LIMITATIONS AND IMPROVEMENT OPPORTUNITIES FOR IMPLICIT RESULT DIVERSIFICATION IN SEARCH ENGINES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YAŞAR BARIŞ ULU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2019

Approval of the thesis:

LIMITATIONS AND IMPROVEMENT OPPORTUNITIES FOR IMPLICIT RESULT DIVERSIFICATION IN SEARCH ENGINES

submitted by **YAŞAR BARIŞ ULU** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _______________

Assoc. Prof. Dr. İsmail Sengör Altıngövde
Supervisor, **Computer Engineering, METU** _______________

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering, METU _______________

Assoc. Prof. Dr. İsmail Sengör Altıngövde
Computer Engineering, METU _______________

Prof. Dr. Özgür Ulusoy
Computer Engineering, Bilkent University _______________

Date:

ABSTRACT

LIMITATIONS AND IMPROVEMENT OPPORTUNITIES FOR IMPLICIT RESULT DIVERSIFICATION IN SEARCH ENGINES

Ulu, Yaşar Barış
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. İsmail Sengör Altıngövde

December 2019, 49 pages

Search engine users essentially expect to find the relevant results for their query. Additionally, the results of the query should contain different possible query intents, which leads to the well-known problem of search result diversification. Our work first investigates the limitations of implicit search result diversification, and in particular, reveals that typical optimization tricks (such as clustering) may not necessarily improve the diversification effectiveness. Then, as our second contribution, we explore whether recently introduced word embeddings can be exploited for representing documents to improve diversification, and show a positive result. Third, as our detailed analysis reveals that the candidate set size plays a critical role for implicit diversification, we propose to automatically predict the size of the candidate set on *per query* basis. To this end, we use a rich set of features based on the inter-similarity of documents and similarity between queries and documents. Finally, we propose caching similarities of document pairs to improve the processing time efficiency of implicit result diversification.

Keywords: Search Engines, Search Result Diversification, Implicit Result Diversification Methods, Machine Learning, Caching

ÖZ

ARAMA MOTORLARINDA DOLAYLI CEVAP ÇEŞİTLENDİRME İÇİN KISITLAMALAR VE GELİŞME FIRSATLARI

Ulu, Yaşar Barış
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. İsmail Sengör Altıngövde

Aralık 2019 , 49 sayfa

Arama motoru kullanıcılarının arama motorundan temel beklentisi sorgu sonuçlarının yapılan sorguyla alakalı olmasıdır. Buna ek olarak, sorgu sonuçları, sorgunun farklı anlamlarını da barındırmalıdır, ki bu problem literatürde arama sonucu çeşitlendirme şeklinde tanımlanmıştır. Çalışmamız ilk olarak geleneksel dolaylı çeşitlendirme metotlarının kısıtlamalarını incelemekte, ve özellikle kümeleme gibi optimizasyon yöntemlerinin çeşitlendirme performansını iyileştirmeyebileceğini göstermektedir. İkinci bir katkı olarak, dokümanları kelime kodlama tabanlı temsil etmenin çeşitlendirme başarımına etkisi incelenmekte ve bu yaklaşımın olumlu sonuç verdiği gösterilmektedir. Üçüncü bir katkı olaraksa aday küme büyüklüğünün dolaylı çeşitlendirme için kritik rol oynadığı gözleminden hareketle bu parametre için sorgu bazında tahminleme yapılması önerilmektedir. Bu amaçla dokümanlar arası, ve sorgu ile doküman arası benzerlikleri temsil eden zengin bir öznitelik kümesi kullanılmaktadır. Son olarak, dolaylı çeşitlendirmenin verimliliğini artırmak üzere dokümanlar arası benzerlikleri saklayan bir önbellek yapısı önerilmektedir.

Anahtar Kelimeler: Arama Motorları, Arama Sonucu Çeşitlendirme, Dolaylı Sonuç Çeşitlendirme Metotları, Makine Öğrenimi, Önbellek

To my love, Selin

ACKNOWLEDGMENTS

I would first like to thank my supervisor Assoc. Prof. Dr. İ. Sengör Altıngövde with my sincere feelings for his guidance and endless support. Without his knowledge and motivation, our study would not be completed.

I would like to thank my love, Selin Ünal for her endless motivation and support in my graduate study.

I would like to thank my colleagues Mehmet Akçay, Can Ünaldı, Andaç Akarsu and Sena Terzi for their support.

I would also like to thank my parents who always give me the motivation every time I need.

This work is partially funded by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant no. 117E861.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AOL	American Online
BM25	Best Matching-25
DCG	Discounted Cumulative Gain
DF	Document Frequency
IA-Select	Intent-Aware Select
IDF	Inverse Document Frequency
IR	Information Retrieval
k-NN	k-Nearest Neighbors
MMR	Maximal Marginal Relevance
MSD	Max-Sum Dispersion
NIST	National Institute of Standards and Technology
NQC	Normalized Query Commitment
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
TREC	Text Retrieval Conference
WE	Word Embedding
xQuAD	Explicit Query Aspect Diversification
α-nDCG	α Weighted Normalized Discounted Cumulative Gain

CHAPTER 1

INTRODUCTION

1.1 Motivation

One of the most popular web search engines, Google, handles over 6 billions of queries per day[1]. Statistically, 84% of the submitted queries contain 3 or less number of words[2], and a certain fraction of these queries are likely to be too short to represent the real underlying intent of the user. Such queries, called as ambiguous or multi-faceted, can be interpreted in different ways [1]. A typical example of an ambiguous query is the query 'Apple', which can be interpreted either as a product, a stock exchange, a fruit, or a street, etc. That is, the user's intention may be retrieving the price of Apple smart phones or computers, or learning about the types of apple fruit in the world. For such queries, web search engines should provide search results that cover as many different meanings of the query as possible.

The necessity of covering possible distinct meanings of the query leads to the search result diversification problem. Generally, this problem is handled by re-ranking an initially retrieved set of documents for a query to provide both relevance and diversity of the results. The search result diversification problem is an instance of the maximum coverage problem which is a typical NP-hard problem [2]. Therefore, several approximate approaches are proposed in the literature in order to handle this problem [2].

Search result diversification methods, namely implicit result diversification and explicit result diversification, try to satisfy user expectation in case of ambiguous or

[1] `https://www.internetlivestats.com/google-search-statistics/`
[2] `https://www.keyworddiscovery.com/keyword-stats.html`

multi-faceted queries [3]. Implicit result diversification methods essentially try to cover different interpretations of a query by taking the inter document similarities into account. On the other hand, explicit result diversification methods employ predefined intents (e.g., sub-queries) of the query in order to cover all aspects of the query.

In this thesis, we concentrate on the implicit result diversification, scrutinize possible limitations of the methods in this category and propose solutions to overcome these limitations.

1.2 Problem Definition and Contributions

By definition, implicit result diversification methods exploit the information (e.g., document content and features) extracted from the initial set of retrieved documents. That is, these methods perform re-ranking of the retrieved set of documents without any external knowledge of the query aspects. A typical implicit search result diversification method contains three main components, such as the similarity function, candidate document set and trade-off parameter. First one is the function that calculates the similarity between document pairs. Second one is the set of documents to be re-ranked. And the last one is the parameter balancing the relevance and diversity of the final ranking. In this thesis, we aim to improve the implicit diversification performance by exploring better solutions to determine these key components.

To this end, our contributions are listed as follows:

- We first explore the impact of a well-known optimization (e.g., [3]), namely, clustering, on the performance of a representative implicit diversification method, i.e., Maximal Marginal Relevance (MMR) [4]. Our experiments reveal that clustering of candidate documents in such a scenario does not help, but may even diminish the diversification performance with respect to the baseline non-diversified ranking.
- To improve implicit diversification performance, we propose to represents candidate documents based on the recently proposed word embeddings [5, 6]. Our

experiments show that using the latter representation is superior to representing documents based on the traditional vector space model (with tf-idf weights).

- We analyze the diversification performance per query by varying the candidate document set size and trade-off parameters for MMR method, and show that the best performing parameters considerably differ for different queries (verifying a similar finding for the trade-off parameter in [7]). Thus, we propose to predict the candidate set size on a *per query* basis, to achieve a more customized diversification of query results. To this end, we employ a rich set of features that capture the retrieval effectiveness (i.e., query performance predictors [7, 8, 9, 10] and pairwise similarity of documents (using alternative document representations).

- Implicit result diversification algorithms typically calculate pairwise similarity of the documents in the candidate set, which is a costly operation to be conducted on the fly. In order to improve the processing efficiency, we propose to use a cache of such pairwise similarity scores.

Note that, a portion of our work presented in this thesis is accepted for publication in European Conference on IR Research (ECIR 2020) with the title "Predicting the Size of Candidate Document Set for Implicit Web Search Result Diversification".

