UNIVERSITÀ DEGLI STUDI DI PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

UNIVERSITY OF PADOVA

Entity search: How to build virtual documents leveraging on graph embeddings

Supervisor:

Prof. Gianmaria Silvello

Student:

Anna Ruggero

Master thesis, Computer Engineering

Department of Information Engineering

UNIVERSITY OF PADOVA, ITALY

A.Y. 2018/2019

7 October 2019

"The future belongs to those
who believe in the beauty of their dreams."
Cit. Eleanor Roosevelt

Acknowledgements

I would like to thank my supervisor, Prof. G. Silvello, for the opportunity given me in developing a thesis in an area that I've always been fascinated with and that I'm passionate about. I really appreciate the time you've given me, the patience you showed in accompanying me on this journey, all the moments of constructive discussion and dialogue that were food for thought and a basis for further deepenings. I would also like to thank all the PhD students of the IMS Research Group for your availability and for helping me with my work through your experience, but above all for welcoming me as a friend and letting me face this journey with a smile on my face.

A special thanks goes to my boyfriend Alvise for always be my rock, to support and encourage me even when I am despondent. I thank you for the determination and courage you have in facing life, they have always been an inspiration for me, pushing me to face my fears. Thank you for always believing in me and for making me a better person, I'm lucky to have you by my side. I would also like to thank all my family for letting me get here and pursue my dreams. In particular the most grateful thanks goes to my mother because your strength is an inspiration to me, constantly spurring me on to give my best and never give up. Thank you for your good heart and for always being there, in good and bad times, because *"A shared pain is a pain halved. A shared joy is a doubled joy"*. Afterwards I would like to give a warm thank you to all my friends for sharing this important goal with me, especially Alexa to be the big sister I always wanted. Thank you for sharing with me, step by step, this experience, for all the advice you gave me at any time of day or night. Thank you for your infinite availability, for your support and for the great laughter you give me, without which my life would never be the same. A big thanks also to my friend Chiara for being present in my life from childhood and for being a solar and trustworthy friend. I thank you all because if I have reached this goal it is also because of you.

Abstract

In this thesis we develop an innovative method for the Information Retrieval Entity Search task. In particular, we propose a new approach that exploits graph embedding techniques and clustering in order to create the documents necessary for the retrieval. The main difference with the state-of-the-art implementation is that we create a document for a set of related entities instead of a single document for each single entity. For the documents creation and evaluation phases we use DBpedia, a standard dataset for entity search task. This dataset is constituted by RDF triples that describe entities properties and that allow us to consider all the stored knowledge as a graph. It is on this graph that we use a graph embedding technique that creates a vector representation for each node considering, not only the entity properties, but also the information concerning its neighborhood and relationships with other entities. These embeddings are then grouped through clustering to obtain sets of similar entities that we use as our documents. For the retrieval we develop two classes of systems: one that uses only the cluster approach, the other that merges the cluster method together with the state-of-the-art one. The main advantage of proposed implementation is that our systems could return to the user not only entities that directly match the user query, but also relevant entities that are not explicitly mentioned. We execute two different type of evaluation: average evaluation and topic-based evaluation. From the average evaluation we see that, in general, some of our systems turn out to be promising and obtain better results compared to the state-of-the art method; from the topic-based evaluation we see that some of our systems outperform the classic one in some specific topics while in others we obtain very similar performances.

