
**DEPARTMENT OF INFORMATICS**
**PROGRAM OF POSTGRADUATE STUDIES IN COMPUTER SCIENCE**

**M.Sc. Thesis**

# Document Re-ranking in a Biomedical Question Answering System

**Dimitris Gkoumas**
**EY1403**

**Supervisor: Ion Androutsopoulos**
**Assistant Supervisor: Prodromos Malakasiotis**

**ATHENS, FEBRUARY 2016**

## Acknowledgements

I would like to express my sincere gratitude to my supervisor Ion Androutsopoulos and my co-supervisor Makis Malakasiotis for their continuing help and guidance throughout this thesis. They made it possible for me to carry out this thesis and they paved the way with their acute remarks and clarifications. Finally I would like to thank my family for their support.

# Contents

# Chapter 1

# Introduction

## 1.1 PubMed/MEDLINE Search Engine

PubMed/MEDLINE is a biomedical search engine developed by the National Institutes of Health (NIH) of the National Library of Medicine (NLM). Figure 1.1 shows that PubMed/MEDLINE has seen a rapid growth in the number of articles it indexes.

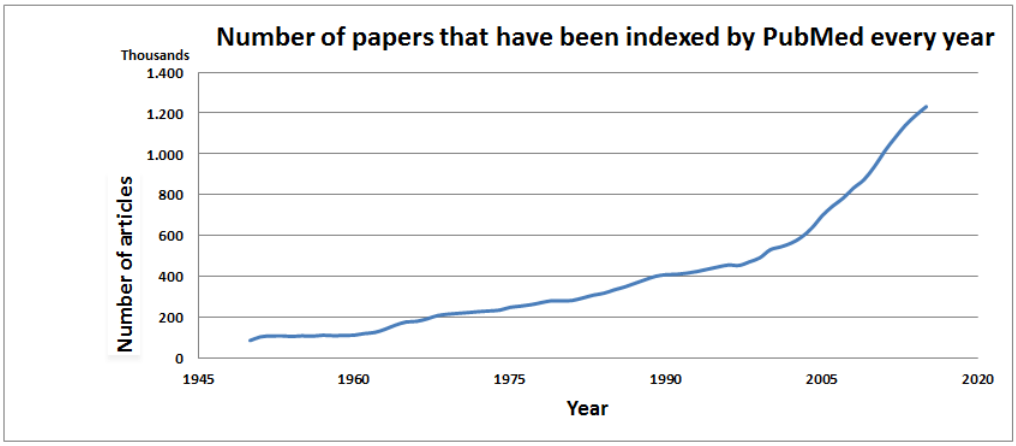


Figure 1.1: How many papers have been indexed by PubMed every year.

Despite the effort that has been put into the development of PubMed/MEDLINE and similar systems (e.g., Google Scholar) the needs of the biomedical researchers have not exactly been met yet [3]. This is because a simple search engine returns a list of possibly relevant articles and it is difficult and time consuming for the researchers to study all of them to

figure out which ones are really relevant and to find the information they need (e.g. in particular sentences or paragraphs). The ideal solution would be a better information retrieval process along with an answer (small summary) based on the relevant evidence. Recently many efforts have been made towards the direction of building such improved biomedical question answering (QA) systems.

## 1.2 Biomedical QA Challenges

The Text REtrieval Conference (TREC) has run two challenges [7] [8] in which the participants had to develop biomedical QA systems. In detail, the participants were given a corpus of 162,259 full-text biomedical documents from PubMed/MEDLINE and biomedical questions. Each system had to return short passages from the texts that answered the corresponding questions, along with the locations of the passages in the documents. The performance of the systems was scored using mean average precision (MAP) at the passage, aspect and document level. In another biomedical QA challenge which was run by BioASQ [20], the systems had to semantically index biomedical articles and return user-understandable answers to corresponding questions. More specifically, in BioASQ the participants could compete in two tasks:
**Task A** (Large-scale online biomedical semantic indexing): systems had to label with MeSH headings new PubMed/MEDLINE abstracts.
**Task B** (Biomedical semantic QA):

1. In Phase A, systems had to:
   (a) annotate input natural language questions with biomedical concepts,
   (b) return relevant articles in English from PubMed/MEDLINE,
   (c) return relevant snippets from the relevant articles, and
   (d) return relevant RDF triples from designated ontologies.
2. In Phase B, the systems had to find and return exact answers (e.g. names of genes, symptoms) and summaries for each question based on the gold (correct) articles, snippets, concepts and triples of Phase A.

## 1.3 A Typical QA System Architecture

A typical Question Answering system architecture is depicted in Figure 1.2. In the picture we have color coded relevant documents as green and irrelevant documents as red.

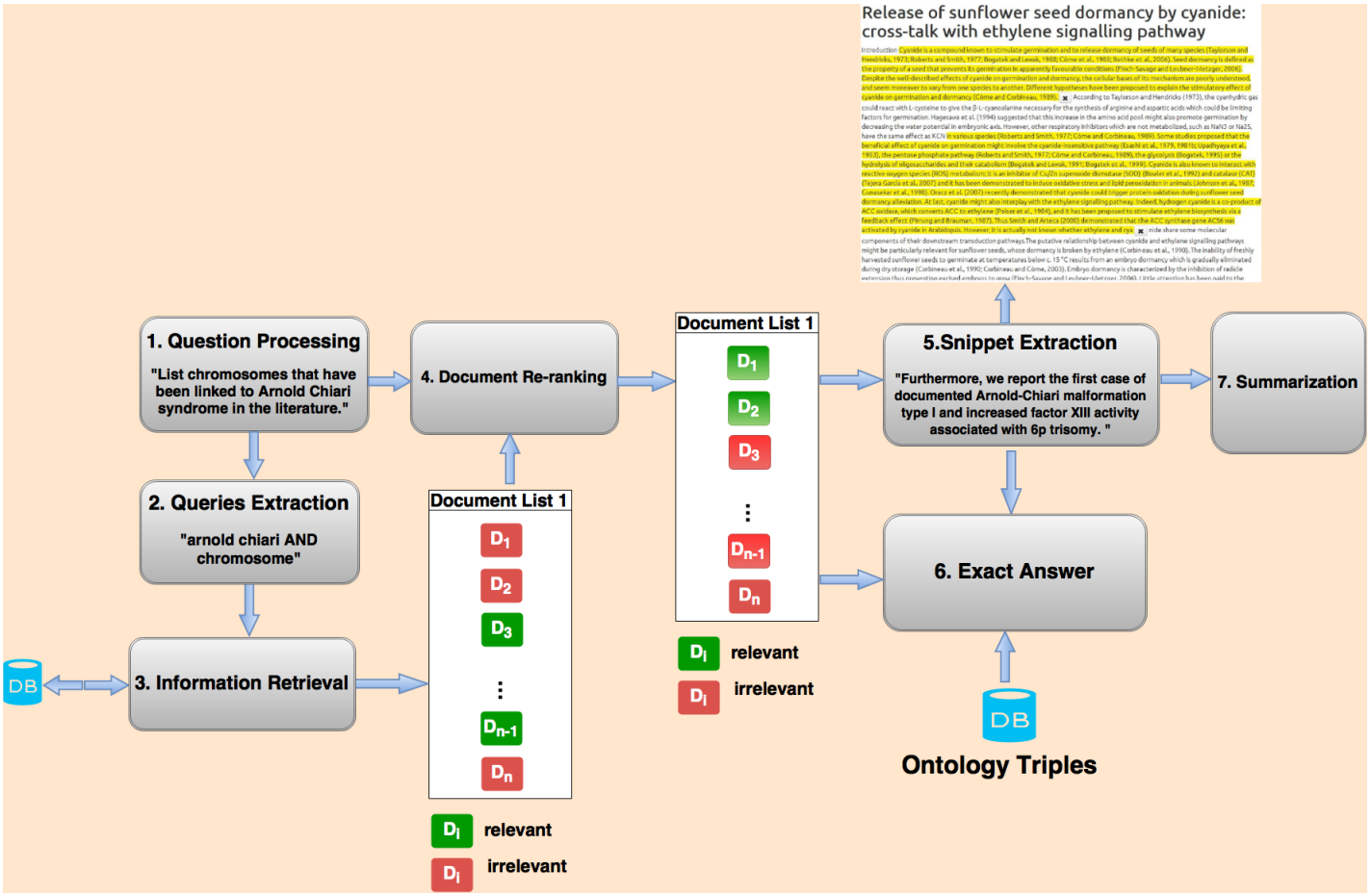


Figure 1.2: A typical architecture of a biomedical QA system.

## 1.4 Goal of the Thesis

This thesis aims to study the problem of deciding if a document is relevant to a specific question. We are going to develop a re-ranking system which will complement an already existing QA biomedical system like the one that is depicted in Figure 1.2. Our system (module number 4) will take as

input a question and a list of biomedical articles retrieved from a biomedical search engine (PubMed/MEDLINE). It will re-rank the articles promoting to the top of the list the ones it considers relevant to the question.

A typical Information Retrieval (IR) system takes as input a query and outputs a list of possibly relevant documents, assigning a relevance score to each document. The higher the score, the higher the document appears in the list. However, there are many occasions where the IR system mistakenly returns irrelevant documents high in the list or relevant documents low in the list, make it unlikely for a user to read them. Taking an example from BioASQ [20], assume we have the question "Are genes symmetrically distributed between leading and lagging DNA strand in bacteria?". Using the query 'genes' AND 'leading' AND 'lagging', PubMed returns 187 articles (Figure 1.3). The BioASQ dataset indicates that only two are relevant to the question, those with pmid 24273314 and 23538833. The first one is ranked $11^{th}$ and the second one $13^{th}$. This means that the first 10 articles are irrelevant and do not answer the question. However, the users have to read them all until they finally find a relevant document at the $11^{th}$ place. Our system aims to re-rank the retrieved list of documents and move those two relevant documents to the top of the list. Thus, we aim for a high precision score at the top of the re-ranked list.

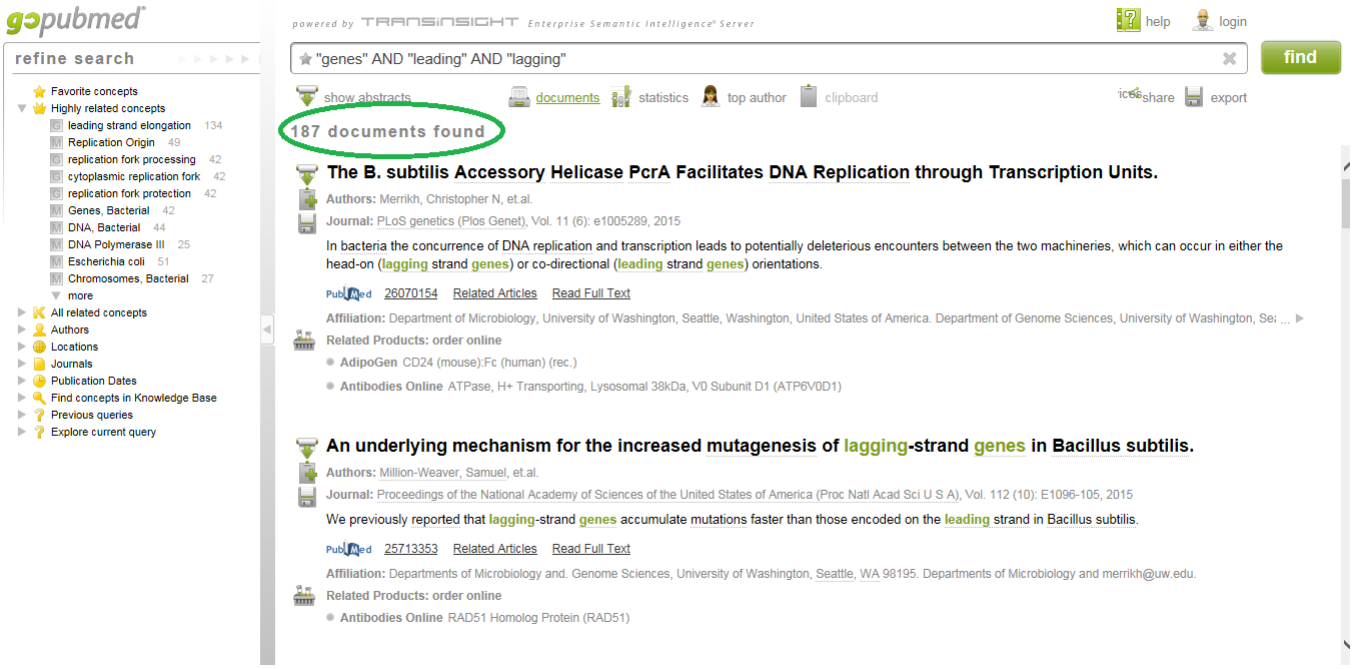


Figure 1.3: List of documents returned by PubMed.

## 1.5 Outline of the Rest of this Thesis

The rest of this thesis is organized as follows:

- Chapter 2 provides a short description of the external tools that we used in order to build our re-ranking system.
- Chapter 3 presents our re-ranking system architecture and the features that we used in its machine learning components.
- Chapter 4 presents the data that we used in order to train and evaluate our system.
- Chapter 5 presents the experimental results on the development and test data.
- Chapter 6 discusses related work.
- Chapter 7 concludes and proposes future directions.