1.3 Organization of the Thesis

In Chapter 2, we provide background information on search result diversification methods as well as the basics of document representation and caching for search. In Chapter 3, the details of the experimental setup and evaluation metrics used throughout the thesis are explained. In Chapter 4, we analyze implicit search result diversification to detect possible limitations, and evaluate the impact of an alternative document representation based on word embeddings on the diversification performance. In Chapter 5, we focus on the performance of implicit diversification with respect to the trade-off and candidate document set size parameters, and describe our strategy to predict the candidate set size. Two basic strategies for employing a cache of pairwise document similarities during implicit search result diversification is presented in Chapter 6. Finally, in Chapter 7, we conclude and point to future research directions.

CHAPTER 2

BACKGROUND INFORMATION

In this chapter some background information is provided in detail. Firstly, search result diversification is defined, and then some approaches of search result diversification is explained. Later we give details of some methods which are examined throughout our experiments.

2.1 Search Result Diversification

As explained before in Chapter 1, search result diversification is re-ranking result document set which are retrieved by their relevance scores. Many have acknowledged the search result diversification as an significant problem [11], [12]. Especially developing a model which rank documents on the basis of combination of relevance score to a query and diversity is one of the crucial problem in search result diversification [1]. In early stages of evolution of result diversification, common retrieval methods which are rests upon Probabilistic Ranking Principle [13] suppose that relevance of documents is irrelevant from others in the collection. On the contrary, as a part of definition of result diversification, relevance specifies not only cohesion between given a query and a selected document but also cohesion between a document and other retrieved documents [3]. As a result of this, most of the diversification techniques have tried to stabilize similarity of query-document pairs for relevance and dissimilarity of document-document pair for diversity in order to achieve result diversification.

Search result diversification techniques was categorized as implicit or explicit [14]. Implicit search result diversification ones cope with diversification problem by content of retrieved document list [4], [15], [16], [17]. On the other hand, explicit search result diversification methods rely on aspects of queries which are externally generated using evidence [14], [18].

One of the earliest example of implicit result diversification methods is Maximal Marginal Relevance (MMR) is proposed by Carbonell and Goldstein in [4]. In their work, two functions, one is for calculating similarity between document and query, the other is for calculation the dissimilarity between among documents are combined together in order to calculate marginal score. At each iteration the document with highest MMR score added to result set. The score can be calculated as follows:

$$MMR(D_i) = \operatorname*{argmax}_{D_i \in R} \left[\lambda Sim_1\Big(D_i, Q\Big) - (1-\lambda) \max_{D_j \in S} Sim_2\Big(D_i, D_j\Big) \right] \tag{2.1}$$

where D stands for a document, Q for a query, S for set of documents that have been selected so far, R for set of candidate documents to be selected. Sim_1 is the similarity between query and document, which also indicates relevance score of document for a given query, Sim_2 is the similarity between two documents. Lastly λ is the trade-off parameter which tunes the relevance score of document and document-document similarity.

Another example of implicit result diversification methods is Max-Sum Dispersion (MSD) is proposed by Gollapudi and Sharma in [19]. It s a greedy algorithm firstly introduced in [20] as a solution to Max-Sum Dispersion Problem. In this algorithm, pair of documents with the largest score are added to result set and formulated as follows:

$$MSD(Di, Dj) = \operatorname*{argmax}_{D_i, D_j \in R} \left[(1-\lambda)\Big(Sim(D_i, Q) + Sim(D_j, Q)\Big) + 2\lambda Div(D_i, D_j) \right] \tag{2.2}$$

where Sim is the similarity between query and document, which is same as in MMR. Samely in MMR, R stands for candidate document set and λ is the trade-off parameter for balance. Different from MMR, Div stands for distance function that calculates dissimilarity between pair of documents.

Another implicit method, namely Sy, which is a simple algorithm proposed in [21] for detecting duplicate and near-duplicate tweets. In brief, Sy moves from top to bottom in candidate document set R and compares pairwise similarity of documents with a predefined threshold value. For a document D_i, if similarity score with D_j, where D_j follows D_i in the ranking in candidate document set R, is greater than the threshold, then D_j is removed from candidate document set R.

In the examples of implicit search result diversification methods, there exists only the information about initially retrieved result set and some details like relevance score, content of documents and queries etc.

In contrast to implicit search result diversification methods, explicit ones benefit from query aspects which are explicitly created by using a taxonomy of information or query logs [18], [22]. Intent-Aware Select (IA-Select) method, one of the explicit search result diversification method, supposes that both queries and documents are classified in compliance with this taxonomy [18]. In each iteration of algorithm, a document is selected according to higher scores which means that it covers different subtopic of a query. The presentation of the scoring function as follows:

$$IA-Select(Q, D_i) = \sum_{Q_a \in Q} P(Q_a|Q)V(D_i|Q, Q_a)) \prod_{D_j \in R} (1 - V(D_j|Q, Q_a)) \quad (2.3)$$

where Q_a is an aspect of query Q, D_i is candidate document to be selected and D_j is the document already selected. Furthermore $P(Q_a|Q)$ and $V(D_i|Q, Q_a)$ are the probability of query aspect Q_a for the query Q and the likelihood of document D_i relevant to the query Q in terms of query aspect Q_a, respectively.

Another example of explicit method is eXplicit Query Aspect Diversification (xQuAD) framework based on probability of relevance and diversity [23]. It also uses query aspects that are acquired by Text REtrieval Conference (TREC) subtopics. This method's scoring function is as follows:

$$xQuAD(Q,D_i) = (1-\lambda)P(D_i|Q) + \lambda \sum_{Q_a \in Q} \left[P(Q_a|Q)P(D_i|Q_a) \prod_{D_j \in R} (1-P(D_j|Q_a))) \right] \tag{2.4}$$

where $P(D_i|Q)$ and $P(D_i|Q_a)$ represents relevance score of candidate document D_i to the query Q and query aspect Q_a respectively, $P(Q_a|Q)$ is the probability of the aspect Q_a for the query Q, $P(D_j|Q_a)$ represents likelihood of the document D_j which is already selected in result set in terms of query aspect Q_a. Lastly λ is the trade-off parameter for balancing the relevance and the diversity measurements.

2.2 Document Representation Methods

Representation of a document plays a key role in implementing implicit result diversification in order to calculate similarity measurement between pair of documents. Basically, we use two main representation of a document namely, TF-IDF (Term Frequency - Inverse Document Frequency) and word embedding model.

2.2.1 TF-IDF Model

TF-IDF often used in Information Retrieval (IR) is meant to represent the importance of a term for a document in a corpus [24]. Furthermore, search engines often use sort of TF-IDF schema as a key instrument in retrieving documents as a result of a query. TF-IDF can be computed as follows:

$$TF - IDF(t_i, d) = TF(t_i, d) \cdot IDF(t_i, d) \tag{2.5}$$

where t is for a term and d is for a document includes term t . $TF(t_i, d)$ and $IDF(t_i, d)$ represent Term Frequency and Inverse Document Frequency respectively, and formulations are as follows:

$$TF(t_i, d) = \frac{|\{t_i \in d\}|}{|d|} \tag{2.6}$$

$$IDF(t_i, d) = \log \frac{|C|}{|\{d \in C : t_i \in d\}|} \tag{2.7}$$

where C stands for a collection or corpus.

In our experiments, the documents are symbolized using TF-IDF model. The illustrations of some documents as follows:

$d_1 = \big[(t_1, tfidf_{t_1}), (t_4, tfidf_{t_4}), (t_5, tfidf_{t_5}), (t_7, tfidf_{t_7})\big]$
$d_2 = \big[(t_1, tfidf_{t_1}), (t_2, tfidf_{t_2}), (t_3, tfidf_{t_3}), (t_4, tfidf_{t_4})\big]$
$d_3 = \big[(t_2, tfidf_{t_2}), (t_3, tfidf_{t_3}), (t_7, tfidf_{t_7}), (t_8, tfidf_{t_8})\big]$

Calculation similarity measurement between pair of documents are simply done by dot product with the help of representing document as a vector of TF-IDF values.

2.2.2 Word Embedding Model

Word embedding, which is also word representation, aims to group words having characteristics in common. This representation is used for the first time in 1986 [25]. In recent years, Skip-gram model, which is an efficient way in which words can be learned as a vector representation from text, is introduced [6]. Later, Glove [26] which has higher performance than previous methods is presented.

Each word is represented as a D-dimensional vector in word embedding and each dimension of a vector indicates how relevant a word is to specific feature. A small example of word embedding vector with 8-dimension is shown in Figure 2.1. According to the figure, "broccoli" and "tomato" have a common feature saying both are vegetable are similar to each other, on the other hand "cat" is different from both.

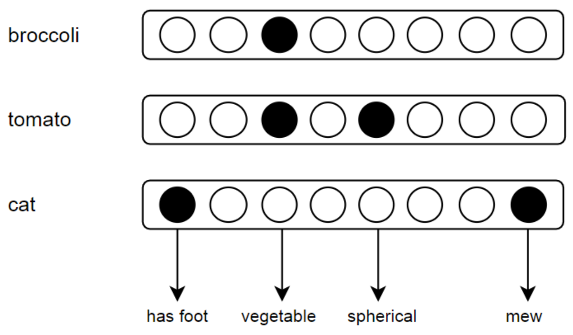

Figure 2.1: Visualization of word embedding vector.