Keywords – Entity Search, Information Retrieval, Graph Embeddings

Contents

List of Figures

List of Tables

1 Introduction

When we are looking for information, it is now customary to open a search engine in our mobile phones or computers and type, in the specific search bar, that query we think best represents our research. This process has become part of people's everyday life and it almost never leads us to think about the mechanism behind that allows us to achieve all this. The information retrieval is the field that study this process, it started way before the 1990s analyzing textual documents and continue today on increasingly specific and current issues. The analysis and the retrieval of textual documents is hence the basic method on which subsequent research has been developed and it remains the reference system today. This method is characterized by the use of a collection of documents, where each document contains text written in natural language. The goal of the information retrieval is to obtain the most relevant documents for the user information need. This approach has the advantage of facilitating the user in making his request because he can use the common language and does not need to know any programming language or the structure of the data that contains the documents. In recent years, however, we are witnessing a change of tendency that leads us from documents to entities, this is due to the fact that nowadays the 40% of the users queries mention entities [10, 9]. If we think about the researches we usually carry out, in fact, it is easy to see that what we often require are information that concern people, places, concepts that can be intuitively associated with concrete entities of the real world. The task of searching these entities takes the name of Entity Search task. In order to give to the user an effective answer it is necessary to have a collection of entities, the dataset on which we execute the retrieval. Usually these datasets are expressed through a knowledge representation model called Resource Description Framework (RDF) that is composed by triples, each triple is composed by three components: subject, predicate and object. Thanks to its structure, this type of representation can be seen as a graph, where subjects and objects are nodes, while predicates are edges. In order to bring the search for entities back to the traditional retrieval method, it is necessary to create documents starting from the entities of the dataset. Traditional text retrieval methods rely on a collection of documents as information base therefore, in order to exploit the same approaches, we have to express our knowledge through documents. In particular we have to create a collection of documents based on entities information in order to use it as our base for the retrieval process. The difficulty of this process lies in the way documents are created. Currently the state-of-the-art methods [10, 32, 6] generates a single document for each entity. This document is created mainly using two approaches: text-based approach and structured-based approach. While the first one simply merges all the information concerning a single entity into a document, the second one takes into account also the type of information that we are storing through the creation of a fielded document. Documents creation constitute hence the main difference between textual and entity retrieval. The main issue with state-of-the-art methods is that they consider only information regarding entities themselves, without considering the context in which they are placed. The innovative approach that we want to introduce in this thesis wants to overcome this problem considering also the entity context.

In our thesis we exploit graph embedding techniques and clustering as basis for the document creation phase. In particular we use graph embeddings to obtain a vector representation for each single entity. These embeddings are generated through the Node2Vec method based on neural network [18], which differs from other algorithms, such as DeepWalk [35] or Line [43], in how it explores the RDF graph in order to generate the input data for the training phase. The embedding features of the vectors returned by Node2Vec contain therefore information concerning the context of the entities and their relationships. The clustering phase exploits these representations in order to create sets of correlated and similar entities that become the documents necessary for the retrieval. The advantage of using this new approach is that it is not simply based on word matching between user query and documents such as the state-of-the-art method, but it allows us to retrieve also entities that are not explicitly mentioned but that are relevant for the topic. This could be done through graph embedding techniques and clustering because our documents contain not a single entity but a set of related ones. It is in fact reasonable to think that if two entities are highly connected, both of them could be considered relevant for a specific topic.

Wanting to outline the steps involved in our approach implementation: we firstly create a database for the storage of all the RDF triples, then we represent each entity with a numeric vector obtained from the embedding process. Once obtained our entities embeddings, we execute a clustering on these vectors and use these clusters as documents in the retrieval phase. The retrieval phase uses the classic BM25 method and returns a ranked list of clusters from which we extract the entities necessary for the final ranked list generation. We implement also systems that use both our approach and the classic one. Finally, we perform an analysis of these systems considering both the average and topic-based evaluation. This phase is executed using DBpedia, a standard dataset for entity search task. From the evaluation studies, some of our systems turn out to be promising and obtain results comparable to the one of the state-of-the-art method. In particular we see that these systems perform better in some specific topic, while in other they obtain the same result of the classic method.

This thesis is organized as follows: in Chapter 2, we give an overview on information retrieval, on entity search and on graph embedding techniques; in Chapter 3, we describe our proposed solution in all its phases; in Chapter 4, we describe the experimental setup, we perform an analysis of our systems performances through both a quantitative and a qualitative evaluation, and we discuss the obtained results; we conclude and outline future directions in Chapter 5.

2 Background

Thanks to the spread of the internet connections and the use of mobile phone, an increasing number of people have had access to the web and in particular to search engines. Everyday more than 4 billion searches are done to the most popular retrieval systems because it has become the fastest and simplest way to answer our questions [12]. This trend brought more attention to information retrieval systems, which have to be continuously updated and optimized. In the last years, indeed, the focus of information retrieval is increasingly shifting to entity search, i.e. the research of specific objects or facts that belongs to our real-world [37].

