1 qid:1 1:0.318804451902468 ...... 75:60.11575 #docid=156843b3-2eff-4300-ac54-431a9e4299ac
3 qid:1 1:0.31880445508537575 ......  75:60.11575 #docid=9bd32286-1f9c-4945-8303-49291d1e9a83
:
:
2 qid:1 1:0.029685016660102252  ......  75:32.7655625 #docid=8bc8a8fe-2894-47a3-9d51-5cd4a7e4c0d2
4 qid:1 1:0.21006587314392622  ......  75:0.0 #docid=31b0893f-2dc9-4aef-96f4-b411b4a0b34f
```

Figure 5 : Sample feature vectors for query-documents labeled pairs

4.1 PCA Reduced Forest for Transcription Dataset

PCA Reduced Forest algorithm was used to train a ranking model based on using a dimensionality reduction technique (PCA) to a defined bags of features for fields. {F_1: title, F_2: description, F_3 : channel, F_4 : tags, F_5: transcription segments and F_6 : the whole document}. Each bag contains the set of features related to each field from the groups of sixes in Table 1 {*covered query term number, covered query term ratio, fieldLength, IDF (Inverse Document Frequency), TF (Sum of Term Frequency) , TF-IDF , Boolean Model, BM25, LM with Jelinek Mercer Smoothing , LM with Absolute Discounting, LM with Dirichlet*}. The rest of query-independent features {1, 2, 3, 4, 5, 72, 73, 74, 75} from Table 1 are added to have a reduced set of 15 features after the PCA reduction step used to learn the forest model. Results are recorded in Table 2 for$ERR@10$ and Table 3 for $NDCG@10$.

From Table 2, the average$ERR@10$ for using Random Forest for training data is 0.4791, while the average$ERR@10$ for PCA Reduced Forest is **0.4998** with an improvement of 4.32%. The average$ERR@10$ for using Random Forest for the test data is **0.4489** and the average$ERR@10$ for PCA Reduced Forest is slightly lower (0.4486). Higher values for each fold and for either training or test data are shown in **bold** in Table 2. It can be seen from Figure 6 (a),(b) that the results from PCA Reduced Forest are quite similar to those for Random Forest, in some folds PCA Reduced Forest is better and in others the Random Forest is better.

Table 2 : ERR@10 for PCA Reduced Forest vs. Random Forest for transcriptions dataset

Algorithm	**ERR@10 for PCA Forest**		**ERR@10 for Random Forest**	
	Train	Test	Train	Test
Fold 1	**0.4752**	0.507	0.4512	**0.5081**
Fold 2	**0.5205**	0.327	0.5102	**0.3352**
Fold 3	**0.5271**	0.3469	0.5108	**0.3543**
Fold 4	**0.4923**	**0.4807**	0.4769	0.4605
Fold 5	**0.484**	0.5812	0.4461	**0.5863**

From Table 3, The average$NDCG@10$ for training data in Random Forest is 0.7653 and the average$NDCG@10$ for training data when using PCA Reduced Forest is **0.7682** which is slightly better with 0.4% improvement. However, the average$NDCG@10$ for test data in Random Forest is **0.6816**, while the average$NDCG@10$ for test data when using PCA Reduced Forest is 0.679. From Figure 6 (c),(d), it is clear that the results from PCA Reduced Forest are quite comparable to those for Random Forest, in some folds it is better and in others the Random Forest is better. This means that the new set of 15 features projected by using the PCA step almost reserves almost all the information in the 75 features and so produce a simpler ranking model with less parameters.