The key point in this model is represent document as a vector with fixed D-dimension. In our experiments we use several approaches (e.g., see [27]) while representing documents via word embeddings, listed as follows:

1. Calculating average of term's word embedding
2. Finding minimum valued features of word embedding
3. Finding maximum valued features of word embedding
4. Finding minimum-maximum valued features of word embedding and concatenating
5. Calculating average of term's word embedding multiplied by TF-IDF values of the term.

We provide a visualization of some of these approaches as follows.

Document representation:
$D : [w_1, w_2, w_3]$

Word embedding representation of terms:
$w_1 : [0.50, 0.20, 0.90]$
$w_2 : [0.90, 0.60, 0.10]$
$w_3 : [0.10, 0.50, 0.70]$

TF-IDF representation of terms:

TF-IDF$_{w_1} = 0.20$

TF-IDF$_{w_2} = 0.60$

TF-IDF$_{w_3} = 0.30$

Table 2.1: Representation of a document as a word embedding.

Method	Document as a Word Embedding
1. Avg. of term	$D : [0.50, 0.43, 0.56]$
2. Min	$D : [0.10, 0.20, 0.10]$
3. Max	$D : [0.90, 0.60, 0.90]$
4. Min-Max Concat.	$D : [0.10, 0.20, 0.10, 0.90, 0.60, 0.90]$
5. TF-IDF Avg.	$D : [0.19, 0.18, 0.15]$

2.3 Predicting Parameters of the Result Diversification Methods

Several diversification methods need to tune a trade-off parameter (aimed to balance relevance versus diversity) in the range [0,1] to achieve its best performance. Normally, tuning of this trade-off parameter can be done uniformly on a query set in order to obtain the higher diversification performance. However, since different queries may exhibit a different level of ambiguity, Santos et al. [7] suggested a selective diversification approach, where the trade-off parameter is predicted for each query.

In addition to examining trade-off parameter, initial candidate set size, which is another key parameter, has an impact on diversification performance. In [28], preliminary experiments for candidate set size prediction is presented for explicit diversification. In contrary, this thesis addresses implicit diversification, which requires features that capture inter-document similarity that are not used in the latter work. Furthermore, we predict both the candidate set size and trade-off parameters, which is different than the setup of the previous work. That is, given an initial retrieval result for a query, we estimate the candidate set size and trade-off parameter on the basis of similar queries that has been processed before.

In our work, we employ k-NN [29] algorithm for prediction (as in [7]), which is a lazy learning approach. We choose k-NN for simplicity,effectiveness and ease of use in spite of some disadvantages such as computational complexity and significant memory requirements.

2.4 Query-Document Relevance Score Calculation

Query-document relevance score which is an important part of implicit diversification methods are calculated using Okapi BM25 ranking function [30] in our experiments. Okapi BM25 ranking function is as follows:

$$BM25(Q,D) = \sum_{Q_i \in Q} IDF_{Q_i} \times \frac{tf_{Q_i,D} \times (k_1 + 1)}{k_1 \times (1 - b + b \times \frac{|D|}{avdl}) \times tf_{Q_i,D}} \tag{2.8}$$

where $tf_{Q_i,D}$ is frequency of term Q_i in document D, $avdl$ stands for average length of document in corpus, b and k_1 are free parameters, usually chosen as 0.75 and 1.2 respectively, and IDF_{Q_i} stands for inverse document frequency [31] and usually calculated as follows:

$$IDF(Q_i) = log\frac{N - N_{Q_i} + 0.5}{N_{Q_i} + 0.5} \tag{2.9}$$

where, N is the size of the corpus and N_{Q_i} is the document count containing term Q_i in corpus.

2.5 Document-Document Similarity Calculation

Other important part of implicit diversification methods is similarity score between pair of documents. In our experiments, we implement several similarity functions, namely, cosine similarity, euclidean similarity and jaccard similarity, and they are formulated as follows:

$$CosineSimilarity(X,Y) = \frac{\sum\limits_{i=1}^{N} X_i \cdot Y_i}{\sqrt{\sum\limits_{i=1}^{N} X_i \cdot X_i} \times \sqrt{\sum\limits_{i=1}^{N} Y_i \cdot Y_i}} \tag{2.10}$$

$$EuclideanSimilarity(X,Y) = 1 - \sqrt{\sum_{i=1}^{N} X_i \cdot Y_i} \tag{2.11}$$

$$JaccardSimilarity(X,Y) = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} \tag{2.12}$$

2.6 Caching in Web Search Engines

As a research topic, caching has been studied for a long time by many researchers [32]. In spite of there exists many studies about web caching, researchers have had an attention to caching in search engines for only a decade [33]. So long as the amount of queries increases over time, significant performance gains and resource savings are obtained by caching strategies in search engines.

In early stages of caching, the idea of caching was that the most frequent or recent data have been stored in devices which have high volume and slow data storage. Afterwards, devices with low volume and fast data storage have been used in caching. In some caching techniques, this idea is accomplished with some specific information and offline preprocessing.

In a typical large scale web search engine, there exists five different cache types according to their contents. These are, namely result, score, intersection, list and document. The contents of the different types of typical search engine caches are shown in Table 2.2.

Caches in search engine can be categorized as static [33], [34], [35] and dynamic [33], [35] depending on their abilities and capacities. In static caches, data items

Table 2.2: Different types of caching.

Cache Type	**Content**
1. Result Cache	Query Results
2. Score Cache	Precomputed Scores
3. Intersection Cache	Intersections of Posting Lists
4. List Cache	Posting Lists
5. Document Cache	Document Content

which will be cached are acquired from frequently accessed before and the content of the caches remains unchanged until the following regular update [36]. On the other hand, in dynamic caches the recent data items stored and are changed dynamically according to recently accessed data.

In our work, we will develop a static cache for implicit search result diversification. Our cache will store the similarity scores of document pairs generated from top-k results of a large number fo queries. As the performance metrics, efficiency (time) and effectiveness (performance of diversification) are investigated through experiments on caching.

CHAPTER 3

EXPERIMENTAL SETUP

In this chapter, we present the details of the data sets used in our experiments in order to analyze implicit diversification methods. Also, evaluations metrics that are used for comparing our results to baseline results are explained.

3.1 Data Set

We employ TREC collections that belong to years 2009 to 2012 in our experiments. Brief details of internal data files based on these data sets are as follows:

- **Query Set:** It contains queries that are performed by search engine. It has a simple and short queries such as *"obama family tree"*, *"horse hooves"*, *wedding budget calculator"*, etc. The number of queries in each data set is provided in Table 3.1.

- **Query Results:** It contains results of queries that are sorted based on a ranking model. In our cases, Okapi BM25 function is used as retrieval and ranking function. In each year, there exists approximately 1000 documents per query. In our experiments, we use discrete number of documents in retrieved results per query in the [10,100] interval. Furthermore, the scores of retrieved documents obtained from BM25 function and document id's are existed in the data. A structure of this data file is shown in Table 3.2.

- **Document Contents:** Documents retrieved as a result of query contain bag

Table 3.1: Number of queries in TREC data sets.

TREC Data Set	Number of Queries
2009HM	50
2010HM	48
2011HM	50
2012HM	50
2009TREC	50
2010TREC	50
2011TREC	50
2012TREC	50

Table 3.2: Query results in TREC data sets

Query Id	Document Id	Score of a Document
$query_i$	$document_j$	$score\_of\_document_j$
$query_i$	$document_k$	$score\_of\_document_k$
$query_i$	$document_l$	$score\_of\_document_l$
$query_i$	$document_m$	$score\_of\_document_m$

of words, in order words, terms. This data is composed of some term details such as a query id, rank of a document including the term, term id and term frequency (TF) as shown in Table 3.3.

Table 3.3: Document contents in TREC data sets.

Query Id	Rank of a Document	Term Id	Term Frequency
$query_i$	$rank_j$	$term_k$	TF_term_k
$query_i$	$rank_j$	$term_l$	TF_term_l
$query_i$	$rank_j$	$term_m$	TF_term_m
$query_i$	$rank_j$	$term_n$	TF_term_n

Table 3.4: Term details in TREC data sets.

Term as a string	Document Frequency	Inverse Document Frequency
$term_i$	DF_i	IDF_i
$term_j$	DF_j	IDF_j
$term_k$	DF_k	IDF_k

- **Term Details:** Term details such as textual representation of a term, document frequency (DF) and inverse document frequency (IDF) are included in this data. Meanwhile, the order of terms is equal to term id. Simply, DF represents the number of documents containing the term and IDF represents a measurement of the amount of information provided by the term, as shown in Table 3.4.