In this chapter we discuss about text information retrieval in Section 2.1 describing its phases. In Section 2.2 we describe Entity search,the DBpedia dataset and we present the state-of-the-art methods for ad-hoc entity search task. In Section 2.3 we discuss about Graph Embeddings presenting the state-of-the-art methods.

2.1 Text Information Retrieval

The information retrieval is the field that concerned with the management and research of information, indeed as defined by Salton in [41]: *Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.* Around the 90s the launch of the Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards Technology and U.S Department of Defense as part of the TIPSTER Text Program [45], contributed to the development of Information Retrieval. This convention greatly affected the research and development of the information retrieval providing extensive text collection for the large-scale evaluation of the retrieval systems. Every year a task is selected and discussed, in particular from 90s to nowadays we can see that the focus of each meeting has changed and grown together with the optimization of old techniques, the discover of new ones and the introduction of new needs. Summarizing briefly the main tracks that lay the groundwork for Entity Retrieval [3]:

- 1992 - TREC;
- 1999 - TREC Question Answering Track;

- 2009 - INEX Entity Ranking and TREC Entity Track;
- 2012 - TREC Knowledge Base Acceleration Track.

The main advantage of the information retrieval system is the simplicity of the query formulation; the user can be unaware of the complexity or structure of the data storage and he does not need to be an expert with the knowledge of specific technical query language. A user generally expresses his information need through a text, usually in natural language, and the information retrieval system tries to match this request with an information object. In this process we can identify three main components: the information need express through a query, the retrieval system and the information object.

From the TREC overview [3, 45] we can see that traditionally the retrieval mainly concerns the research of information inside a set of documents from which it obtains a list of ordered pages that are supposed to be of major interest for the user. Driven by the motto *"users want answers, not documents"* a new front of IR was introduced with the TREC Question Answering Track [3]; here the structure of both the query and results has been modified: the first became a question formulated in natural language, while the second became a short answer. In the 2009 conference it was firstly introduced the entity search which we will describe in more detail in the next section, briefly it consists in the research of specific objects or facts that belongs to the real-world. With this track and the use of entities as the main information objects we have had an increase in the reliance on structured data sources called knowledge bases, leading this trend to became the main track of the conference of the 2012. This last mentioned conference aims at developing technologies that can expand and maintain these entity information structures.

2.1.1 IR Phases

The Information Retrieval process, as shown in Figure 2.1, is composed mostly of four parts:

1. Acquisition;
2. Indexing;
3. Retrieval;
4. Evaluation.

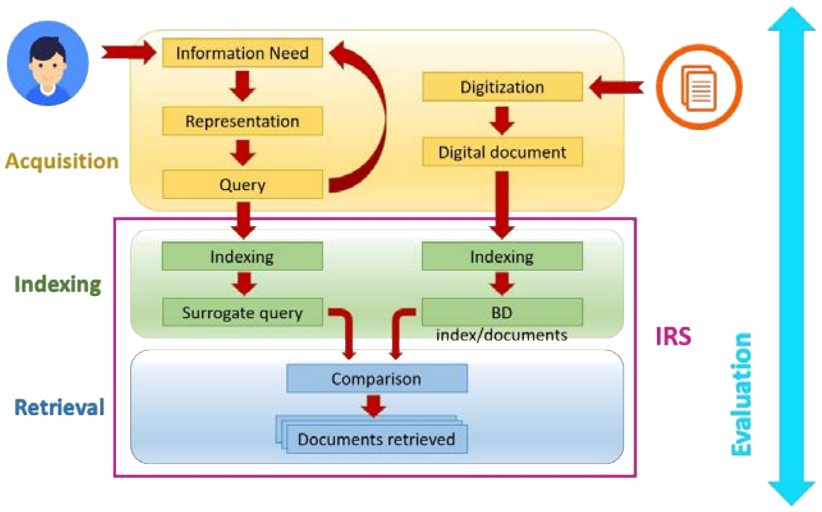


Figure 2.1: U model of an information retrieval process and its component and phases. In particular it reports the acquisition phase execute before retrieval, the Information Retrieval System (IRS) and the last evaluation phase that covers all the process.