# Chapter 2

# External Tools

We used a variety of external tools during the development of our system. More specifically we used:

1. Unified Medical Language System UMLS - MetaMap [1].
2. Word2Vec [13] [12] [14].
3. Earth Movers Distance [10] Java implementation.[1]
4. LibLinear logistic regression implementation[2] version 1.95 [4], learning to rank SVM (Support Vector Machine) [2] implementation.[3]
5. Stanford Tokenizer[4] version 3.5.0.

The most important ones are described in the following sub-sections.

## 2.1 Unified Medical Language System UMLS - MetaMap

The Unified Medical Language System (UMLS) is a biomedical ontology and thesaurus. For the purposes of this thesis we used MetaMap[5] to exploit UMLS as a Named Entity Recognizer in order to find **biomedical concepts, synonyms** and **semantic** types inside texts. MetaMap takes as

[1] https://github.com/telmomenezes/JFastEMD
[2] https://www.csie.ntu.edu.tw/∼cjlin/liblinear
[3] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[4] http://nlp.stanford.edu/software/tokenizer.shtml
[5] https://metamap.nlm.nih.gov/

input a biomedical text and discovers concepts inside it. Figure 2.1[6] depicts an example output of MetaMap when we give it as input the phrase "heart attack".

```
Phrase: "heart attack"
Meta Candidates (8):
  1000 C0027051:Heart attack (Myocardial Infarction) [Disease or
   Syndrome]
   861 C0018787:Heart [Body Part, Organ, or Organ Component]
   861 C0277793:Attack, NOS (Onset of illness) [Finding]
   861 C0699795:Attack (Attack device) [Medical Device]
   861 C1261512:attack (Attack behavior) [Social Behavior]
   861 C1281570:Heart (Entire heart) [Body Part, Organ, or Organ
   Component]
   861 C1304680:Attack (Observation of attack) [Finding]
   827 C0004063:Attacked (Assault) [Injury or Poisoning]
Meta Mapping (1000):
  1000 C0027051:Heart attack (Myocardial Infarction) [Disease or
   Syndrome]
```

Figure 2.1: Example of MetaMap output for the phrase "heart attack".

In Figure 2.1, MetaMap returned 8 possible candidates and assigned a score to each one of them, which denotes how relevant the candidate is to the input phrase. Then from these candidates the program produces the mappings. In our case just one Meta Mapping with score 1000 is returned. Figure 2.2[7] describes in detail the meaning of each output part.

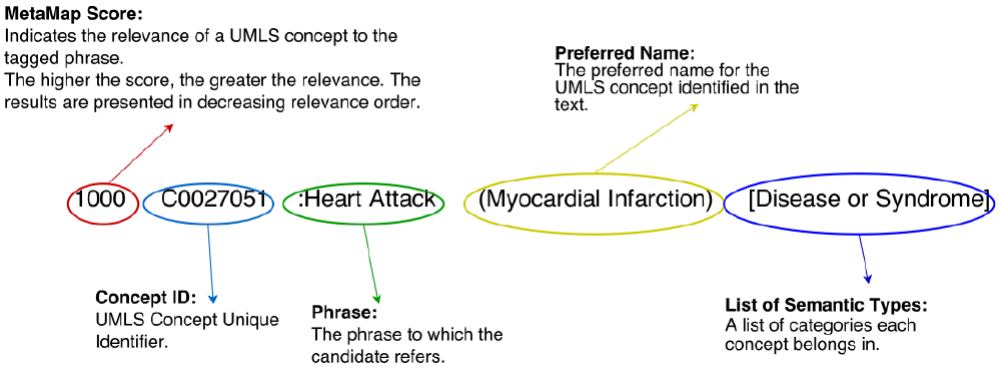


Figure 2.2: Explanation of MetaMap output.

[6] http://ii.nlm.nih.gov/Publications/index.shtml
[7] Picture taken from [5].

The MetaMap output parts that we use to tag the biomedical documents and questions are:

1. **Concept Id:** The unique Id of the concept.
2. **Preferred name:** The name of the concept (which may be the same with the matched string).
3. **Matched Words:** A list representing (1) the correspondence between the words in the preferred name of the candidate concept words of the input text and (2) any lexical variation in the matching.
4. **Semantic types:** A list of categories the identified concept belongs in (e.g., the semantic type of "heart attack" is "Disease or Syndrome").

## 2.2 Embeddings - Word2Vec

Word2Vec [13] [12] [14] maps words to dense vectors of real numbers, called "embeddings". The essence is that syntactically or semantically similar words are mapped to nearby vectors. In our case Word2Vec was applied to a large biomedical corpus consisting of 10,876,004 abstracts.[8] We use the skip-gram model of Word2Vec with negative sampling, 300 dimensions and default parameters. The resulting vectors (embeddings) were used as features of our re-ranking filter.

## 2.3 Learning to Rank SVM

In order to build our re-ranking system, among other approaches, we also used a learning to rank SVM [9]. Unlike ordinary SVMs, a learning to rank SVM (also called Ranking SVM) assumes that the instances of each class are ordered by preference (e.g., relevance of degree). During training, it forms pairs of instances and learns to predict the correct order (ranking) of the elements of each pair, which leads to more training examples (pairs taken than individual instances).

## 2.4 Word Mover's Distance

In our system, during feature extraction, we also use the Word Mover's Distance (WMD) [10]. WMD is a special case of the Earth Mover's Distance

[8] http://bioasq.lip6.fr/tools/BioASQword2vec/

(EMD) [16] [17] and it measures the dissimilarity between two documents by exploiting the token embeddings of each document. In essence, it sums the distances the embeddings of the words of one document need to travel in order to be transformed into the word embeddings of the other document. For the purposes of this thesis, we measure the dissimilarity between a document abstract and a question. We compute the distance between two word embeddings as their Euclidean distance.

# Chapter 3

# The Re-ranking System of this Thesis

## 3.1 System Architecture

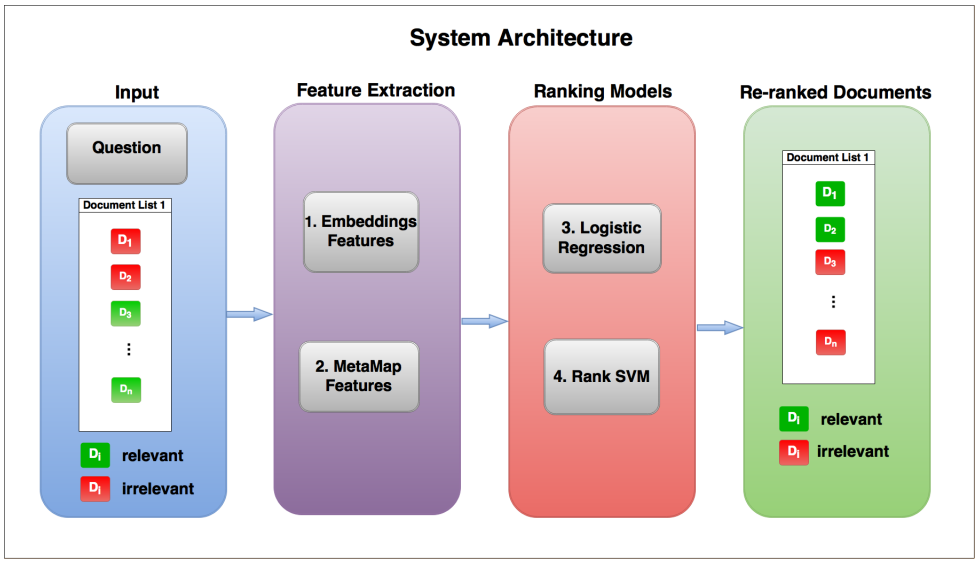


Figure 3.1: Re-ranking System Architecture.

The architecture of our system is depicted in Figure 3.1. At the first stage, our system takes as input a biomedical question and a list of biomedical documents that a search engine like PubMed returns. After that, during feature extraction, the document abstracts and the question are trans-

formed into feature vectors.To achieve that we use MetaMap to find biomedical evidence inside the abstracts and the question and word embeddings as derived from Word2Vec. During this stage we also use Stanford's tokenizer to split the abstracts and the question into tokens. At the next stage a trained classifier like logistic regression or rank SVM is used to predict the degree to which a document (abstract) is relevant to the question and a re-ordered list with documents is constructed. We should mention that in our datasets there are no preferences between relevant documents. Thus, we treated the problem as binary classification, where a document is either relevant to the question or irrelevant. When using logistic regression, the confidence of the classifier that the document is relevant is used as the document's score. In the learning to rank SVM, we require (during training) all the relevant documents to be ranked above all the the irrelevant ones per question.

## 3.2 MetaMap Features

For the first set of features, we use MetaMap and compute the features exactly as in [5]. We consider each document's abstract as a long sentence and represent it as $D$. Also we represent the question as $Q$. The reader is reminded that given two sets $S_1$ and $S_2$ their Dice coefficient is:

$$\frac{2 \cdot |S_1 \cap S_2|}{|S_1| + |S_2|} \tag{3.1}$$

The features of this set are:

1. Dice coefficient between the sets of distinct MetaMap concept IDs of $Q$ and $D$, respesctively.
2. Dice coefficient between the sets of distinct MetaMap preferred names of $Q$ and $D$, respesctively.
3. Dice coefficient between the sets of distinct MetaMap matched words of $Q$ and $D$, respesctively.
4. Dice coefficient between the sets of distinct MetaMap semantic types of $Q$ and $D$, respesctively.
5. Number of distinct semantic types shared by $Q$ and $D$, divided by the number of the distinct semantics types of $Q$; intuitively, this is the recall of $Q$'s semantic types in $D$.

6. Number of distinct concept IDs shared by $Q$ and $D$, divided by the number of distinct concept IDs of $Q$; intuitively, this is the recall of $Q$'s concept IDs in $D$.

## 3.3 Features from Word Embeddings

For the second set of features, we use the Word2Vec embeddings. We compute these features using the formulas of [5], as follows:

1. **Euclidean distance based similarity using centroids.** This is computed as:

$$Sim(Q, D) = \frac{1}{1 + d(c(D), c(Q))} \tag{3.2}$$

where $c(Q)$, $c(D)$ are the centroid vectors of the embeddings of $Q$ and $D$ respectively, defined below, and $d(a, b)$ is the Euclidean distance between two vectors $a$,$b$ which is computed as follows, where m is the dimensionality of the vectors:

$$d(a, b) = \sqrt{\sum_{i=1}^{m} (a_i - b_i)^2} \tag{3.3}$$

In order to compute the centroid of a document $D$ (or a question $Q$), $D$ (or $Q$) is first tokenized and for each token $t$ the corresponding embeddings $v(t)$ is retrieved. The centroid of $D$ (similarly, $Q$) is then computed as follows, where $n$ is the number of tokens of $D$ (or $Q$). In effect, the centroid also takes into account the term frequencies of the tokens.

$$c(D) = \frac{1}{n} \cdot \sum_{i=1}^{n} v(token_i) \tag{3.4}$$

2. **Euclidean distance based similarity using idf-weighted centroids.** This is the same score as in Equation 2, with the difference that the vector of each token is weighted by the corresponding inverse token frequence ($idf$) score. Hence, the centroid of $D$ (similarly, $Q$) is now computed as follows:

$$c(D) = \frac{\sum_{i=1}^{n} v(token_i) \cdot idf_i}{\sum_{i=1}^{n} idf_i} \tag{3.5}$$

where the $idf$ scores have been computed on 10.876.004 abstracts of PubMed as follows:

$$idf(token_i) = \log(\frac{10.876.004}{df(token_i)}) \tag{3.6}$$

and $df(token_i)$ is the document frequency of $token_i$ in the collection, i.e., the number of documents it occurs in.

3. **Euclidean distance based similarity between the document and question tokens.** To compute this set of features, we first create a set containing the tokens of $D$ and a set containing the tokens of $Q$. We then compute the similarities between all the possible pairs of tokens ($t_Q$, $t_D$) of the two sets as follows:

$$Sim(t_Q, t_D) = \frac{1}{1 + d(v(t_Q), v(t_D))} \cdot \frac{idf_{t_Q} \cdot idf_{t_D}}{maxIdf^2} \tag{3.7}$$

where $v(t_Q)$ and $v(t_D)$ are the embeddings of $t_Q$, $t_D$, respectively, and $maxIdf$ is the maximum idf of all the distinct tokens of the training data that we used to train the logistic regression or ranking SVM.
We then compute the following features:

- Maximum of the similarity scores $Sim(t_Q, t_D)$, over all of the pairs of tokens $t_Q$, $t_D$ of the question and document, respectively.
- Minimum of all the similarity scores $Sim(t_Q, t_D)$.
- Median of all the similarity scores $Sim(t_Q, t_D)$.
- Average of the three minimum similarity scores $Sim(t_Q, t_D)$.
- Average of the three maximum similarity scores $Sim(t_Q, t_D)$.
- Average of all the similarity scores $Sim(t_Q, t_D)$.