- **Relevance Information:** It contains relevance judgments information that indicates a document is relevant or non-relevant with respect to given query. Our performance metrics which are explained later in this chapter are generated using relevance information.

3.2 Evaluation Metric

We calculate α-nDCG@k [37] which is advanced version of nDCG [38] with using α is equal to 0.5 in order to measure the diversification performance, where k is cut-off value, used always as 10. This process is performed by the aid of `ndeval` software[1] provided by NIST. `ndeval` uses diversified result file and relevance information as inputs. Then, it generates metrics for each query and mean of those.

[1] `https://trec.nist.gov/data/web10.html`

CHAPTER 4

ANALYZING IMPLICIT RESULT DIVERSIFICATION METHODS

In this chapter, we share experiment outcomes of implicit result diversification methods with several settings such as representing document as $TF-IDF$ vector or Word Embeddings and calculating similarity measurement between pair of documents. In addition to that, some possible optimizations are made to improve diversity. So, we compare results of implemented diversification methods to non-diverse results according to evaluation metrics such as α-nDCG@10.

Candidate set size, which is one of key parameter of implicit result diversification methods, is used with fixed value. Top 100 documents from initial result set are selected as a candidate set for diversification. The other key parameter, trade-off value (λ) for MMR and threshold value (β) for SY is increased with 0.05 between values 0 and 1 for each diversification. Best performing trade-off (λ) and threshold (β) values are presented for each TREC runs in experiment results.

First of all we look at the diversification performance of MMR algorithm with representing document as a $TF-IDF$ vector belonging to its terms. For each run, left side of MMR which is query-document relevance score is used as in TREC data set and right side which is penalize the relevance score according to document-document similarity calculated with cosine similarity. The problem behind this setup is left and right side of MMR equation is not in same range. Because of cosine similarity by nature is between 0 and 1, we normalize query-document relevance scores by Virtual Document Normalization [39]. By this settings, experiment results are summarized in Table 4.1. Our first attempt shows that MMR is under-performing contrary to findings in [3] except for 2010HM run.

Table 4.1: MMR results using TF-IDF vector of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	0.2360	0.95
2010HM	0.2427	**0.2461**	0.80
2011HM	0.4680	0.4581	0.95
2012HM	0.3218	0.2911	0.95

In addition to MMR, SY algorithm is employed for analyzing with same setup and results of each run are shown in Table 4.2. Under same circumstances such as candidate set size and similarity function, SY algorithm has marginal effect on all runs in contrast to MMR.

Table 4.2: SY results using TF-IDF vector of documents with best α-nDCG@10 scores and threshold (β) values for each run.

Run ID	Initial	Diversified	Threshold (β) Value
2009HM	0.2520	**0.2564**	0.80
2010HM	0.2427	**0.2534**	0.70
2011HM	0.4680	**0.4687**	0.45
2012HM	0.3218	**0.3273**	0.90

In our first attempt in MMR, we realize that documents which are irrelevant and in lower ranks are selected in our result set which brings unexpected results. In second one, we try to promote documents that are possibly relevant and in higher ranks.

Our second attempt is to shrink effects of similarity part (right side of MMR) on MMR score with checking initial ranks of already selected documents. Instead of penalizing MMR score with exact similarity score, we multiply similarity score by a formula which is shown in Eq. 4.1 based on initial rank of document which is in result set. In this way, documents which are similar to top ranked documents in result set can be more eligible. We call this optimization as DCG calculation, since the formula is similar to DCG [38].

$$MMR(D_i) = \underset{D_i \in R}{\text{argmax}} \left[\lambda Sim_1\Big(D_i, Q\Big) - (1-\lambda) \max_{D_j \in S} \left[\frac{1}{\log_2(10-i)+1} Sim_2\Big(D_i, D_j\Big) \right] \right] \tag{4.1}$$

In Eq. 4.1, k represents the rank of document D_j from initial result set. Results of DCG calculation method in MMR is shown in Table 4.3. A trivial improvement is achieved in comparison with original MMR results in Table 4.1. It shows that diversification performance is still insufficient by comparison with the baseline of runs.

Table 4.3: MMR results using DCG calculation and TF-IDF vector of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	0.2464	0.95
2010HM	0.2427	0.2405	0.95
2011HM	0.4680	0.4666	0.95
2012HM	0.3218	0.3181	0.75

As in proposed framework by He et al. [3], we try to cluster initial retrieved documents in order to bring similar documents together. And then, a simple cluster ranker is implemented according to documents which each cluster contains. After selecting documents for the candidate set from clusters that scored highly with cut-off T, MMR as a diversification method is performed on selected candidate set. The overall process is represented as in Fig 4.1. The idea behind selecting documents from top ranked clusters is that we may eliminate irrelevant documents from initial retrieval list.

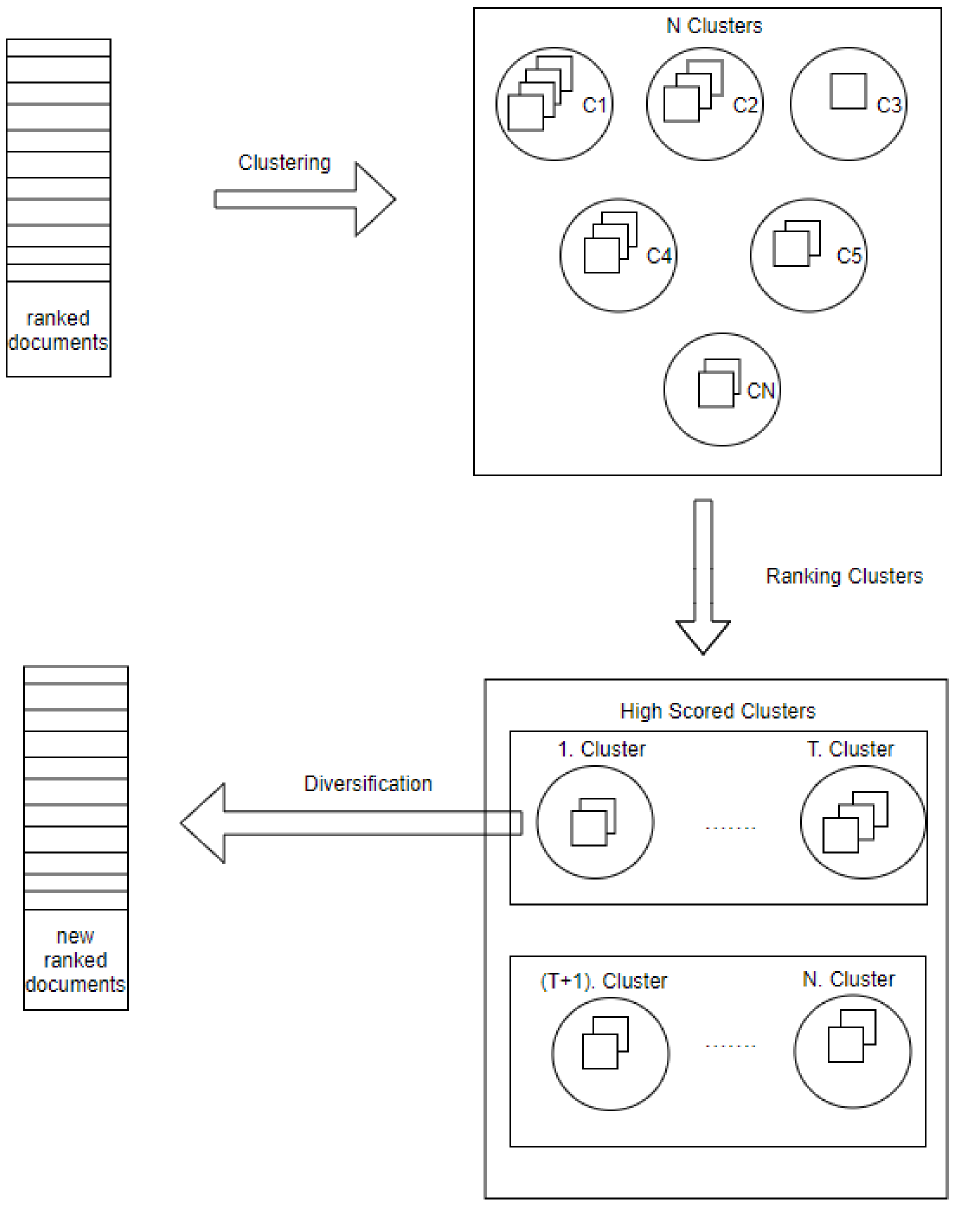

Figure 4.1: Visualization of clustering framework.