The first phase describes the process of query and document acquisition, in particular the digitization of textual documents and the representation of the user information need through a query. In the second phase the system processes documents analysing their form and content in order to create a structured representation that is more useful for our retrieval purposes. This is supported by the use of a specific data structure called *Inverted Index* which connects a term with the documents in which it appears. This structure is used by the retrieval system which, first of all, executes a lexical manipulation on the user query and then does a match between this form of query and the inverted index through a specific retrieval model in order to find the most relevant documents. At the end of the all retrieval process there is the evaluation, whose aim is to rate the efficiency and the effectiveness of the system. This is done using different type of measures. Thanks to these measures and the possibility of reproducing some scientific studies we can make a comparison between different systems and different retrieval models, using the results to improve them.

2.1.1.1 Indexing

The processing of documents is executed in the indexing phase. This phase consist of five sub-parts, shown in Figure 2.2, that are not always all

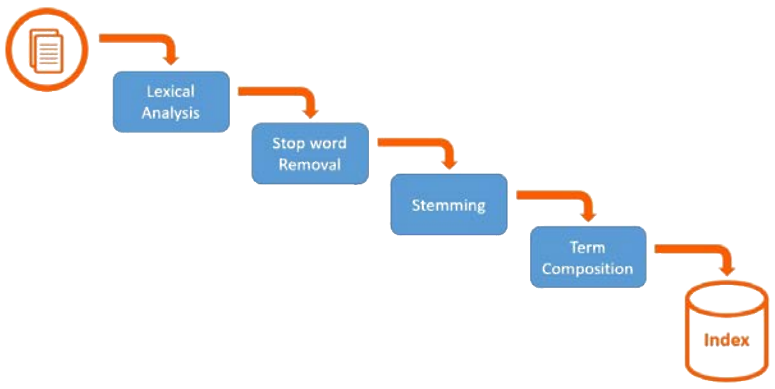


Figure 2.2: The five different phases of the indexing process.

implemented:

- Lexical Analysis;
- Stop words removal;
- Stemming;
- Terms composition;
- Weights assignment.

In the lexical analysis the documents are examined in order to find the various tokens contained within them. In case of textual documents the tokens are the terms and they can be identified thanks to separator characters like white spaces and punctuation marks. After this phase a stop word removal is done. A word is a *stop word* if it has poor information content, namely those words that are very common in a language or that are function words; generally these are: propositions, articles, etc. It would be better to remove them as little as possible in order to maintain flexibility and recognize also phrases with a high frequency of stop words like: *To be or not to be.* Once we have our words we can observe that some of them only differ for their inflection or derivation while their semantic meaning remains the same. We can think on reducing these words into the same one, for example removing their suffixes and maintaining the root of the word, called *stem*, evaluating their frequency for the index creation. This process takes the name of *Stemming.* Instead of reducing words into a single one we can also think of composing terms that usually appear together, this process can help to make stemming less generic. For example instead of using the single stem: "*comput*" we can give more information about the context composing it with the stem: "*data*" obtaining: "*data comput*. The last phase of indexing is the weights assignment. Here we associate to each word a value that represents its importance in each document where it appears. Indeed it is logic to think that not all the words in a document describe its information meaning with the same effectiveness. The weight we assign could be of two types: binary or frequency based. The binary value simply identifies if the word is present in the document while the frequency identifies the number of word occurrences in the document (*Term Frequency - TF*) or in the collection (*Inverse Document Frequency - IDF*).

2.1.1.2 Retrieval

The retrieval phase exploits the information obtained from the indexing in order to give at the user the documents that come closest the information need of the user. In particular it uses term frequency at the base of more complex statistical models in order to give to each document a score that represents its relevance respect to the user query.