4. The same features as in 3, but without weighting with *idf* scores. Hence, the similarity is now computed as follows:

$$Sim(t_Q, t_D) = \frac{1}{1 + d(v(t_Q), v(t_D))} \tag{3.8}$$

## 3.4 Word Mover Distance Features

The Word Mover Distance (WMD) features are also computed on the word embeddings of the questions and documents (abstracts), but they presented and studied seperately. They are the following:

1. **WMD distance based similarity between the question and a document.**

$$Sim(Q, D) = \frac{1}{1 + WMD(Signature(D), Signature(Q))} \tag{3.9}$$

$Signature(D)$ (similarly, $Signature(Q)$) is defined as follows:

$$Signature(D) = \{(x_{D_1}, embedding_{x_{D_1}}), .., (x_{D_n}, embedding_{x_{D_n}})\} \tag{3.10}$$

Where $x_{D_i}$ is the term frequency of the i-th token of the document $D$ (similarly, qusestion $Q$) divided by the number of tokens of $D$ (similarly, Q):

$$x_{D_i} = \frac{tf_i}{\sum_j tf_j} \tag{3.11}$$

The distance between two word embeddings is computed in the WMD implementation as their Euclidean distance. Consult [10] for the definition of WMD.

2. **IDF-weighted WMD distance based similarity between the question and a document.**
   This is the same as in Equation 3.9, with the difference that $x_{D_i}$ is now:

$$x_{D_i} = \frac{tf_i \cdot idf_i}{\sum_j tf_j \cdot idf_j} \tag{3.12}$$

# Chapter 4

# Data

## 4.1 BioASQ DataSet

In this section we briefly describe the data that we used to train and evaluate our re-ranking system. We used the Question Answering datasets of BioASQ from the years 2013, 2014, 2015. The data consist of English questions of four types (Yes/no, Factoid, List, Summary), some PubMed queries for each question and the PubMed ids of relevant abstracts per question.

## 4.2 Obtaining Irrelevant Documents and Abstracts

The main problem we had to deal with was to find irrelevant documents for the questions. We also had to obtain the abstract of each document (relevant and irrelevant). We followed the procedure depicted in Figure 4.1.

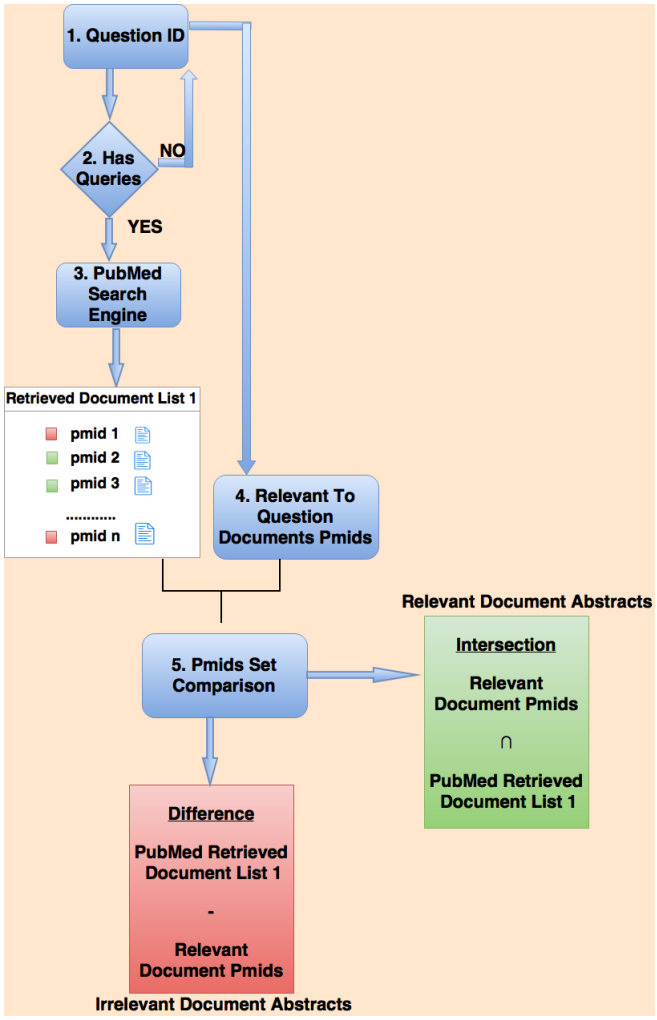


Figure 4.1: Procedure to obtain the abstract of relevant and irrelevant documents.

In detail :

1. For each question, obtain the question ID.
2. Check if there are PubMed queries in the BioASQ dataset for the specific question.
3. If there are no queries, go to the next question.
4. If there are queries for the question, give them to the PubMed search engine and get the results ({PubMed Retrieved Document List} ). The

results are document abstracts (texts) with their corresponding pmids.

5. Get the pmids of the relevant documents for the specific question from the BioASQ dataset({Relevant Documents Pmids}).

Make the following operations:

1. The **irrelevant documents** are obtained by taking the difference of the two sets or more precisely:

   **{PubMed Retrieved Document List} – {Relevant Documents Pmids}**

2. The **relevant documents** are obtained by taking the intersection of the two sets or more precisely:

   **{PubMed Retried Document List} ∩ {Relevant Document Pmids}**

## 4.3 Data Statistics

As Figure 4.2 shows for the first year of the BioASQ competition (2013) we had almost no queries so we ended up with only 5 questions for which we could acquire relevant-irrelevant documents. For this reason we ignored this year's questions.

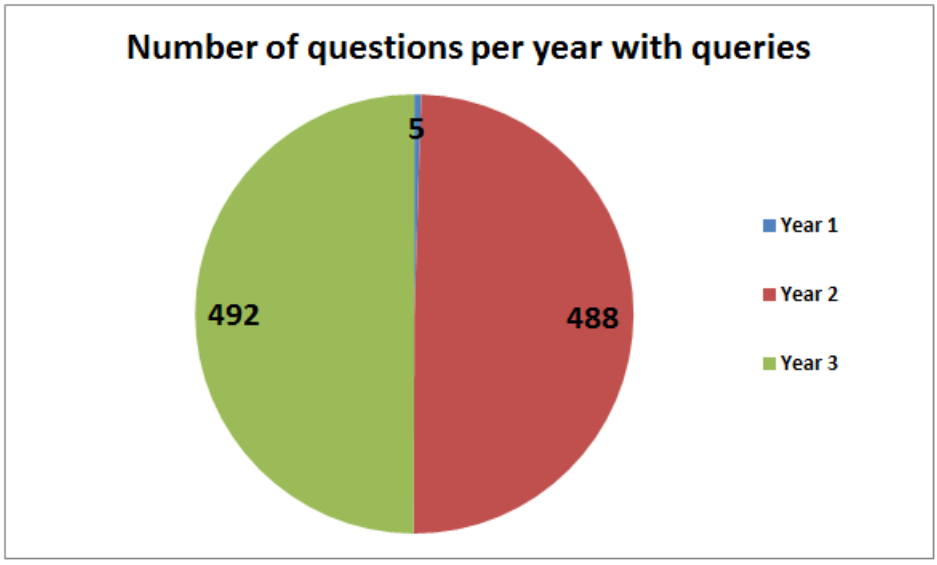


Figure 4.2: Number of questions with queries per year.

Figure 4.3 shows the number of queries per question of the second year (2014). We observe that the majority of the questions do not have more than 10 queries.

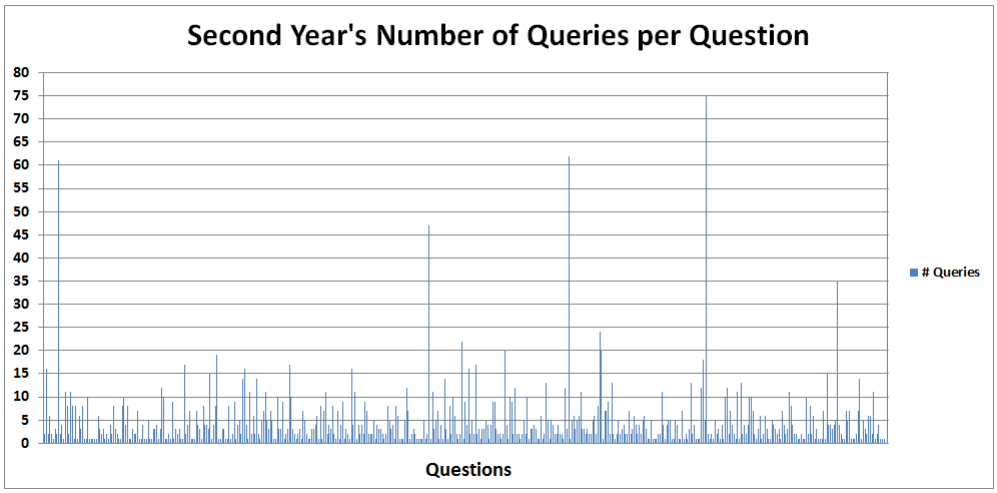


Figure 4.3: Second year's number of queries per question.

Figure 4.4 shows the number of irrelevant documents per question of the second year. We can see that most questions have several hundreds of irrelevant documents.

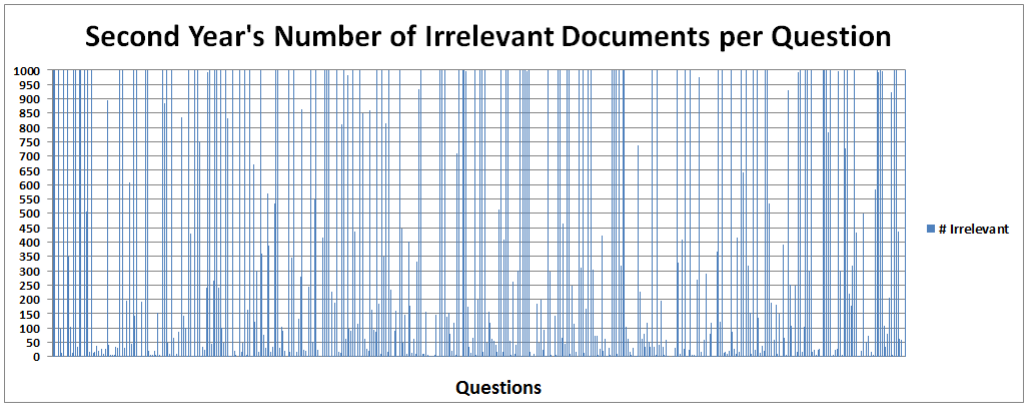


Figure 4.4: Second year's number of irrelevant documents per question.

Figure 4.5 shows the number of relevant documents per question of the second year. We notice that most questions do not have more than 25 relevant documents.

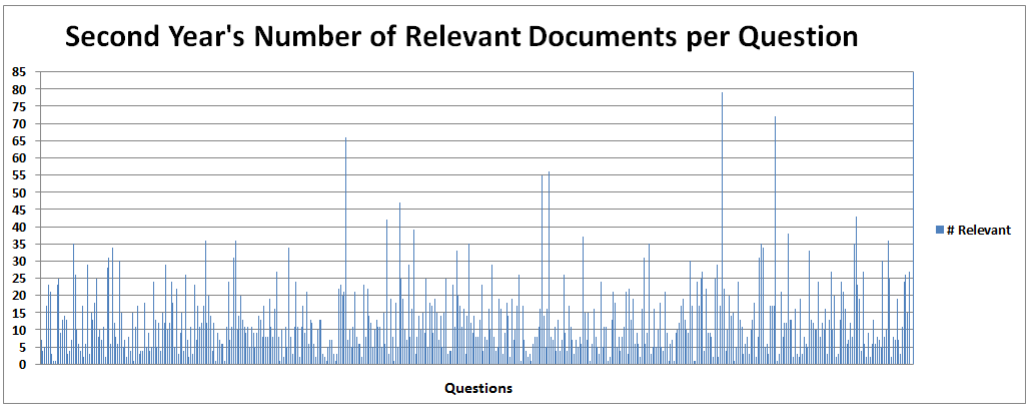


Figure 4.5: Second year's number of relevant documents per question.

Figure 4.6 shows the number of queries per question of the third year (2015). We observe that the majority of the questions do not have more than 10 queries.

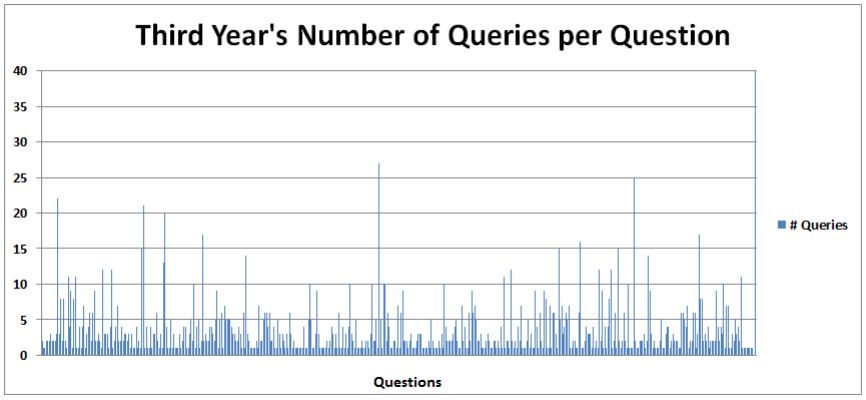


Figure 4.6: Third year's number of queries per question.