In order to cluster documents, we adapt K-means Clustering [40] that uses document contents as in vector space. At the beginning, predefined number of clusters are created, in our work we select as 10. And then, randomly selected 10 documents from initial candidate set are assigned to clusters. Algorithm iteratively select and assign documents to closest center of cluster and after each iteration center of clusters is updated. Algorithm iterates until convergence or reaching maximum number of

Table 4.4: MMR results using clustering framework and TF-IDF vector representation of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	0.2371	0.95
2010HM	0.2427	0.2341	0.90
2011HM	0.4680	0.4474	0.90
2012HM	0.3218	0.2954	0.95

iteration defined before. In ranking step, each cluster score is calculated by averaging the scores of the documents that the cluster includes. And finally, the documents which are obtained from top T ranked clusters are used as candidate document set. It is because K-Means clustering performs poorly, especially several clusters contain one or two documents, that we select T as 5.

Experiment results in Table 4.4 shows that, MMR algorithm with Clustering Framework underperforms compared to pure MMR which is shown in Table 4.1 as well as in contrary to literature. Possible problems with adaptation of Clustering Framework are deciding number of clusters, selecting center of clusters randomly and cut-off T value for eliminating clusters with lower scores. It could be said that a lots of tuning are needed for improving diversification performance with Clustering Framework.

Since representing document as a *TF-IDF* vector could not be powerful for implicit result diversification methods, we try another document representation called Word Embedding. As explained in previous chapters, words or terms are represented as a D-dimensional vector. We select 100-dimensional Glove model for our experiments. Several representation methods such as averaging term word embeddings, selecting min values or max values from word embeddings, concatenating min-max values from word embeddings and averaging term word embeddings which are multipled by own *TF-IDF* values. All representations are tested on experiments and results are given below:

Table 4.5: MMR results using average word embedding of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	0.2214	0.95
2010HM	0.2427	0.2321	0.80
2011HM	0.4680	0.4267	0.85
2012HM	0.3218	0.2971	0.65

Table 4.6: MMR results using minimum word embedding of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	**0.2593**	0.65
2010HM	0.2427	**0.2437**	0.35
2011HM	0.4680	**0.4684**	0.90
2012HM	0.3218	**0.3222**	0.90

Table 4.7: MMR results using maximum word embedding of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	**0.2523**	0.70
2010HM	0.2427	0.2425	0.20
2011HM	0.4680	**0.4708**	0.60
2012HM	0.3218	0.3204	0.95

Table 4.8: MMR results using min-max concatenation word embedding of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	**0.2531**	0.70
2010HM	0.2427	**0.2573**	0.60
2011HM	0.4680	**0.4693**	0.75
2012HM	0.3218	0.3215	0.95

Table 4.9: MMR results using TF-IDF mean word embedding of documents with best α-nDCG@10 scores and trade-off (λ) values for each run.

Run ID	Initial	Diversified	Trade-off (λ) Value
2009HM	0.2520	0.2413	0.95
2010HM	0.2427	**0.2461**	0.95
2011HM	0.4680	**0.4693**	0.90
2012HM	0.3218	0.3060	0.95

Except from representing document as word embedding with averaging all terms' word embeddings which is shown in Table 4.5, MMR with word embeddings yields better diversification performance than representing document as a *TF-IDF* vector. In word embedding representation, due to the fact that simply getting an average of all word embeddings leads to lose information about documents, the results have no change to have better performance than other approaches such as min, max, min-max concatenate, tf-idf mean. Among all other approaches, diversification performance totally depends on used data set. For example, highest diversification performance for 2009HM and 2012HM data set is achieved by representing document as minimum valued word embeddings and results of them are shown in Table 4.6. For 2010HM data set, best diversification performance obtained by using minimum-maximum concatenation of word embeddings represented in Table 4.8. Lastly, maximum valued word embedding representation of a document is beneficial for diversification performance on 2011HM data set.

For SY algorithm, the problem arises from representing document as mean, minimums, maximums and minimums-maximums concatenation of word embeddings. In these four approach, similarity measurement of pair of documents are so close that it is almost 1. Those representations make it impossible use SY algorithm which only depends of similarity thresholds. According to our experiments, threshold values purely in range $[0.970, 0.1]$ make some sense of using it, unfortunately it yields poor performance. On the other hand, representing documents as tf-idf mean of word embeddings is more sensitive to use SY algorithm. The experimental results of the approach is shown in Table 4.10. However, it could be stated that SY algorithm with word embedding have no remarkable effects on diversification performance.

Table 4.10: SY results using TF-IDF mean word embedding of documents with best α-nDCG@10 scores and threshold (β) values for each run.

Run ID	Initial	Diversified	Threshold (β) Value
2009HM	0.2520	0.2363	0.95
2010HM	0.2427	**0.2463**	0.95
2011HM	0.4680	0.4551	0.95
2012HM	0.3218	0.3115	0.95

In contrast to findings in literature, diversification performance of implicit result diversification methods is not efficient as we expect. Observing previously produced results of implicit result diversification methods shows us better diversification performance can be obtained by selecting trade-off (λ) value for each query as proposed in [7]. The representation of optimal trade-off (λ) value that yields best α-nDCG@10 metric for each query in MMR with *TF-IDF* document representation for 2010HM data set is shown in Fig. 4.2.

One another issue about poor performance of implicit result diversification is that ratio of relevant documents in candidate set is relatively decreasing with increasing candidate document set size. It means that finding relevant and diverse documents from candidate document set becomes much harder when candidate set is large enough. The ratio of relevant documents in candidate set for all data sets used through our experiments is shown in Fig. 4.3

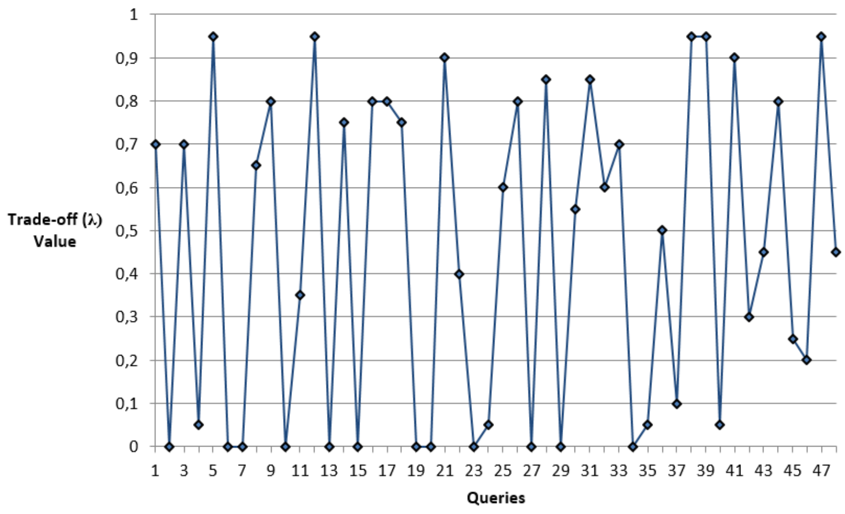

Figure 4.2: Visualization of optimal trade-off value for each query in 2010HM data set.

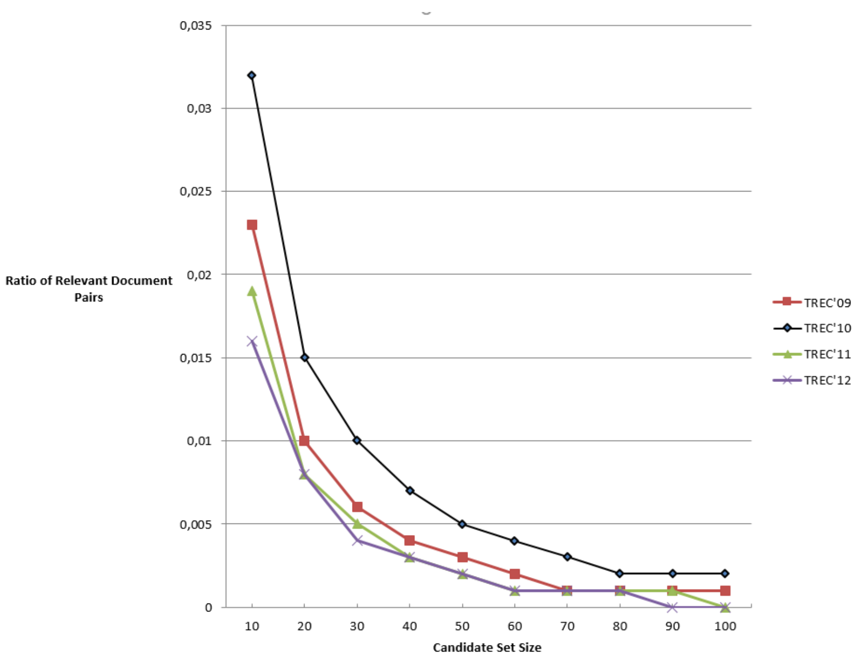

Figure 4.3: Ratio of relevant documents in candidate set for all data sets.

In addition to examining impact of trade-off (λ) parameter on diversification performance for each query, varying candidate set size per query has better performance than fixed candidate set size for explicit search diversification as proposed in [28]. Our experiments which is conducted with altering candidate set size and trade-off

(λ) parameter shows that much more diversification performance can be achieved by selecting optimal values of both two key parameters belonging to implicit result diversification. In next chapter, we will try to find optimal values of those by learning and predicting methods.