Table 3 : NDCG@10 for PCA Reduced Forest vs. Random Forest for transcriptions dataset

Algorithm	**NDCG@10 for PCA Forest**		**NDCG@10 for Random Forest**	
	Train	Test	Train	Test
Fold 1	0.7914	**0.5435**	**0.7953**	0.5428
Fold 2	**0.7648**	0.6317	0.7603	**0.6692**
Fold 3	0.7856	**0.5909**	**0.799**	0.5796
Fold 4	**0.7572**	0.7823	0.7396	**0.7918**
Fold 5	**0.7422**	**0.8464**	0.7324	0.8244

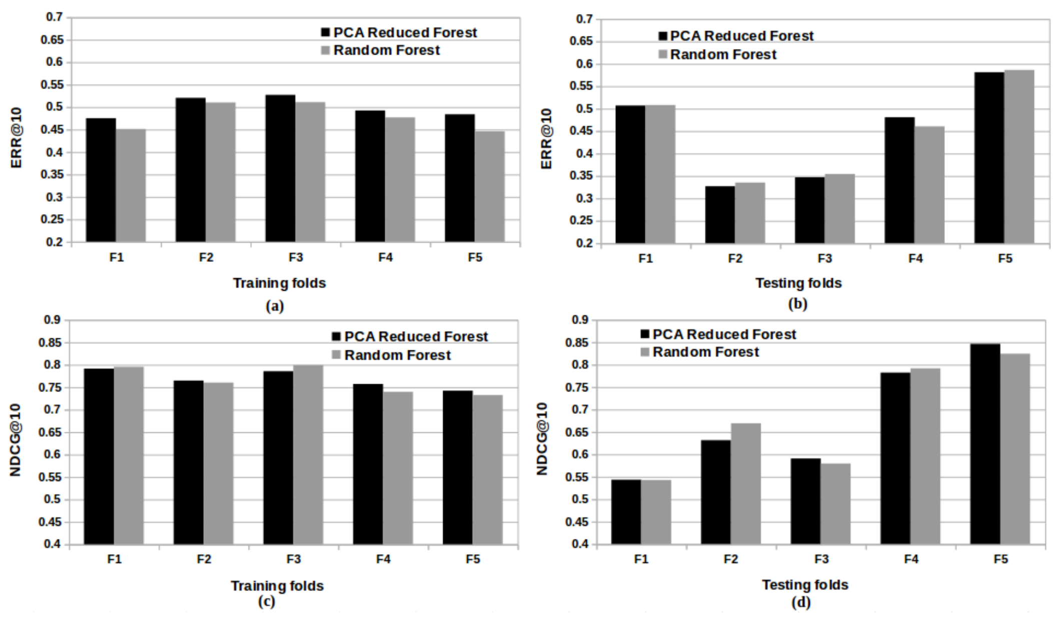


Figure 6 : PCA Reduced Forest vs. Random Forest algorithm for transcriptions dataset

Figure 6 shows that PCA Forest achieves very comparable results to Random Forest in terms of $ERR@10$ and $NDCG@10$ for unseen test data. In addition, the learned PCA Reduced Forest model shows better performance for training data in terms of both the average $NDCG@10$ and $ERR@10$ than the Random Forest model. To evaluate the algorithm more and explore its behavior with larger datasets, we conducted a similar experiment on the Microsoft LETOR dataset [18] with results presented in the next subsection.

4.2 PCA Reduced Forest for Microsoft LETOR dataset

The same algorithm was adapted to use bags for web fields' features and metadata from the Microsoft LETOR dataset [18] which has 136 features to have a total of 14 bags. Using PCA as a dimensionality reduction technique by projecting each bag of features to a new space and use the first principal component as a score for summarizing all the set of features related to every field. Random Forest algorithm is used to train a simpler model for the reduced set of only 14 features.

For the $ERR@10$ evaluation, PCA Reduced Forest shows better improvement for both training and test data for all folds as indicated in **bold** in Table 4. The average $ERR@10$ achieved by PCA Reduced Forest is **0.31586** for training data and **0.31526** for test data. On the other hand, for Random Forest, the average $NDCG@10$ achieved for training data is 0.29226 and 0.29194 for test data. An 8.08% improvement is achieved for training data when using PCA Reduced Forest over using the Random Forest and 7.99% improvement for test data.

Table 4: ERR@10 for PCA Reduced Forest vs. Random Forest for Microsoft LETOR dataset

Algorithm	**ERR@10 for PCA Forest**		**ERR@10 for Random Forest**	
	Train	Test	Train	Test
Fold 1	**0.3129**	**0.3161**	0.2922	0.2936
Fold 2	**0.3151**	**0.3137**	0.2911	0.291
Fold 3	**0.3159**	**0.3138**	0.2918	0.2922
Fold 4	**0.3184**	**0.3128**	0.2929	0.2897
Fold 5	**0.317**	**0.3199**	0.2933	0.2932

From Table 5, it can be found that the average $NDCG@10$ for training data in case of PCA Forest is **0.39776** while it is 0.3914 for Random Forest with 1.62% improvement. For test data, the average $NDCG@10$ is **0.39622** for PCA Reduced Forest and 0.39118 for Random Forest with 1.29% improvement. Figure 7 (c),(d) show this improvement for PCA Reduced Forest over Random Forest for training data and test data respectively.