Figure 4.7 shows the number of irrelevant documents per question of the third year. We can see that most questions have several hundreds of irrelevant documents.

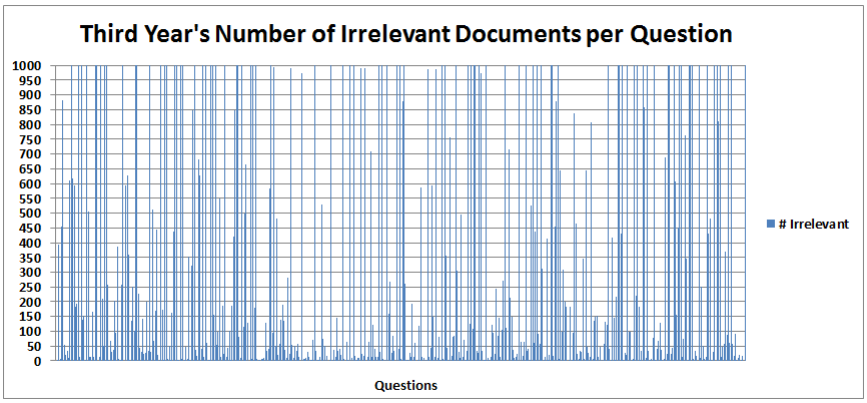


Figure 4.7: Third year's number of irrelevant documents per question.

Figure 4.8 shows the number of relevant documents per question of the third year. We notice that most questions do not have more than 20 relevant documents.

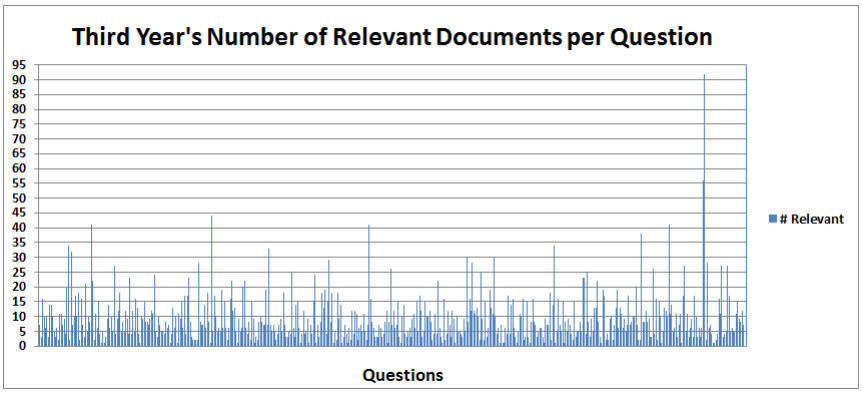


Figure 4.8: Third year's number of relevant documents per question.

## 4.4 Addressing Class Imbalance

From the above diagrams, it is obvious that we have a **class imbalance** problem, since irrelevant documents outnumber the relevant ones [19]. To construct a more balanced dataset for our experiments we under-sampled the majority class (irrelevant documents). In order to find out how many and which irrelevant documents to keep, we studied how far down the list

of the PubMed returned documents we have to go until we find the last relevant document. The following figures depict this for each year's data.

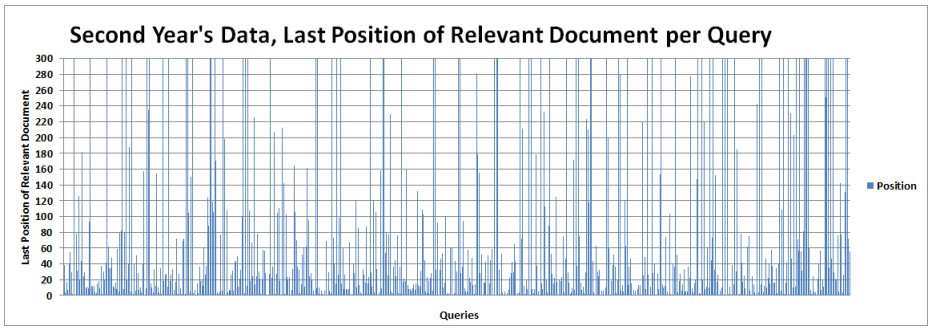


Figure 4.9: Second year's data. How far down the PubMed returned list we have to go to find the last relevant document for each query.

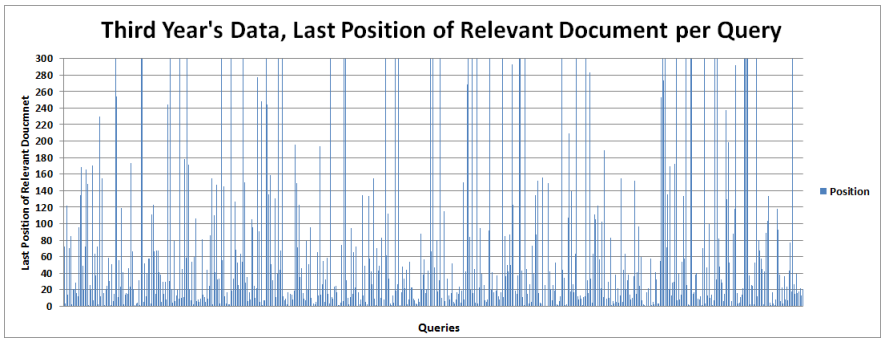


Figure 4.10: Third year's data. How far down the PubMed returned list we have to go to find the last relevant document for each query.

At this point we should mention that BioASQ dataset has some idiosyncrasies:

1. It is not specified which query returns each relevant document.
2. When a query was too general and was returning too many documents, experts could create another one more specific.

As we can see for each query, in general, we do not have to go further than the $100^{th}$ place in order to find all the relevant documents. For the relevant class we decided to keep the top 40 (relevant) documents of each question. If a question had less than 40 relevant documents we kept them

all. If there were more than one queries for a question, we kept the top relevant document of the first query, the top relevant document of the second query etc., then the second relevant document of the first query, the second relevant document of the second query etc., until we had 40 relevant documents in total. For the irrelevant class we used a variable threshold to decide how many (irrelevant) documents to keep, again keeping the top irrelevant documents of the queries when multiple queries per question where available. Figures 4.11 and 4.12 show the proportion of each class for the corresponding thresholds for each year's data.

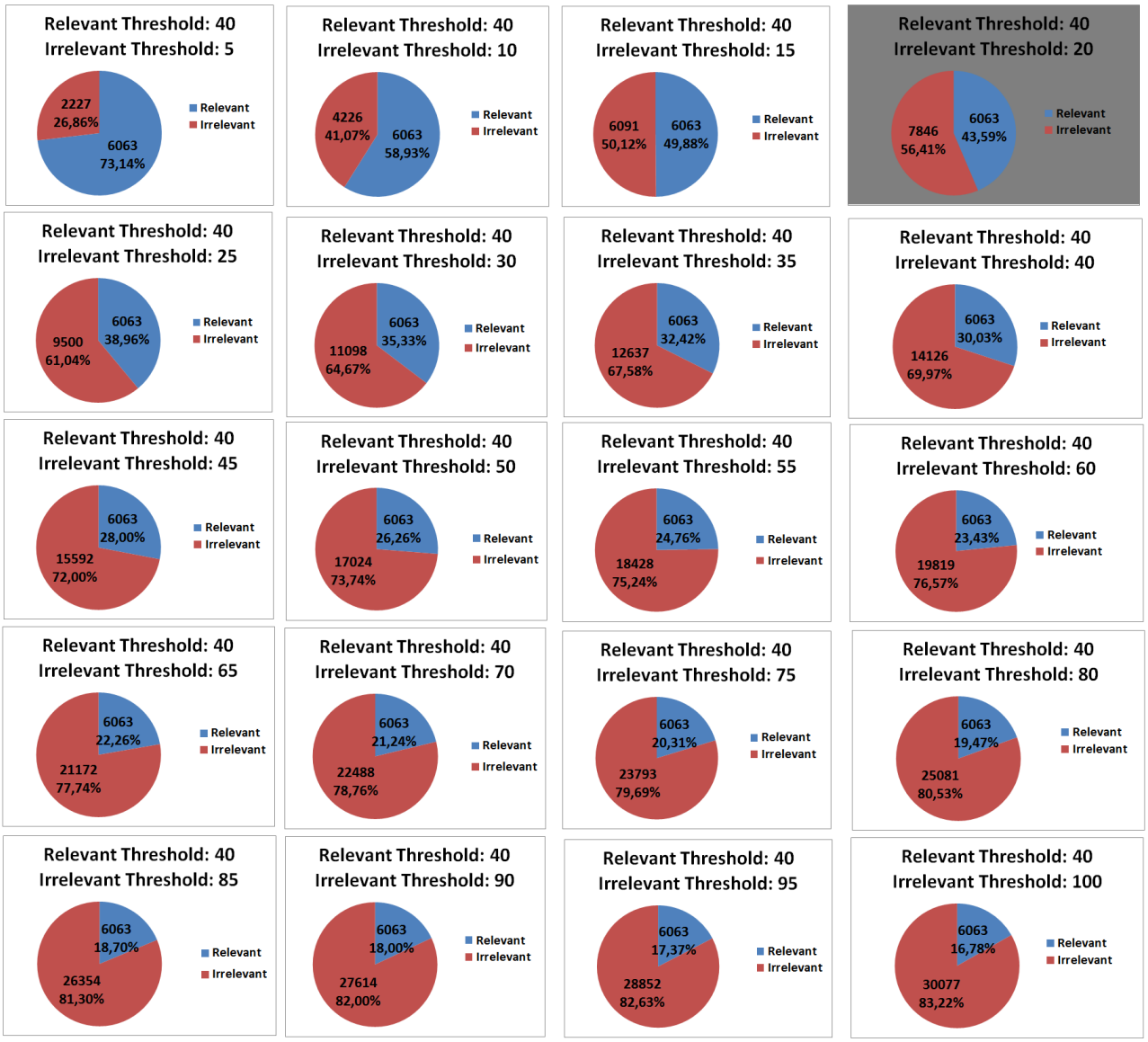


Figure 4.11: Seconds Year's class ratio for each threshold.

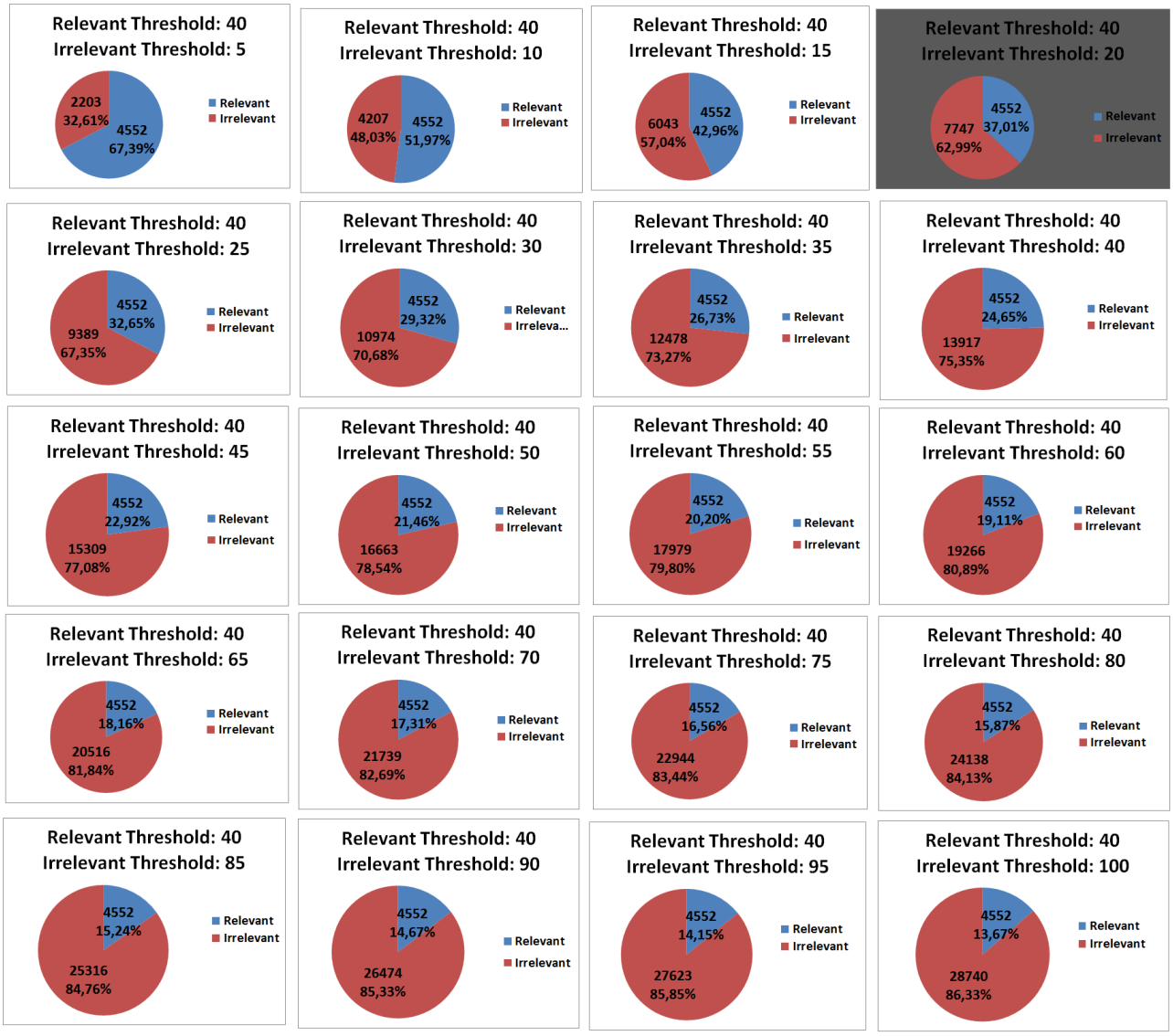


Figure 4.12: Third Year's class ratio for each threshold.