Historically the first information retrieval model was the *Boolean model*. In this system the user information need, expressed by a query, it is associated with the set of documents that make that interrogation true. This model does not rank the resulting documents because it simply assign a value of 1, marking as relevant, those ones that match the query, and a value of 0, marking as not relevant, those one that do not match the query. This model has two main issues: one is that the user needs to know boolean operations and a somehow structured language in order to use correctly the system and obtain the information required; the second is that the returned list of documents is not ordered so we need a further external step called *Coordination level* that returns a ranking list basing on a measure of how much a query is true for a specific document.

The necessity of having a ranked list of documents in response to a query and the necessity of giving different weights to documents' terms relying on their frequencies bring to the development of another model called *Vector Space Model*. This model represents each document like a vector of dimension t where t is the number of index terms and d_{ij} represents the weight of the j-th term in the i-th document D_i:

$$D_i = (d_{i1}, d_{i2}, ..., d_{ij}, ..., d_{it})$$

The same structure is associated to the query and the terms that compose it:

$$Q = (q_1, q_2, ..., q_j, ..., q_t)$$

where t is the number of index terms and q_j represents the weight of the j-th term in the query. In the real case, for both the document and the query, t is the number of terms of all the collection.

Representing documents and query as vectors allows us to compare this two elements through a measure of similarity called *cosine similarity*. This measure is defined as:

$$\text{Cosine}(D_i, Q) = \frac{\sum_j (d_{ij} \cdot q_j)}{\sqrt{\sum_j d_{ij}^2 \cdot \sum_j q_j^2}}$$

and takes value 0 if the two vectors has no common terms or features, value 1 if the two vectors are identical.

Once the model has been chosen, the weighting scheme to be adopted remains to be decided. Most common ones are those based on term frequency and inverse document frequency. The first one reflects the importance of a term in a specific document while the second one reflects the importance of a term in the collection. For the IDF we can notice that a term that is present in many documents will be less significant for the description of the information content of the documents. The term frequency associated to the term k in the document D_i with j between 1 and t is defined as:

$$\text{tf}_{ik} = \frac{f_{ik}}{\sum_j f_{ij}}$$

where f_{ij} is the number of occurrences of the term t_j in the document D_i and $\sum_j f_{ij}$ is the sum of the frequencies of all the terms in the document D_i. Usually the lengths of the various documents can be very different one to each other therefore, in order to reduce the impact of terms that appear many time in a document just because of its dimension, it is preferable to use TF in a logarithmic form like:

$$\text{tf}_{ik} = \frac{\log(f_{ik} + 1)}{\sum_j \log(f_{ik} + 1)}$$

where the factor "+1" is a correction factor used to avoid the undefined value obtained for $f_{ik} = 0$. The inverse document frequency instead is defined as:

$$\text{idf}_k = \log\left(\frac{N}{n_k}\right)$$

where N is the number of documents and n_k is the total number of documents in which the term k is present.
Using this two measures we can now define a weighting scheme for documents and queries. For documents we have:

$$d_{ik} = \log(f_{ik} + 1) \cdot \log\left(\frac{N}{n_k}\right)$$

while for queries we have:

$$q_k = \log(f_k + 1) \cdot \log\left(\frac{N}{n_k}\right)$$

Noting that IR systems rely on probability because we do not know with certainty if a document is relevant for a query or not and we only search for a best match between them; in the 70s a new type of model was proposed: the *Probabilistic Model*. Here the idea is to design a weighting scheme based on a probability distribution. The most common probabilistic model used is the BM25 described in the paper [40].

2.1.1.3 Evaluation

From the phases described above we notice that a high flexibility is provided choosing which parts of indexing to implement and which kind of retrieval model to use. This peculiarity leads us to ask ourselves which kind of retrieval pipeline is best suited to our needs and so can give us better results and performances. We would like to make different experiments with different choices in retrieval phases and compare them.