CHAPTER 5

PREDICTING IMPLICIT RESULT DIVERSIFICATION PARAMETERS

As demonstrated in previous chapter, distinct usage of trade-off (λ) parameter changes the diversification performance significantly. In this chapter, we will investigate the performance of implicit search result diversification methods by analyzing both parameters, namely trade-off (λ) and candidate set size with changing values.

In this chapter, firstly diversification performance for each query with changing trade-off (λ) value and candidate set size will be shown. Then we will explain several features that are used through learning process in order to predict both of the parameters.

5.1 Analyzing Candidate Set Size and Trade-off (λ) Value Selection

Since the input of implicit result diversification method is only initially retrieved result set, it can be said that candidate set which is a sub set of initial result set directly influence on how diversification performs. In order to show this point, MMR as a implicit diversification method is applied on four different data sets with different baselines. Candidate set size is varied through experiments from 10 to 100 with an step size 10 and lambda is valued from 0.05 to 1.0 as in previous chapter.

As shown in Table 5.1 with detailed scores for 2009HM data set obtained by MMR method with word embeddings, candidate set sizes and trade-off (λ) parameters that performs best changes for each query. (The queries have baselines with 0 scored are excluded from the table) It can be said that performance of the implicit result diversification can be improved considerably when optimum trade-off and candidate set size parameters are predicted correctly.

Table 5.1: Best α-nDCG@10 scores of each query with best trade-off (λ) value and candidate set size. The scores that exceeds baseline scores is highlighted and fail ones are italicized.

Query ID	Candidate Set Size	Trade-off (λ) Value	α-nDCG@10
1	10	0.05	**0.6767**
2	100	0.05	**0.1826**
3	10	0.55	**0.7043**
4	10	0.05	0.1723
5	10	0.80	0.1781
8	20	0.10	**0.0771**
9	10	0.60	**0.1879**
11	30	0.05	**0.4450**
12	10	0.25	**0.6630**
14	20	0.05	**0.5044**
16	10	0.55	**0.1047**
17	10	0.90	0.6061
18	100	0.20	**0.8703**
20	100	0.05	**0.0869**
21	10	0.55	**0.4714**
22	20	0.05	**0.1897**
24	30	0.10	**0.0771**
25	30	0.20	**0.2462**
26	10	0.75	0.6098
27	100	0.05	**0.1811**
28	100	0.20	**0.0760**
30	100	0.45	**0.1634**
31	20	0.05	**0.2660**
32	10	0.90	**0.4173**
33	30	0.05	**0.3200**
35	100	0.05	**0.1758**
36	30	0.05	**0.1861**
37	100	0.05	**0.2683**
38	10	0.25	*0.3573*
39	10	0.05	**0.1815**
40	10	0.05	0.4268
41	10	0.20	**0.6008**
42	100	0.20	**0.1523**
43	10	0.95	*0.3906*
44	10	0.85	0.2688
45	10	0.60	0.7394
46	50	0.05	**0.4554**
47	10	0.25	**0.7935**
48	10	0.2	**0.5906**
49	10	0.15	**0.2488**
50	10	0.85	*0.9079*

Another detail of the results of MMR method using word embedding is shown in Table 5.2. Number of queries that have performance score above and under baseline scores are listed in that table. It shows that high percentage of queries have higher performance than baselines with help of the implicit diversification methods.

Table 5.2: Number of queries performed high and poor according to baseline scores with optimized parameter selection.

Run ID	Number of Queries	Number of Well Performed Queries	Number of Poor Performed Queries
2009HM	50	31	3
2010HM	48	36	5
2011HM	50	35	3
2012HM	50	34	11

The overall performances of each data sets with MMR algorithm is represented in Table 5.3. Samely, this performances are recorded using fixed, and also different, candidate set and trade-off parameter (λ) for each of the query. It is totally true that using best parameters with word embeddings representation provides higher performance than tf-idf. Because of this, in next section we will benefit from the results obtained by using word embeddings.

Table 5.3: Best overall performance metrics (α-nDCG@10) obtained by MMR algorithm with both document representations.

Run ID	Baseline	MMR with TF-IDF	MMR with Word Embeddings
2009HM	0.2520	0.2924	0.3044
2010HM	0.2427	0.2961	0.3137
2011HM	0.4680	0.5114	0.5194
2012HM	0.3218	0.3787	0.4452

5.2 Predicting the Candidate Set Size and Trade-off Parameter

It is stated that with the help of optimizing candidate set size and trade-off value per query can significantly improve the performance of implicit result diversification methods in previous chapter. We develop a learning model for predicting candidate set size and trade-off value for each query. In order to accomplish this task, we use classification methods of Weka framework [41] which is an open source machine learning tool. Weka basically contains various of machine learning algorithms for analyzing and modelling data.

Classification is a learning technique which categorizes instances into different labels or classes on the basis of several features. Although Weka framework provides lots of classifier, we have tried some of them such as Naive Bayes [42], Multilayer Perceptron [43], IBk (k-NN) [44] and J48 tree [45]. Since classifiers except for IBk have bad prediction accuracy, we have mainly focused on IBk for learning and prediction steps.

In this section, our purpose is to learn and predict optimal candidate set size and trade-off value for each query with a set of features. The features used throughout learning and shown in Table 5.4. Since each feature is calculated with varying candidate set sizes with starting 10 to 100 with an increase of step size 10, there exist approximately 10 different scores for each feature.

1. Ratio of Documents' Scores

   As proposed in [46], when differences in scores between first document and last document increase in result set, the probability of irrelevant documents that appears in result set also increases. For this reason, we have used ratio of the first to last document's scores in candidate document set with different size.

2. Mean of Documents' Scores

   This feature is simply calculated by averaging scores of documents in candidate document set with different size.

3. Decrease in Mean of Documents' Scores

Table 5.4: Feature list with properties.

Feature	Description	Count
scoreRatio	Ratio of top to last document's score	10
scoreMean	Mean of scores in document set	10
scoreMeanDecrease	Decrease in mean of scores in document set	9
scoreMedian	Median of scores in document set	10
standardDeviation	Standard deviation of scores	10
variance	Variance of scores	10
coefficientOfVariation	Coefficient of variation	10
minPairwiseTfIdfSimilarity	Minimum pairwise (td-idf vector) similarity	10
maxPairwiseTfIdfSimilarity	Maximum pairwise (td-idf vector) similarity	10
avgPairwiseTfIdfSimilarity	Average pairwise (td-idf vector) similarity	10
minPairwiseWESimilarity	Minimum pairwise (WE vector) similarity	10
maxPairwiseWESimilarity	Maximum pairwise (WE vector) similarity	10
avgPairwiseWESimilarity	Average pairwise (WE vector) similarity	10
minPairwiseEntitySimilarity	Minimum pairwise (Entity list) similarity	10
maxPairwiseEntitySimilarity	Maximum pairwise (Entity list) similarity	10
avgPairwiseEntitySimilarity	Average pairwise (Entity list) similarity	10
NQC	Scores of Normalized Query Commitment	10

In addition to the previous feature, decrease between those such as difference between mean of top 10 document's scores and mean of top 20 document's scores are included to our feature set.

4. Median of Scores

   The score of document which appears in the middle of candidate document set is used as feature.

5. Variance

   Variance is the mean squares of differences between scores of documents in candidate set and mean of them.

6. Standard Deviation

   Standard deviation is a measurement that represents how disperse data [47]. It is calculated by extracting square root of the variance.

7. Coefficient of Variation

   It is simply calculated by dividing standard deviation by mean score of documents in candidate set.

8. Minimum/Maximum/Average Pairwise Similarity of TF-IDF Vectors of Documents

   These three features indicates minimum,maximum and average pairwise similarity between documents which are represented by TF-IDF vectors. It is calculated by cosine similarity function as described in Eq. 2.10 among all documents in candidate set with different sizes.

9. Minimum/Maximum/Average Pairwise Similarity of Word Embeddings of Documents

   These three features indicates minimum,maximum and average pairwise similarity between documents which are represented by Word Embeddings. It is calculated by cosine similarity function as described in Eq. 2.10 among all documents in candidate set with different sizes.

10. Minimum/Maximum/Average Pairwise Similarity of Entities of Documents

These three features indicates minimum,maximum and average pairwise similarity between documents which are represented by entity vectors. Entity vectors belonging to documents are constructed by DBPedia[1] and the similarity measurement is calculated by jaccard similarity function as described in Eq. 2.12.