From these figures we concluded that for a relevant threshold at 40 and an irrelevant threshold at 20 we have balanced data.[1] Thus for each question we kept the first 40 relevant documents (or fewer if they did not have that many) and at most 20 irrelevant documents. In practice, most ques-

[1]See gray images in Figures 4.11 and 4.12.

tions have fewer than 40 relevant documents and, hence, there are more irrelevant documents in total, even though the threshold for irrelevant documents (20) is lower than the threshold for the relevant documents (40).

## 4.5 Creating Training - Development - Test Sets

The BioASQ data are organized in five batches. Figures 4.13 and 4.14 show the batches for each year data. We used all five batches of the second year as our training dataset, the first two batches of the third year as our development dataset and the remaining three batches of the third year as our test set.

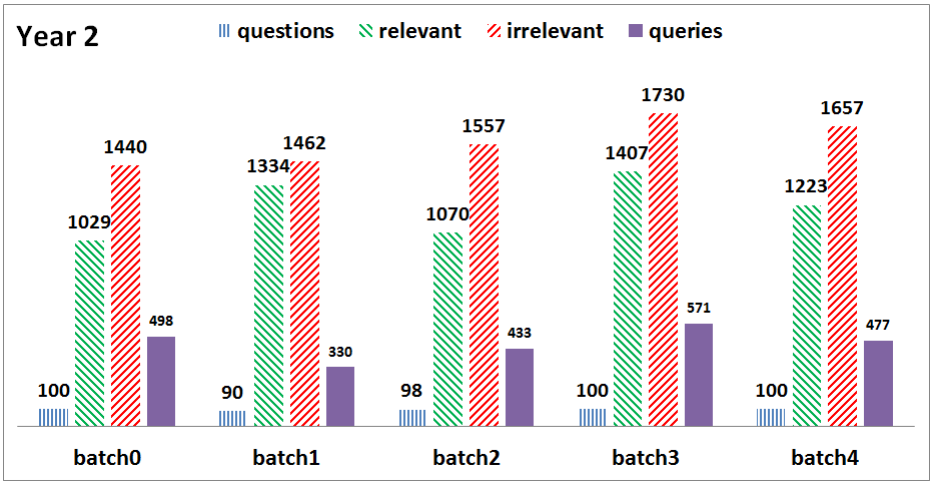


Figure 4.13: Statistics about the batches of the second year (2014).

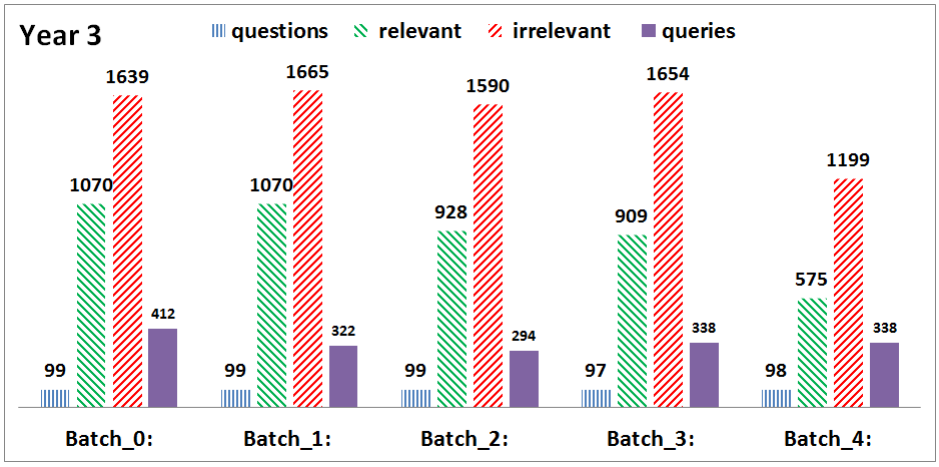


Figure 4.14: Statistics about the batches of the third year (2015).

Figures 4.15 - 4.17 show the ratio of relevant and irrelevant documents in the training development and test data, after the undersampling with the corresponding class thresholds.

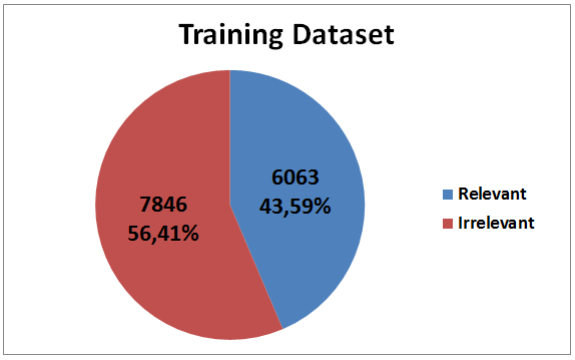


Figure 4.15: Relevant and irrelevant documents in the training dataset.

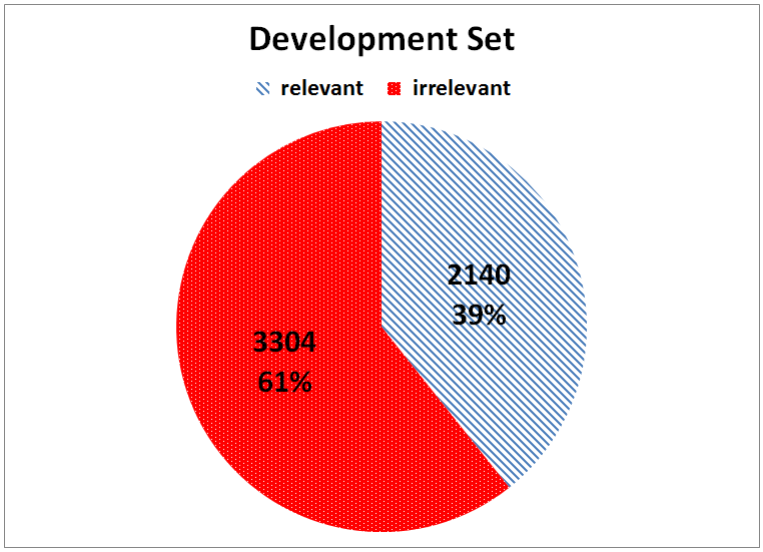

Figure 4.16: Relevant and irrelevant documents in the development dataset.

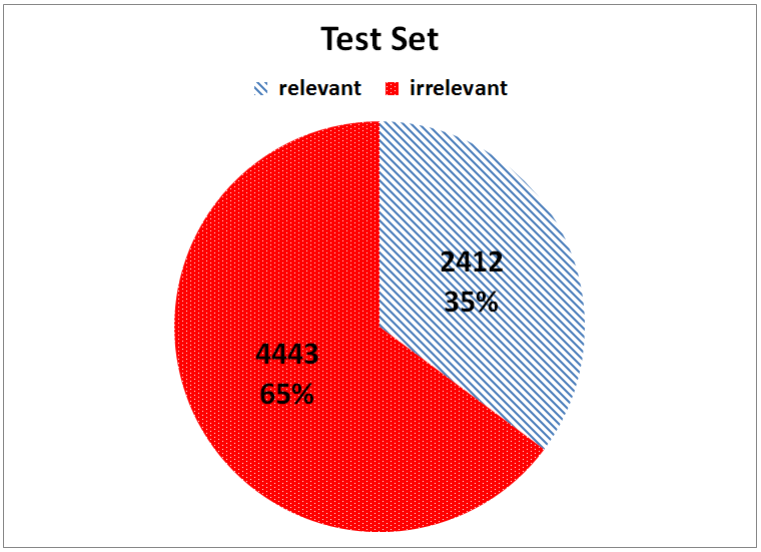

Figure 4.17: Relevant and irrelevant documents in the test dataset.

Figures 4.14-4.20 provide more statistics about the data we used.

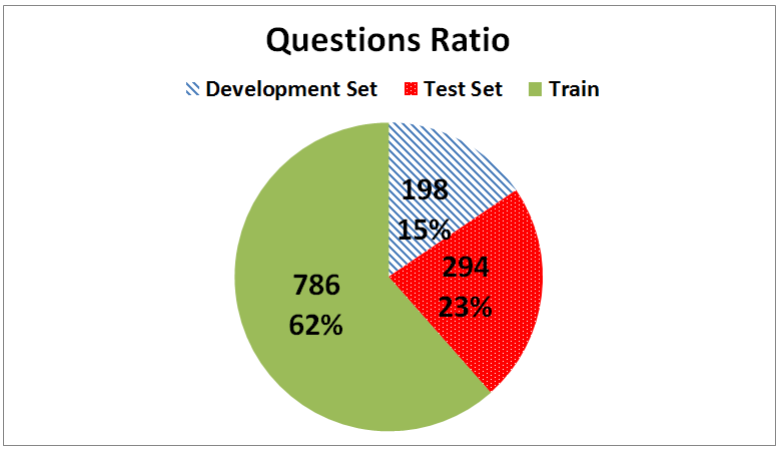

Figure 4.18: Distribution of questions in the training, development and test sets.

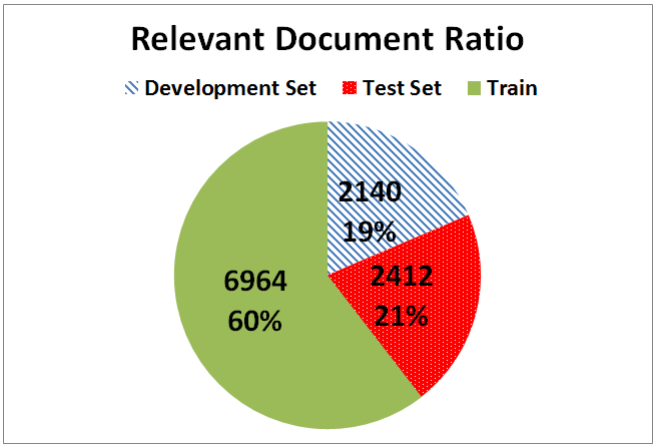

Figure 4.19: Distribution of relevant documents in the training, development and test sets.

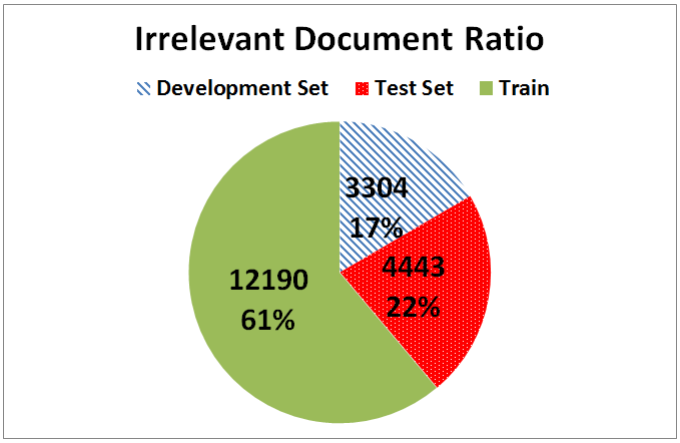


Figure 4.20: Distribution of irrelevant documents in the training, development and test sets.

## 4.6 Data Visualization

Using Weka [6] we created the following diagrams in order to visualize our data. The first 6 features are the ones based on MetaMap, features 7 to 20 are the ones based on word embeddings, and features 21 and 22 are the last ones that use WMD. Each feature is normilized as follows, where $\mu$ is the feature's mean in the training data and $\sigma$ its standard deviation.

$$x = \frac{x - \mu}{3 \cdot \sigma} \tag{4.1}$$

Figure 4.21 depicts the distribution of the feature values of the development data.