The standard de-facto for the evaluation process is the Cranfield paradigm. It was developed in the 60s by Cyril Cleverdon, a librarian of the College of Aeronautics in Cranfield. He execute two experiments, one following the other and call respectively: *Cranfield 1* and *Cranfield 2*. With the first one Cleverdon wanted to evaluate the most common retrieval systems. In order to do this he chose 3 indexers to index 18000 articles with all the retrieval systems, than he chose 1500 queries, based on the *known item search* task, to formulate and evaluate on the 4 indexes. The known item search task aims to identify a unique document that is surely relevant for the query. 400 of the 1500 queries were then verified by experts in order to ensure that they were questions that the users would really do. He evaluated the time spent for the research and if it was successful or not. The 35% of the research were unsuccessful, probably due to the indexing phase. This was the first example of *Failure Analysis* in IR. With the second experiment Cleverdon therefore wanted to repeat the process in a more structured manner. He chose the documents and the queries for the retrieval and built preliminary a collection for the experiment. The notion of collection was introduced now for the first time. The *Collection* is the set of documents on which we execute the retrieval. Formally a collection is defined as a triple: $C = \{D, T, RJ\}$ where C is the experimental collection, D is the set of documents, T is the set of topics and RJ is the set of relevance judgments. In order to obtain the set of relevance judgments a method called *Pooling* can be used. This process selects a subset of documents to evaluate and use this one to measure the effectiveness of the retrieval systems. The pooling exploits the concept of run for the selection of the subset of documents. A run is formally defined as [2]:

Let:
$D = \{d_1, d_2, ..., d_n\}$ be a set of documents
$T = \{t_1, t_2, ..., t_m\}$ be a set of topic
Given a natural number $N \in \mathbb{N}^+$ called *run length*, a **run** is defined as the function:

$$R : T \to D^N$$
$$t \mapsto r_t = (d_1, d_2, ..., d_N)$$

such as $\forall t \in T$, $\forall j,k \in [1, N] \mid j \neq k \Rightarrow r_t[j] \neq r_t[k]$ where $r_t[j]$ is the j-th element in the vector r_t

The approach plans to consider a set of run computed for a specific topic and select the first top-k documents of each one: *depth of the pool.* Than it creates a set from the union of these documents and assigns a relevance judgment to each one choosing from a specific set. This set could be binary and in this case we have two possible values for the relevance: relevant and not relevant, or multigraded and in this case we could have many possible values that can be for example: not relevant, partially relevant, fairly relevant and highly relevant. After the collection building he finally wanted to execute the experiment analyzing the indexing methods properties. The collection for Cranfield 2 was composed of documents and queries that users would really use, in particular it was made of 1400 documents and 221 questions. With this experiment Cleverdon also introduce two measures for the evaluation that are the Precision and the Recall. The *Recall* measure is defined as:

$$\text{Recall} = \frac{D_{rr}}{D_{rc}}$$

where D_{rr} is the number of relevant retrieved documents while D_{rc} is the number of relevant documents in the collection also known as *Recall Base*. The term D_{rc} can be calculated as:

$$D_{rc} = D_{rr} + D_{rn}$$

where D_{rn} is the number of relevant but not retrieved documents. Recall defines how many relevant documents have been retrieved in response to a specific query compared to the totality of relevant documents in the collection. The **Precision** measure is defined as:

$$\text{Precision} = \frac{D_{rr}}{D_{rq}}$$

where D_{rq} is the number of documents, both relevant and not relevant, retrieved by the query and can be defined as:

$$D_{rq} = D_{rr} + D_{nr}$$

with D_{nr} indicating the number of non-relevant document that are been retrieved by the query. The Precision then indicates the number of relevant document retrieved in response to a specific query compare to the totality of retrieved documents.

The Cranfield paradigm therefore modeled a research task building a specific test collection that can be reuse in different experimental context enabling the reproduction of experiments and their comparison. In order to evaluate a retrieval system we have to:

- Select a set of documents that is relevant for the task.
- Select a set of topics that truly represents the information needs of the

users.

- Define the relevance judgments used to mark documents as relevant or not for a specific topic.

Regarding the evaluation measures, others have been developed after the Cranfield experiment such as: *E-measure*, *F-measure*, *prec@k*, *11pt-AP*, *AP*, *MAP*, *CG*, *DCG* and *nDCG*. We will give just a short overview of the most used ones.