11. Normalized Query Commitment Scores

    As proposed in [8], NQC analyzes query performance by checking difference in scores of documents in candidate result set and it is calculated by:

$$NQC(q_i) = \frac{1}{Sim(C, q_i)} \times \sqrt{\frac{1}{N} \times \sum_{d \in D_{q_i}^{[N]}} \left(Sim(d, q_i) - mean_{d \in D_{q_i}^{[N]}}(Sim(d, q_i))\right)^2} \tag{5.1}$$

    where Sim is the relevance score of given a document$d$ for a query q_i. C represents the corpus as adocument by concatenating all documents inside it and Nis the number of documents in candidate set.

12. Optimum candidate set size (class label)

    For each query, best performing candidate set size is used as a class label. This feature only appears in data set which used to predict candidate set size.

13. Optimum trade-off λ value (class label)

    For each query, best performing trade-off λ value is used as a class label. This feature only appears in data set which used to predict trade-off λ value.

Two data set are built from these all features in order to predict candidate set size and trade-off values. Since IBk performs better than other classification methods for 2009HM data set, we have continued to use IBk to other data sets.

At first step, candidate set sizes are trained and predicted through 5-fold cross validation. Likewise, this process is done for trade-off values. After obtained both parameters for each query, MMR algorithm is applied to each query to get diversified results.

[1] http://dbpedia.org

Our results which are obtained by prediction are listed in Table 5.5. Results proves that performance of diversification obtained with predicted candidate set size and trade-off value for each query outperforms the results obtained with fixed candidate set size (100) and varying trade-off λ values for all queries stated in Chapter 4.

Table 5.5: MMR diversification results with predicted candidate set size and trade-off values by IBk algorithm. All α-nDCG@10 scores shows average scores of queries.

Run ID	Baseline α-nDCG@10	MMR with Prediction α-nDCG@10
2009HM	0.2520	**<u>0.2612</u>**
2010HM	0.2427	**0.2554**
2011HM	0.4680	**<u>0.4750</u>**
2012HM	0.3218	**0.3392**
2009TREC	0.2530	**0.2589**
2010TREC	0.3716	**<u>0.3768</u>**
2011TREC	0.5312	**0.5379**
2012TREC	**0.4942**	0.4890

For 2009HM run, in previous chapter best α-nDCG@10 score achieved by MMR algorithm with documents are represented as WE vector. MMR algorithm produces score of 0.2593 with fixed candidate set size ($= 100$) and optimal trade-off λ value ($= 0.65$). On the other hand, with adaptive usage of both parameters for 2009HM provides more diversity with score of 0.2612. According to baseline metrics, overall diversity improved by 3.6%.

In 2010HM run 0.2554 α-nDCG@10 score is achieved by predicting candidate set size and trade-off parameter. Approximately 5.2% improvement is gained in this data set.

The best diversification score (α-nDCG@10 = 0.4697) of run 2011HM is achieved by using MMR algorithm with WE. With the help of prediction, score of 2011HM rises up to 0.4750 where improvement achieved by 1.5%. Since the baseline score of 2011HM is much higher than the other runs, improvement achieved by prediction is not perfect as the others. It shows us that diversification performance with strong baselines (in our cases 2011HM data with baseline 0.4680) is very limited to improvement even with optimal candidate set size and trade-off value.

For 2012HM run, SY algorithm have performed better than MMR by selecting candidate set size as 100 and threshold β value as 0.90 in previous chapter. By using predicted parameters, the best improvement in all runs is achieved for 2012HM run. Diversification performance reaches to 0.3392 and 5.4% improvement is gained.

For more competitive TREC runs, the relative gains are in the range 1.2% to 2.3% (except the 2012 run, where there is a relative degradation of 1%). Given that the latter runs are employing sophisticated approaches far beyond HM runs, our findings are promising. Note that, in some cases (underlined in Table 5.5) improvements with respect to baselines are statistically significant (using paired t-test at 0.05 confidence level).

If we compare results achieved by learning both parameters in Table 5.5 to the results in previous chapter, it is totally true that learning the parameters provides better results. On the other hand, maximum scores achieved by optimal values (per query) shown in Table 5.3 is far beyond the our performance which shows us that some more improvement is still possible.

CHAPTER 6

CACHING DOCUMENT SIMILARITIES FOR IMPLICIT RESULT DIVERSIFICATION

In this chapter, we will briefly explain how our cache is develop, what is stored inside the cache and how we used the cache in implicit search result diversification.

As explained in Chapter 2, most of the implicit search result diversification methods have two main functions. The first one calculates similarity scores of a document and a query, the second one computes dissimilarity or similarity between candidate document and already selected document for a result set. At each iteration these calculations are done until finding top-k documents for result set. If we think the efficiency of a implicit diversification employed on a large scaled web search engine under this circumstances, it can be said that calculations especially finding similarity scores between documents can not be efficient. Therefore, a simple cache structure adapted in diversification in order to boost efficiency of these calculations.

With the help of AOL Query Logs, we have computed pairwise similarity between highly ranked documents from each query result and stored in the cache. In order to analyze cache performance, MMR algorithm using word embeddings as a diversification method, is employed on our home made runs namely 2009HM, 2010HM, 2011HM and 2012HM. Approximately 200k pairwise document similarity is calculated during diversification process. As a result of this, how cache hits changes with varying numbers of documents cached is represented in Figure 6.1. The highest number of cache hits is achieved in 2012HM data with 56621 times when pairwise similarity is calculated among top 100 documents from 100k query results. And, the lowest one observed in 2011HM data with 26079 times.

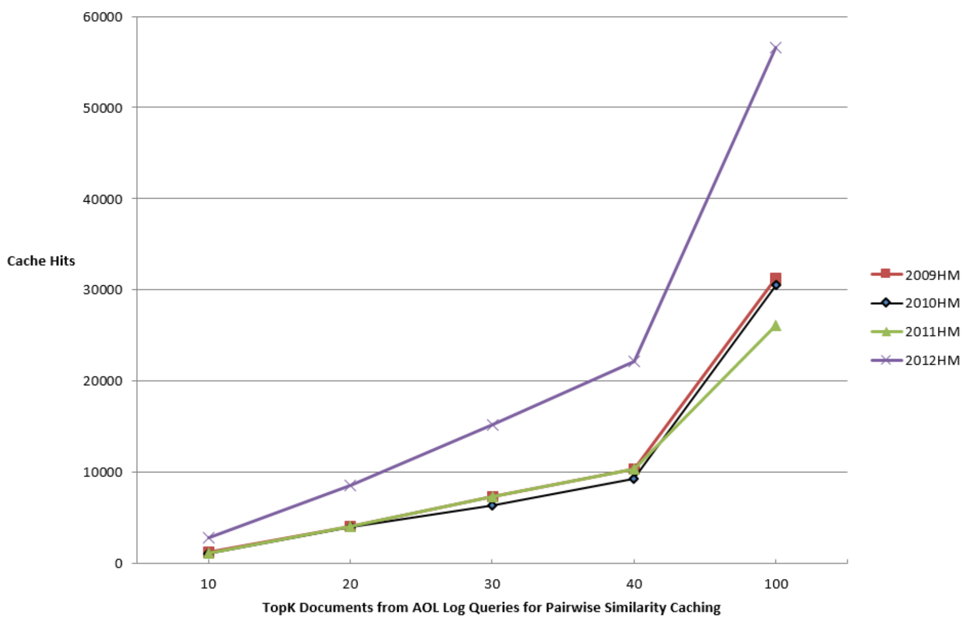

Figure 6.1: Visualization of cache hits.

By nature of caching, increasing number of high ranked documents for caching results in growing size of the cache. The overall number of documents to compute pairwise similarity and cache sizes are shown in Table 6.1. It is assumed that two integer values for document identifiers and a single double value for similarity score are stored in memory.

Table 6.1: Different cache sizes obtained from using varying number of documents.

Number of Documents	Number of Pairs	Cache Size
10	4253726	60 Megabyte
20	17709013	250 Megabyte
30	40128427	600 Megabyte
40	71365775	1 Gigabyte
100	433684174	6 Gigabyte

In addition to examining cache hits, efficiency in time and effectiveness in diversification performance are investigated through caching experiments with using pairs of top 100 documents from each query results. We have implemented two different caching strategy in MMR. In first one, namely Case-1, the similarity score of pairwise document is fetched from the cache, if it contains. And, if the cache does not have the score, it is calculated at run time. On the other hand, in second case, namely Case-2, if cache does not have the similarity score of pair of the documents, average similarity score is used instead of calculating.

Table 6.2: Efficiency and effectiveness of original MMR.

Run ID	**Original MMR (aNDCG@10)**	**Original MMR Time (ms)**	**No. of Similarity Calculations**
2009HM	0.253145	3.5	4215
2010HM	0.257372	3.3	4215
2011HM	0.469732	3.5	4215
2012HM	0.321568	3.5	4215

Table 6.3: Efficiency and effectiveness of Case-1 Caching in MMR.