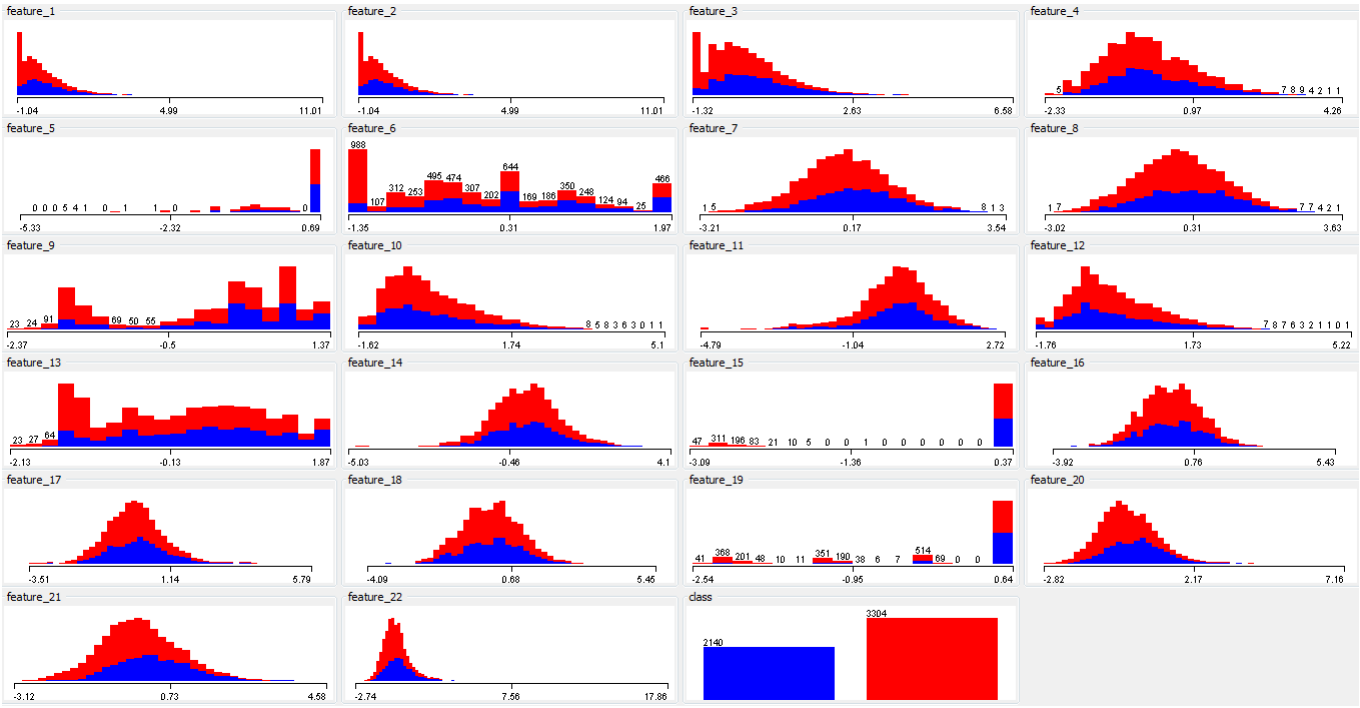


Figure 4.21: Distribution of feature values.

Figure 4.22 depicts the correlation of the MetaMap features and Figure 4.23 the correlation of the embeddings features. In Figure 4.22 we notice that the Dice coefficient of MetaMap **concept IDs** between $Q$ and $S$ (feature_1) and the dice coefficient of MetaMap **preferred names** (feature_2) seem to be linearly correlated. Also feature_2 and feature_3 (Dice coefficient of MetaMap **matched words** between $Q$ and $S$) seem to be positively correlated.

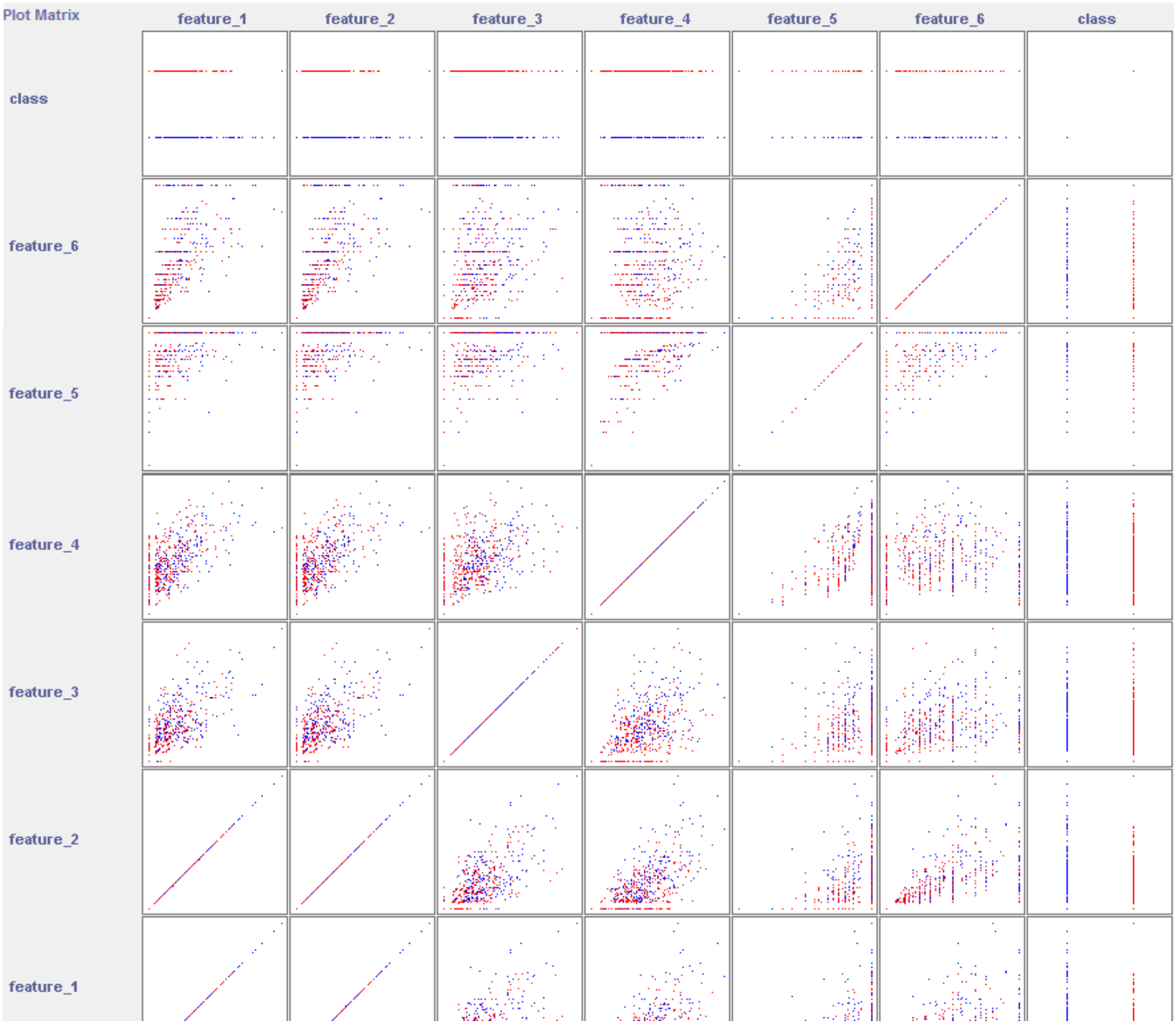


Figure 4.22: Correlation of MetaMap features. Feature_1 and feature_2 are linearly correlated.

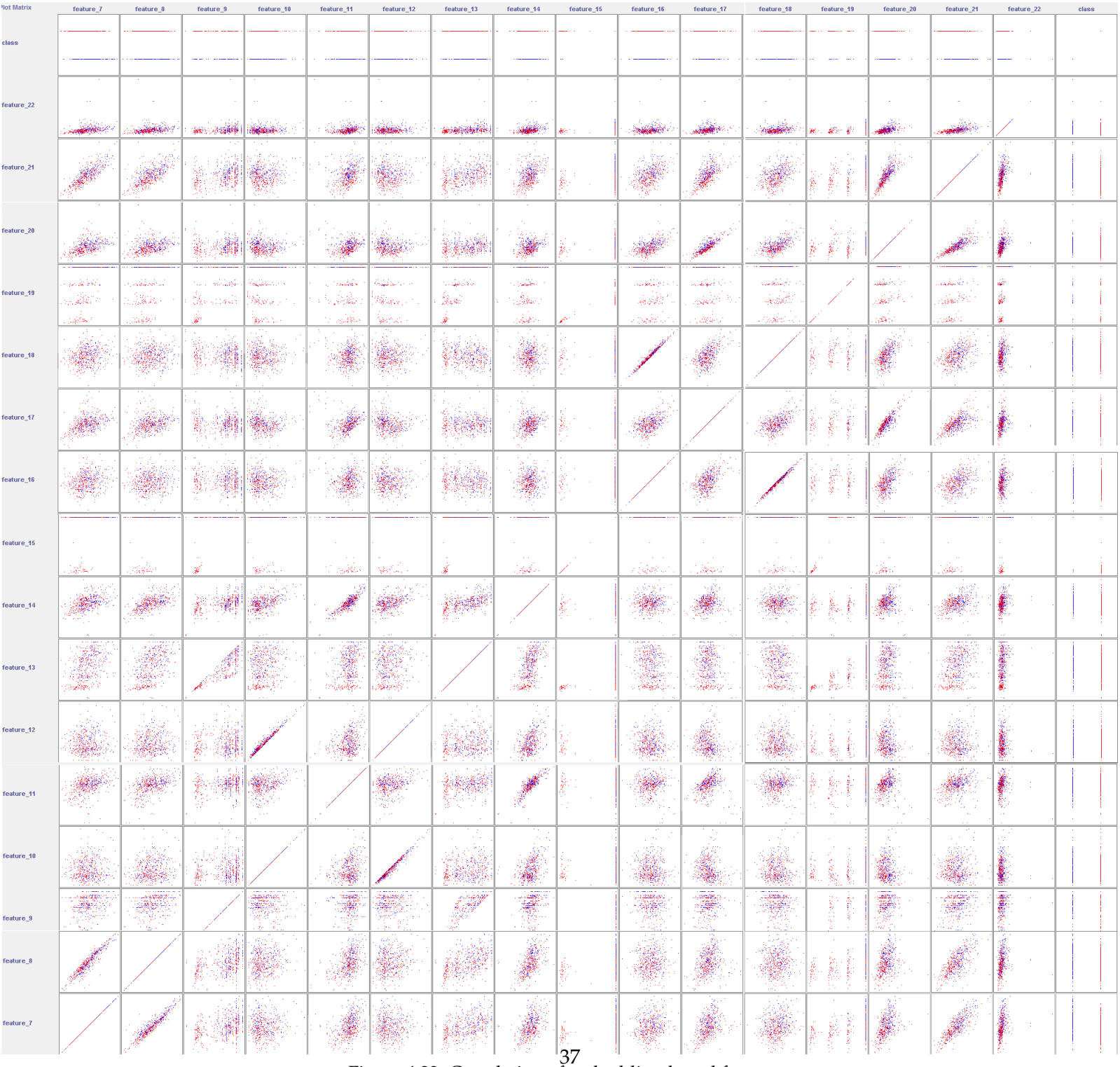




Figure 4.23: Correlation of embedding-based features.

# Chapter 5

# Experiments

## 5.1 Evaluation Measures

In order to evaluate the performance of our re-ranking system we used Mean Interpolated Precision (MIP) at different recall levels, and Mean Average Interpolated Precision (MAIP) scores. In order to compute them we followed the steps described in [5]. For the reader's convenience we repeat them here (with the appropriate modifications) :

- **Mean Interpolated Precision (MIP):** For each question $Q_k$ of the test dataset, we rank each document $D_i$ by descending confidence (of the classifier) that it is a relevant document. We then mark each $D_i$ in the list with its true category (relevant or irrelevant document). Afterwards we compute the precision $P_{Q_k}(D_i)$ and the recall $R_{Q_k}(D_i)$ up to each relevant $D_i$ (including the relevant $D_i$) in the list as follows:

$$P_{Q_k}(D_i) = \frac{|\text{relevant documents of } Q_k \text{ up to } D_i|}{i} \quad (5.1)$$

$$R_{Q_k}(D_i) = \frac{|\text{relevant documents of } Q_k \text{ up to } D_i|}{R_k} \quad (5.2)$$

where $R_k$ is the set of relevant documents for question $Q_k$. For each recall level r $\in$ {0.0,0.1,0.2, .. , 1.0}, we then compute the Interpolated Precision IP$_{Q_k}(r)$ of question $Q_k$ at recall level $r$ as follows:

$$\text{IP}_{Q_k}(r) = \max_{\text{i:}R_{Q_k}(D_i)\geq r} P_{Q_k}(D_i) \tag{5.3}$$

By averaging over all the questions $Q_k$, we obtain the Mean Interpolated Precision at recall level $r$:

$$\text{MIP}(r) = \frac{1}{k} \cdot \sum_{k=1}^{K} \text{IP}_{Q_k(r)} \tag{5.4}$$

where $k$ is the number of test questions.

- **Mean Average Interpolated Precision (MAIP):** To compute the MAIP score of a system, we first average its $\text{IP}_{Q_k}(r)$ over all recall levels, separately for each question $Q_k$, obtaining its Average Interpolated Precision (AIP) for $Q_k$:

$$\text{AIP}_{Q_k} = \frac{1}{11} \cdot \sum_{r\in\{0.0,\ldots,1.0\}} \text{IP}_{Q_k(r)} \tag{5.5}$$

By averaging over all the questions, we obtain the MAIP score:

$$\text{MAIP} = \frac{1}{K} \cdot \sum_{k=1}^{K} \text{AIP}_{Q_k} \tag{5.6}$$

## 5.2 Results on Development Data

In order to evaluate our system, we compare it against 5 baselines:

1. A random re-ranking system.
2. A re-ranking system that uses only the Euclidean distance based similarity using centroids (feature 1 of Section 3.3).
3. A re-ranking system that uses only the Euclidean distance based similarity using idf-weighted centroids (feature 2 of Section 3.3).