We can explain this result that the new surrogate scores obtained by PCA represent a new features set with less dependency among them which enhances the results of the Random Forest algorithm that is affected by the dependency and collinearity of features across the trees in the ensemble.

Table 5: NDCG@10 for PCA Reduced Forest vs. Random Forest for Microsoft LETOR dataset

Algorithm	**NDCG@10 for PCA Forest**		**NDCG@10 for Random Forest**	
	Train	Test	Train	Test
Fold 1	**0.4171**	**0.4169**	0.3903	0.3926
Fold 2	**0.3928**	0.3885	0.3904	**0.3893**
Fold 3	**0.393**	**0.3923**	0.3914	0.3922
Fold 4	**0.3927**	0.3899	0.3921	**0.3901**
Fold 5	**0.3932**	**0.3935**	0.3928	0.3917

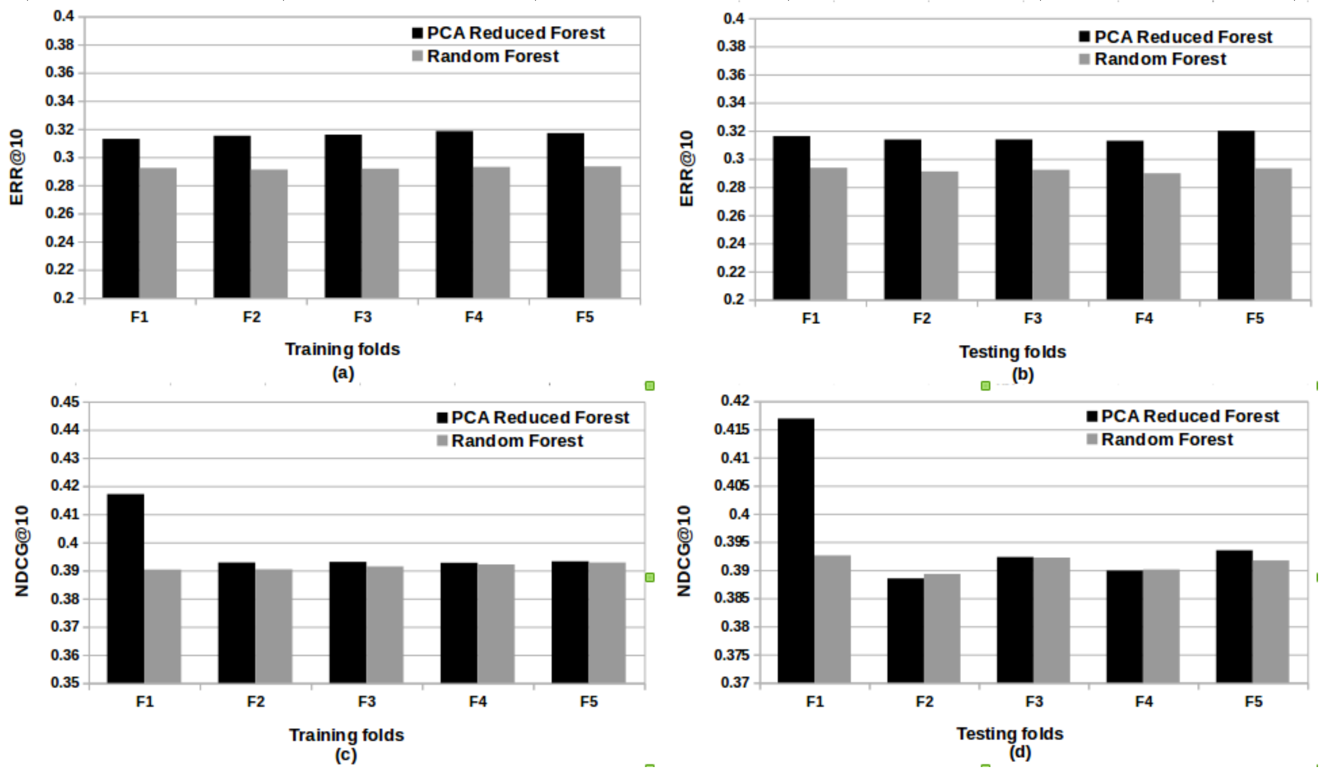


Figure 7: PCA Reduced Forest vs. Random Forest algorithm for Microsoft LETOR dataset

5. CONCLUSION

The proposed PCA Reduced Forest algorithm learns a simpler learning to rank model than the standard Random Forest learned model. It is based on using PCA to get reduced set of principal components of features bags. The learned PCA reduced forest model uses 15 features instead of 75 features for transcriptions dataset and 14 features instead of 136 for Microsoft LETOR dataset. Moreover, the PCA reduced model outperforms the Random Forest learned model especially for$ERR@10$. It is particularly at advantage for large datasets having a lot of features as it is the case in Microsoft LETOR dataset. The only drawback for PCA reduced Forest is that all features still need to be extracted before the reduction step. This is our current research work to only extract a reduced set of features in order to decrease the search response time.