Run ID	**MMR & Cache Case-1 (aNDCG@10)**	**MMR & Cache Case-1 Time (ms)**	**No. of Similarity Calculations**
2009HM	0.253145	2.8	3591
2010HM	0.257372	2.8	3619
2011HM	0.469732	2.9	3706
2012HM	0.321568	2.6	3086

Firstly original MMR performance, corresponding MMR execution time in millisecond and number of pairwise similarity calculation are shown in Table 6.2 in order to show our baseline.

In Tables 6.3 and Table 6.4, two different usage of caching performances are shown. According to experimental results, caching in MMR decreases the amount of time, especially in Case-2. Also, it reveals that using average similarity score for unseen pair of documents decreases effectiveness of MMR algorithm slightly.

Table 6.4: Efficiency and effectiveness of Case-2 Caching in MMR.

Run ID	MMR & Cache Case-2 (aNDCG@10)	MMR & Cache Case-2 Time (ms)
2009HM	0.250793	0.6
2010HM	0.245713	0.5
2011HM	0.467226	0.6
2012HM	0.318221	0.6

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this thesis, we focused on the implicit search result diversification methods and proposed approaches to improve their effectiveness and efficiency. We first demonstrated that traditional implicit result diversification methods with predefined parameters rarely outperform the initial retrieval results that are obtained by a typical ranking function, such as Okapi BM25. Even when these methods provide better diversification performance than the baselines, the performance gains are found to be only marginal. We further showed that implicit diversification benefits from the proposed idea of employing word embeddings based document representations, but the performance improvements are still limited.

As a remedy, we proposed to predict the candidate set size, a key parameter for implicit diversification, using a rich set of features on a per query basis. By predicting the candidate set size (together with λ, as in [7]) and employing word embeddings, we achieved higher diversification performance.

Finally, we proposed caching pairwise document similarities to improve the implicit diversification efficiency. Our experiments using a representative diversification method, MMR [4], revealed that the proposed caching idea can provide significant gains in processing efficiency.

In our future work, we plan to use document embeddings (e.g., Doc2Vec [48]) to represent document in the context of result diversification. As a second research direction, we will exploit additional (e.g., click-based) features for better prediction of the diversification parameters. Finally, we aim to employ and evaluate dynamic caching strategies to further improve the efficiency of implicit diversification techniques.

REFERENCES

[1] L. Xia, J. Xu, Y. Lan, J. Guo, and X. Cheng, “Learning maximal marginal relevance model via directly optimizing diversity evaluation measures,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Santiago, Chile), pp. 113–122, 09-13 August 2015.

[2] R. L. T. Santos, C. Macdonald, and I. Ounis, “Search result diversification,” *Foundations and Trends in Information Retrieval*, pp. 1–90, 2015.

[3] J. He, E. Meij, , and M. de Rijke, “Result diversification based on query-specific cluster ranking,” *Journal of the American Society for Information Science*, pp. 550–571, 2011.

[4] J. G. Carbonell and J. Goldstein, “The use of mmr, diversity-based reranking for reordering documents and producing summaries,” in *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336, 1998.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 3111–3119, 2013.

[6] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *ICLR Workshop*, 2013.

[7] R. L. T. Santos, C. MacDonald, and I. Ounis, “Selectively diversifying web search results,” in *CIKM*, 2010.

[8] A. Shtok, O. Kurland, and D. Carmel, “Predicting query performance by query-drift estimation,” in *Proceedings of the 2Nd International Conference on Theory*

of Information Retrieval: Advances in Information Retrieval Theory, pp. 305–312, 2009.

[9] D. Carmel and O. Kurland, "Query performance prediction for ir," in *Proc. of SIGIR*, pp. 1196–1197, 2012.

[10] G. Markovits, A. Shtok, O. Kurland, and D. Carmel, "Predicting query performance for fusion-based retrieval," in *Proc. of CIKM*, pp. 813–822, 2012.

[11] B. R. Boyce, "Beyond topicality : A two stage view of relevance and the retrieval process," *Information Processing & Management*, vol. 18, no. 3, pp. 105–109, 1982.

[12] W. Goffman, "A searching procedure for information retrieval," *Information Storage and Retrieval*, vol. 2, no. 2, pp. 73–78, 1964.

[13] S. Robertson, "The probability ranking principle in ir," in *Readings in Information Retrieval*, pp. 335–336, 1997.

[14] R. L. T. Santos, C. Macdonald, and I. Ounis, "On the role of novelty for search result diversification," *Information retrieval*, vol. 15, no. 5, pp. 478–502, 2012.

[15] S. Sanner, S. Guo, T. Graepel, S. Kharazmi, and S. Karimi, "Diverse retrieval via greedy optimization of expected 1-call@k in a latent subtopic relevance model," in *CIKM*, 2011.

[16] J. Wang and J. Zhu, "Portfolio theory of information retrieval," in *Proc. 32nd Int. ACM SIGIR Conf. on Research and Development in IR*, pp. 115–122, ACM, 2009.

[17] C. Zhai and J. D. Lafferty, "A risk minimization framework for information retrieval," *Information Processing & Management*, vol. 42, pp. 31–55, 2006.

[18] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong, "Diversifying search results," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pp. 5–14, 2009.

[19] S. Gollapudi and A. Sharma, "An axiomatic approach for result diversification," in *Proc. WWW*, 2009.

[20] R. Hassin, S. Rubinstein, and A. Tamir, “Approximation algorithms for maximum dispersion,” *Operations Research Letters*, vol. 21, pp. 133–137, 1997.

[21] K. Tao, F. Abel, C. Hauff, G.-J. Houben, and U. Gadiraju, “Groundhog day: near-duplicate detection on twitter,” in *Proc. WWW*, 2013.

[22] R. L. T. Santos, J. Peng, C. Macdonald, and I. Ounis, “Explicit search result diversification through sub-queries,” in *Proceedings of ECIR*, pp. 87–99, 2010.

[23] R. Santos, C. Macdonald, and I. Ounis, “Exploiting query reformulations for web search result diversification,” in *Proceedings of the 19th International Conference on World Wide Web, WWW ’10*, pp. 881–890, 2010.

[24] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[25] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *The Journal of Machine Learning Research*, pp. 1137–1155, 2003.

[26] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proc. of EMNLP*, 2014.

[27] C. D. Boom, S. V. Canneyt, T. Demeester, and B. Dhoedt, “Representation learning for very short texts using weighted word embedding aggregation,” *Pattern Recognition Letters*, vol. 80, pp. 150–156, 2016.

[28] M. Akcay, “Analyzing and boosting the performance of explicit result diversification methods for web search,” Master’s thesis, METU, December 2016.

[29] W. Aha, D. Kibler, and M. K. Albert, “Instance-based learning algorithms,” *Machine Learning*, vol. 6, pp. 37–66, 01 1991.

[30] S. Robertson and H. Zaragoza, “The probabilistic relevance framework: Bm25 and beyond,” *Foundations and Trends in Information Retrieval*, vol. 3, pp. 333–389, Apr. 2009.

[31] K. S. Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[32] W. Effelsberg and T. Haerder, "Principles of database buffer management," *ACM Trans. Database Syst.*, vol. 9, pp. 560–595, Dec. 1984.

[33] E. Markatos, "On caching search engine query results," *Comput. Commun.*, vol. 24, pp. 137–143, Feb. 2001.

[34] R. Baeza-Yates and F. Saint-Jean, "A three level search engine index based in query log distribution," in *String Processing and Information Retrieval*, pp. 56–65, 2003.

[35] T. Fagni, R. Perego, F. Silvestri, and S. Orlando, "Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data," *ACM Trans. Inf. Syst.*, vol. 24, pp. 51–78, Jan. 2006.

[36] R. Ozcan, I. Altingovde, and O. Ulusoy, "Static query result caching revisited," pp. 1169–1170, 01 2008.

[37] C. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, "Novelty and diversity in information retrieval evaluation," pp. 659–666, 01 2008.

[38] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, pp. 422–446, 2002.

[39] A. Ozdemiray and I. Altingövde, "Explicit search result diversification using score and rank aggregation methods," *Journal of the Association for Information Science and Technology*, vol. 66, no. 6, pp. 1212–1228, 2015.

[40] S. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Inf. Theor.*, vol. 28, no. 2, pp. 129–137.

[41] I. Witten, I. H, F. , and E. , *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. 10 1999.

[42] P. Langley and S. Sage, "Induction of selective bayesian classifiers," in *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, pp. 399–406, 1994.

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition," pp. 318–362, 1986.

[44] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, pp. 175–185, 1992.

[45] S. Wang, M. Zhou, and G. Geng, “Application of fuzzy cluster analysis for medical image data mining,” *Journal of Computational Physics - J COMPUT PHYS*, pp. 631 – 636, 2005.

[46] A. M. Ozdemiray and I. S. Altingovde, “Query performance prediction for aspect weighting in search result diversification,” in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1871–1874, ACM, 2014.

[47] J. Martin Bland and D. Altman, “Statistics notes: Measurement error,” *BMJ*, vol. 313, p. 744, 1996.

[48] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *Proc. of ICML*, pp. 1188–1196, 2014.