4. A re-ranking system that uses only the WMD distance based similarity (feature 1 of Section 3.4).

5. A re-ranking system that uses only the IDF-weighted WMD distance based similarity (feature 2 of Section 3.4).

Figures 5.1 and 5.2 show that Logistic Regression and Rank SVM achieve almost identical scores and have slightly better performances than the baselines. An interesting observation is that the use of idf scores does not help in the case of the baseline that uses centroids of embeddings, and leads to significantly lower performance in the WMD baseline. Also we can see that the centroid baseline without idf is almost identical to the WMD without idf baseline. Because WMD is susceptible to the number of words, perhaps in the future we could try to compute the WMD between the question and each sentence of the document and keep the best.

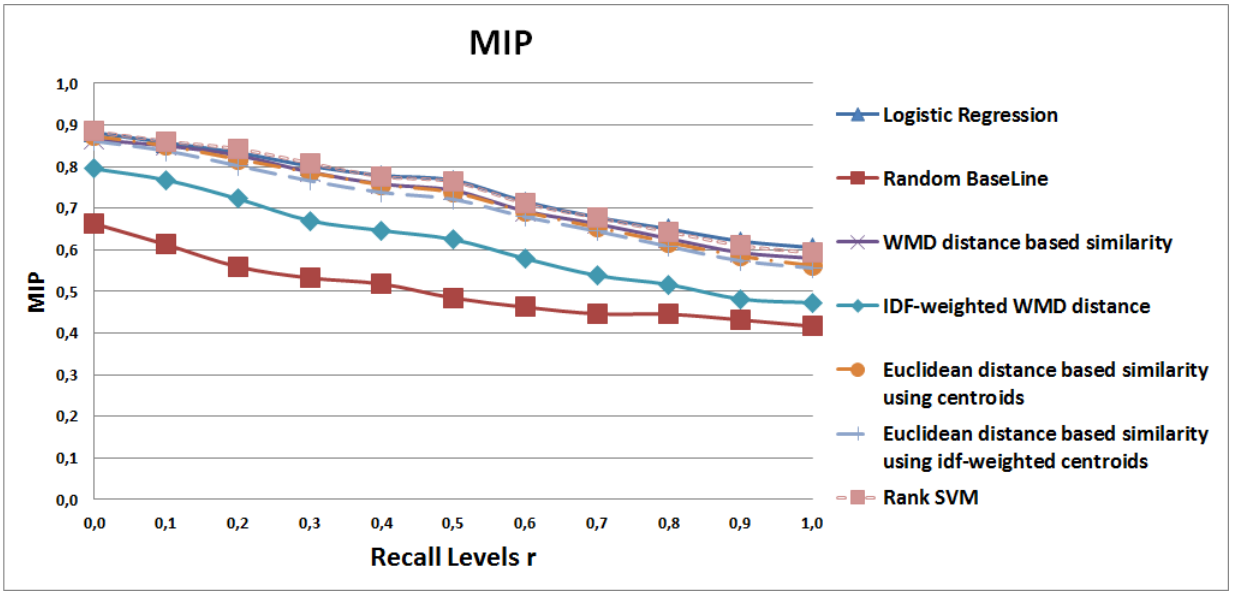


Figure 5.1: Mean interpolated precision at different recall levels.

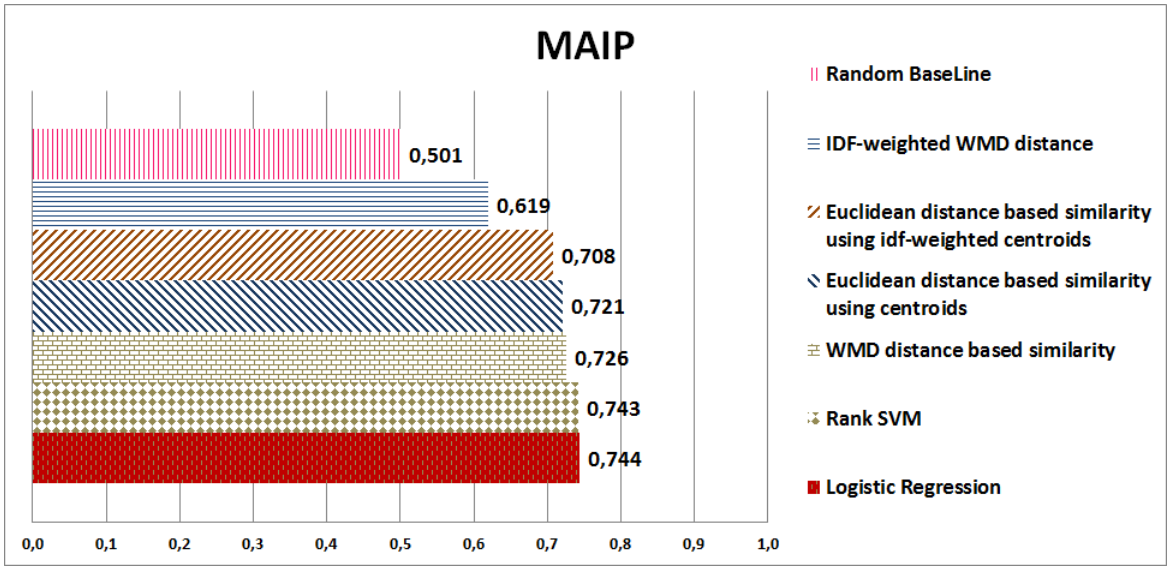


Figure 5.2: Mean average interpolated precision.

## 5.3 Results on Test data

The experiments on the test data (Figures 5.3 and 5.4) show similar results as the ones on the development dataset and the same conclusions are drawn. We notice that the mean average interpolated precision is lower than on the development set.

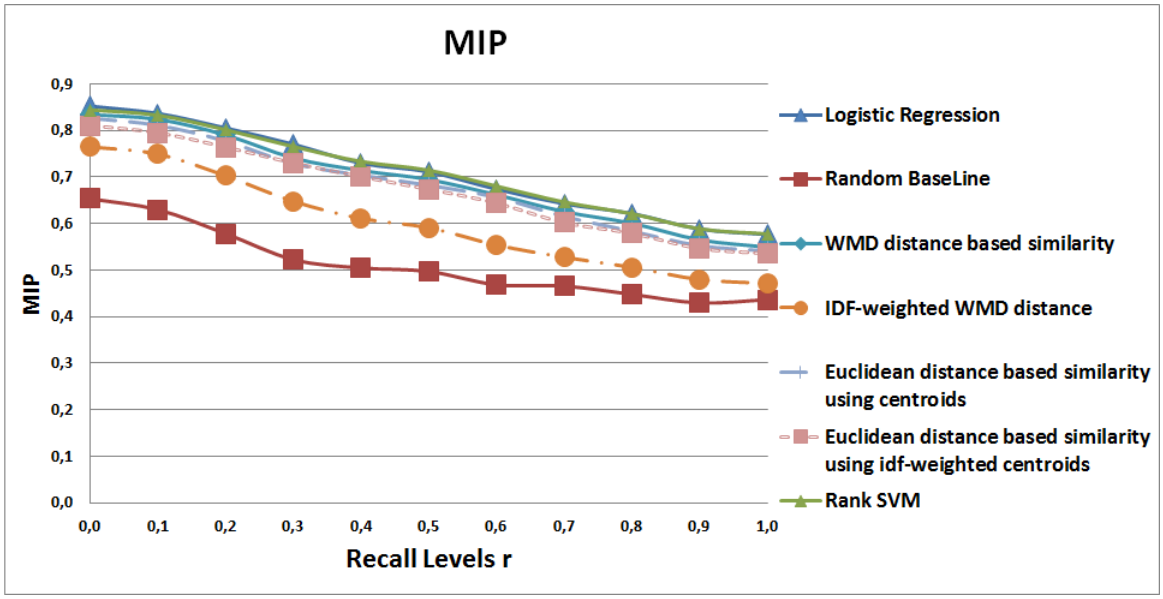


Figure 5.3: Mean interpolated precision at different recall levels on test data.

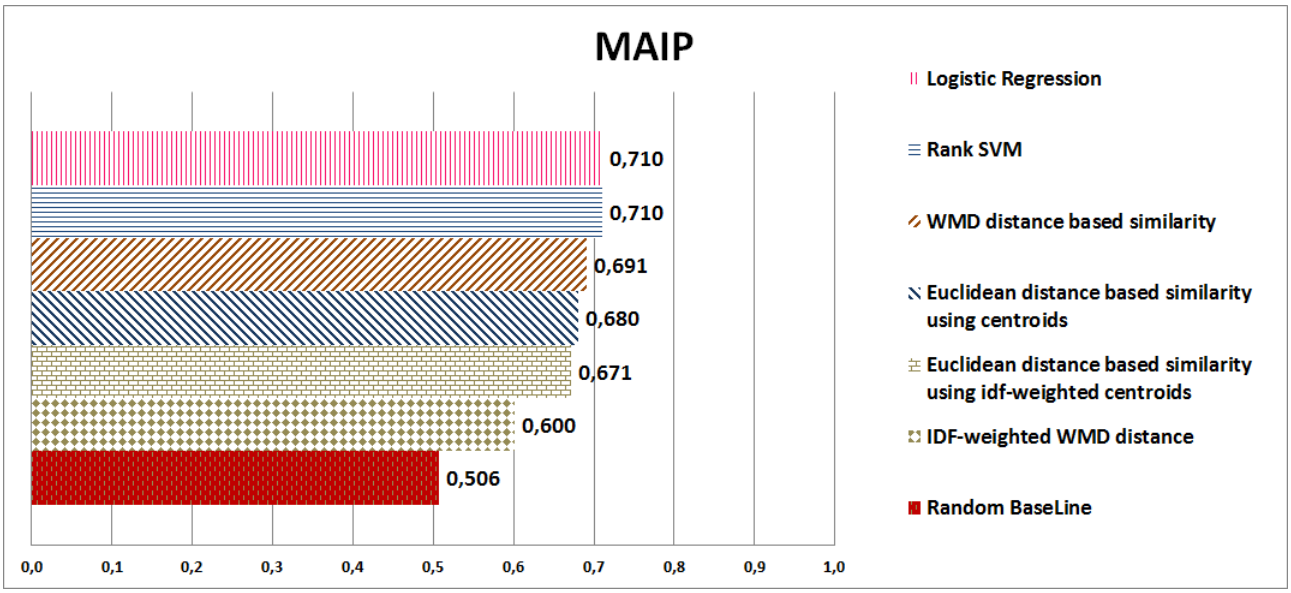


Figure 5.4: Mean average interpolated precision on test data.

Figure 5.5 depicts the mean average interpolated precision per question type. We notice that our system (logistic regression, Rank SVM) has better

results on all types of questions.

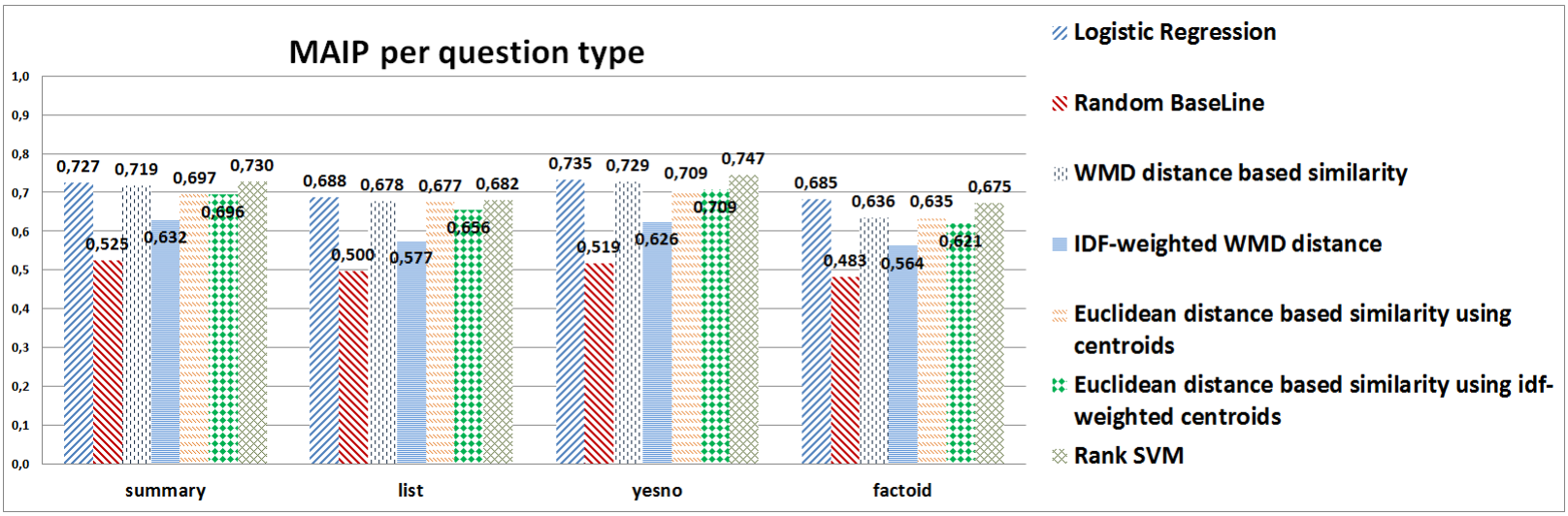


Figure 5.5: Mean average interpolated precision per qusetion type on test data.