REFERENCES

[1] M. Larson and G.J.F. Jones. Spoken content retrieval: A survey of techniques and technologies. Foundations and Trends in Information Retrieval, 5(4-5):235–422, 2012.
[2] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H. Y. Shum: An Empirical Study on Learning to Rank of Tweets. In Proceedings of the 23rd International Conference on Computational Linguistics. 295-303. Beijing, China, 2010.
[3] J. Chen, C. Xiong, and J. Callan: An Empirical Study of Learning to Rank for Entity Search. In Proceedings of the 39 th International ACM SIGIR conference on Research and Development in Information Retrieval, 737-740. Pisa, Italy, 2016.
[4] X. Zhao, X. Li, and Z. Zhang: Multimedia Retrieval via Deep Learning to Rank. IEEE Signal Processing Letters 22: 1487 – 1491, 2015.
[5] L. Breiman. Random forests. Machine Learning, 45:5–32, 2001.
[6] Q.Wu, C.J.C. Burges, K.M. Svore, and J. Gao. Adapting boosting for information
retrieval measures. Information Retrieval, 13(3):254–270, Jun 2010.
[7] C. J. Burges, E. Renshaw, T. Shaked, A. Lazie, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In Proceedings of ICML, pages 89–96, 2005.
[8] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In Proceeding of ICML, pages 129 – 136, 2007.
[9] R. Herbrich, K. Obermayer, and T. Graepel. Large margin rank boundaries for ordinal regression. Advances in Large Margin Classifiers, pages 115–132, 2000.
[10] T.Y. Liu. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 2009.
[11] Y. Bengio, A. Courville, P. Vincent, "Representation learning: A review and new perspectives", IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, pp. 1798-1828, Aug. 2013.
[12] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied
to ad hoc information retrieval. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 334–342, New York, USA, 2001.
[13] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In Proceedings of the 18th ACM conference on Information and knowledge management, pages 621–630, 2009.
[14] J. Kalervo and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems, 20:422–446, 2002.
[15] Ananth Mohan, Zheng Chen, and Kilian Q Weinberger. Web-search ranking with initialized gradient boosted regression trees. Journal of Machine Learning Research-Proceedings Track, 14:77–89, 2011.
[16] Pierre Geurts and Gilles Louppe. Learning to rank with extremely randomized trees. In JMLR: Workshop and Conference Proceedings, volume 14, 2011.
[17] M. Ibrahim and M. Carman: Comparing Pointwise and Listwise Objective Functions for Random Forest Based Learning-to-Rank. ACM Transactions on Information Systems, Vol. 34:4, 2016.
[18Microsoft learning to rank dataset: https://www.microsoft.com/en- us/research/project/mslr/, last accessed: September 2017.
[19] Verse by verse quran dataset: http://www.everyayah.com, last accessed: July 2017.
[20] D. Weenink and P. Boersma. Praat: doing phonetics by computer: http://www.praat.org/, last accessed: July 2017.
[21] Youtube Data API: https://developers.google.com/youtube/v3/, last accessed: July
2017.
[22] Apache solr 6.0.0 : http://lucene.apache.org/solr/, last accessed: May 2016.