Figures 5.6 and 5.7 show that more data will not help our Logistic Regression system.

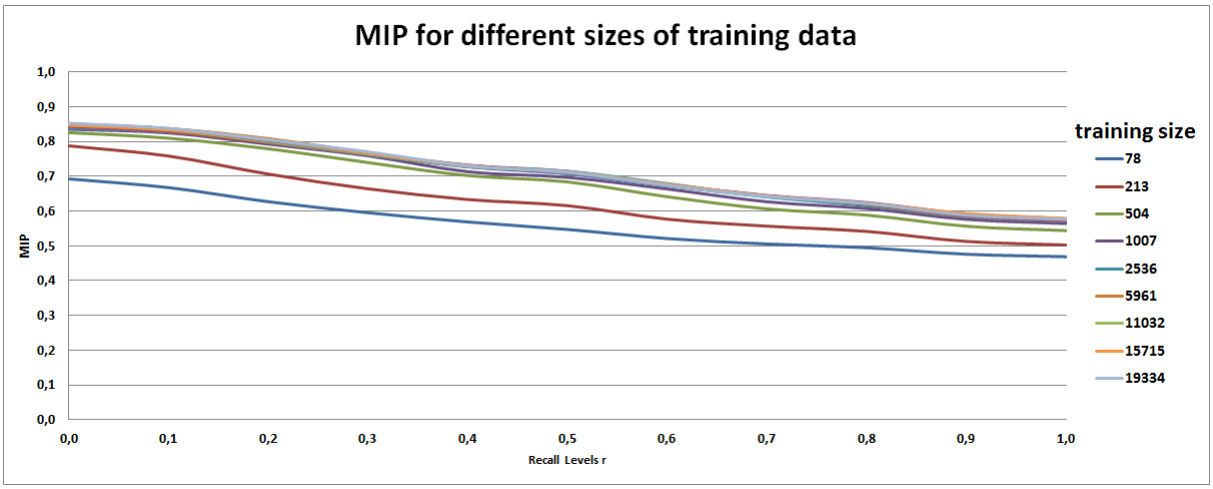


Figure 5.6: Mean interpolated precision at different recall levels for varying size of the training set.

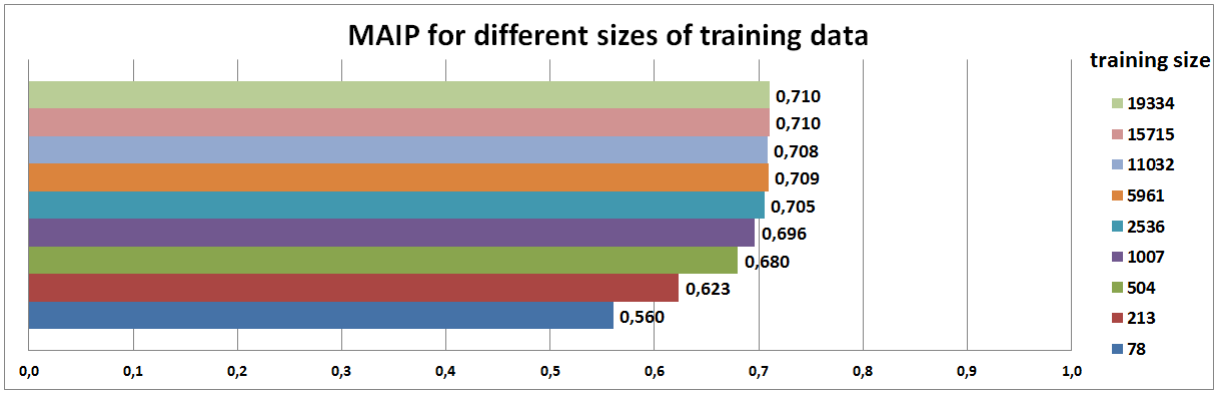


Figure 5.7: Mean average interpolated precision at different recall levels for varying size of the training set.

Finally Figures 5.8 and 5.9 depict the contribution of each feature set in our Logistic Regression system performance. The two feature sets alone have similar scores, but when they are combined they achieve slightly higher results. The WMD based features are included in the "Embeddings".

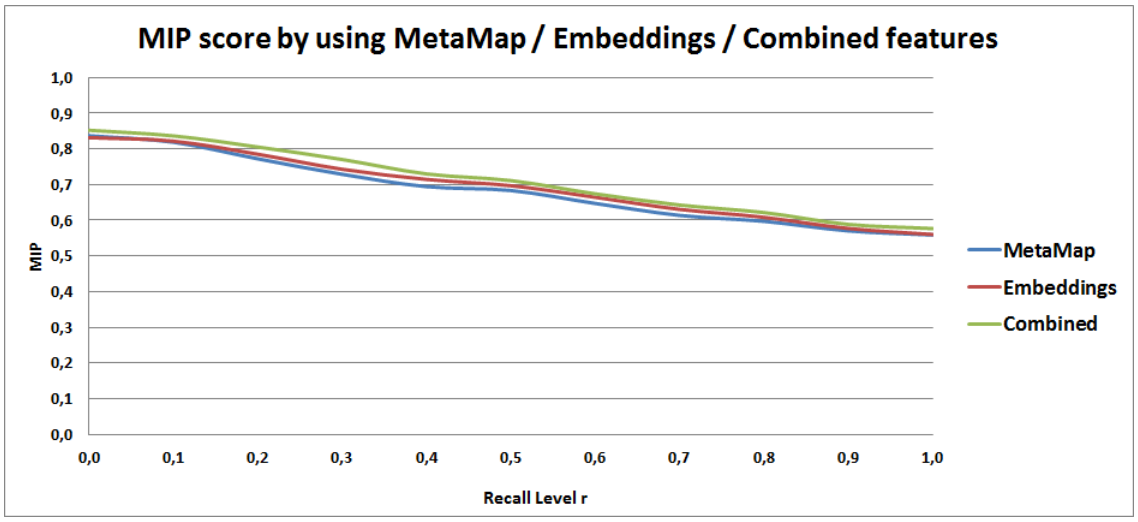


Figure 5.8: Mean interpolated precision for each set of features on test data.

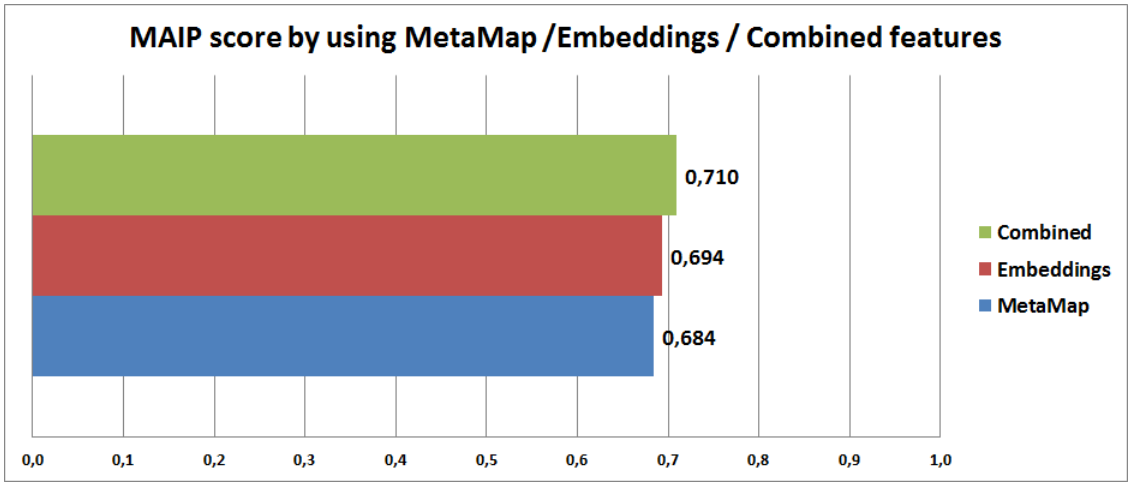


Figure 5.9: Mean average interpolated precision for each set of features on test data.

# Chapter 6

# Related Work

There are several studies on document re-ranking systems which aim to improve the performance of the Information Retrieval phase. In [23] a query expansion method named Q-Rank is proposed. In this method the initial query is used to find "similar adjacent" queries from large query logs. The query logs were used to find lexical overlaps with initial IR results and by this a significant improvement in the initial ranking of a commercial web search engine (MSN Search) was obtained.

In another study [22], for a document to be re-ranked in a higher position it must be both relevant to the question and to provide novel information. The problem of detecting fresh and diverse aspects in biomedical documents was modeled using a probabilistic aproach derived from the survival analysis theory. The more times an aspect is appears, the less probable it is to provide new information. This system uses Wikipedia as an external resource in order to find biomedical aspects; Wikipedia was found to be better than UMLS.

In [11] a re-ranking method based on semi-dynamic hierarchical clustering is presented. The system first uses an inverted-file which returns in decreasing similarity order the documents that have common terms with the query. Each document is represented by a vector which contains evidence of nouns (extracted through a part-of-speech tagging), document frequencies and term frequencies and an appropriate weight for each term. The documents are then clustered using the Reciprocal Nearest Neighbors [15] and Ward's method [21]. Each cluster has a representative vector centroid based on the vectors of the documents that it contains. Similarities scores between the query and each cluster centroid are then computed. Then results of the inverted-file method and the cluster analysis are combined to

provide a re-ranked list to the user.

# Chapter 7

# Conclusions and Future Work

During this thesis a biomedical document re-ranking system, that could be used to augment a QA system, was developed. Our system takes as input a biomedical question and a list of biomedical documents and it re-ranks them promoting to the top the documents it considers relevant to the question. Our system exploits biomedical specific tools (MetaMap) and embeddings of words (produced by Word2Vec). Also a recently proposed distance measure (WMD) was studied, which shows promising results. We used biomedical data to train and evaluate a logistic regression classifier and a ranking SVM, and they outperformed all the baselines. However, the performance of some of the baselines (especially the WMD baseline) was also very close. The evaluation process revealed that more training data would not increase our system's performance. The system was evaluated using mean interpolated precision (MIP) at different recall levels and mean average interpolated precision (MAIP). We note that our experiments were performed on a balanced dataset that we constructed by undersampling the majority class (irrelevant documents). It would be particularly interesting to examine how well our system performs as a component of a question answering system with real-life unbalanced data.

There are a number of interesting approaches that could be tried in order to improve the performance of our system. Higher dimension embeddings could be used as features and instead of Word2Vec the embeddings could be produced with GloVe [18]. Also a system that would take under consideration the novelty of the documents' answers could be built. If a relevant to the question biomedical concept (MeSH heading) appears for the first time in a document the document may contain "fresh" information.

Furthermore, instead of representing a document's abstract as a long sentence, we could split it in real, shorter sentences. Query expansion, maybe with Wikipedia, could be studied and other methods to collect irrelevant documents could be tried. Maybe we could consider as irrelevant the documents that another system has reported as irrelevant.

# References


[1] Alan R Aronson and François-Michel Lang. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, 2010.

[2] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM, 2006.

[3] Matthew E Falagas, Eleni I Pitsouni, George A Malietzis, and Georgios Pappas. Comparison of pubmed, scopus, web of science, and google scholar: strengths and weaknesses. *The FASEB journal*, 22(2):338–342, 2008.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] M. Georgiou. Relevant snippet extraction in biomedical question answerring. 2013.

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[7] William R Hersh, Aaron M Cohen, Phoebe M Roberts, and Hari Krishna Rekapalli. Trec 2006 genomics track overview. In *TREC*, 2006.

[8] William R Hersh, Aaron M Cohen, Phoebe M Roberts, and Hari Krishna Rekapalli. Trec 2007 genomics track overview. In *TREC*, 2007.

[9] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[10] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 957–966, 2015.

[11] K Lee, YoungChan Park, and Key-Sun Choi. Document re-ranking model using clusters. Technical report, KORTERM-TR-99-03, 1999.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[15] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[16] Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In *Computer Vision–ECCV 2008*, pages 495–508. Springer, 2008.

[17] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *Computer vision, 2009 IEEE 12th international conference on*, pages 460–467. IEEE, 2009.

[18] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[19] M Mostafizur Rahman and DN Davis. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3(2):224, 2013.

[20] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al.

An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1, 2015.

[21] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[22] Xiaoshi Yin, Jimmy Xiangji Huang, Zhoujun Li, and Xiaofeng Zhou. A survival modeling approach to biomedical search result diversification using wikipedia. *Knowledge and Data Engineering, IEEE Transactions on*, 25(6):1201–1212, 2013.

[23] Ziming Zhuang and Silviu Cucerzan. Re-ranking search results using query logs. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 860–861. ACM, 2006.