The **Average Precision (AP)** is a measure that, compare to the previous one, gives more importance to those runs that put relevant document in top position (namely with a high rank), so it is called a *Top Heavy measure*. This is a good effect because often we want our system to give us back relevant documents as the first ones on the returned list. This measure is defined as:

given a topic $t \in T$, a recall base RB_t, $REL = \{\mathrm{nr},\mathrm{r}\}$ and a run r_t of length $N \in \mathbb{N}^+$ such that:

$$\forall i \in [1, N], \tilde{r_t} = \begin{cases} 0 & \text{if } \hat{r_t}[i] = \mathrm{nr} \\ 1 & \text{if } \hat{r_t}[i] = \mathrm{r} \end{cases}$$

$$\mathrm{AP} = \frac{1}{\mathrm{RB}_t} \sum_{k=1}^{N} \widetilde{r}_t[k] \frac{\sum_{h=1}^{k} \widetilde{r}_t[h]}{k}$$

where T is the set of queries concerning the same information needs, REL is a set of relevant judgments that, in this case, are binary (they can only assume two types of values: non relevant or relevant) and the run r_t is the ranked list of documents resulting from the retrieval process. We can see from the definition that the division by k allow us to weight relevant documents with higher rank more than those with lower rank.

The **Mean Average Precision (MAP)** is based on the AP with the difference that this measure allows us to make an evaluation of the run on all the topics of the collection and not just only on a single one. It is defined as:

$$\mathrm{MAP}(R) = \frac{\sum_{t \in T} \mathrm{AP}(r_t)}{|T|}$$

While these two measures are defined using binary relevance judgments, the DCG and nDCG are defined for multi-graded relevance judgments. In multi-graded relevance the weight associated to each document can take more than two values, for example they could be:

- Highly Relevant (HR) $\rightarrow 3$
- Fairly Relevant (FR) $\rightarrow 2$
- Partially Relevant (PR) $\rightarrow 1$
- Not Relevant (NR) $\rightarrow 0$

The **Discounted Cumulative Gain (DCG)** is a multi-graded top heavy measure that is based on the measure of *Discounted Gain (DG)* which use a *discounting function* to progressively reduce the weight of the documents going from those with high rank to those with low rank. The DG is defined as:

given a run $R(t)$ of length $N \in \mathbb{N}^+$ and a logarithmic base $b \in \mathbb{N}^+$, $\forall k \in [1, N]$:

$$\mathrm{dg}^b_{r_t}[k] = \begin{cases} \tilde{r_t}[k] & \text{if } k < b \\ \frac{\tilde{r_t}[k]}{\log_b k} & \text{otherwise} \end{cases}$$

Another advantage from the discounting function is that it allows to modify the DG depending on the type of task we want to evaluate; for example if we are dealing with a Web Search we would choose b=10 that is the standard number of pages returned in this kind of search while if we are dealing with a mobile phone we will use a smaller value of b because in the screen we could see lesser pages. Then the DCG is:

$$DCG[j] = \sum_{k=1}^{j} \mathrm{dg}^b_{r_t}[k]$$

We can observe that this measure can take each kind of value and it is not included between 0 and 1, so a process of normalization can be added, obtaining a new kind of measure called **Normalized Discounted Cumulative Gain (nDCG)**. The normalization process it is done using the notion of **Ideal Run**. This run is the one with the best rank of documents, formally: the ideal run $I(t) = i_t$ is the one that satisfies these constraints:

1. recall base: $\forall t \in T, |\{j \in [1, N] \mid \mathrm{GT}(t, i_t[j]) \succ \min(\mathrm{REL})\}| = \mathrm{RB}_t$

2. ranking: $\forall t \in T, \forall j, k \in [1, N] \mid j < k \Rightarrow \hat{i}_t[j] \succeq \hat{i}_t[k]$

Than we can define the nDCG like:

$$\text{nDCG}^b[j] = \frac{\sum_{k=1}^{j} \text{dg}^b_{r_t}[k]}{\sum_{k=1}^{j} \text{dg}^b_{\hat{i}_t}[k]}$$

Thanks to this normalization we can use this measure to compare different run on the same topic or to calculate the mean of nDCG values for the same run on different topics.

2.2 Entity Search

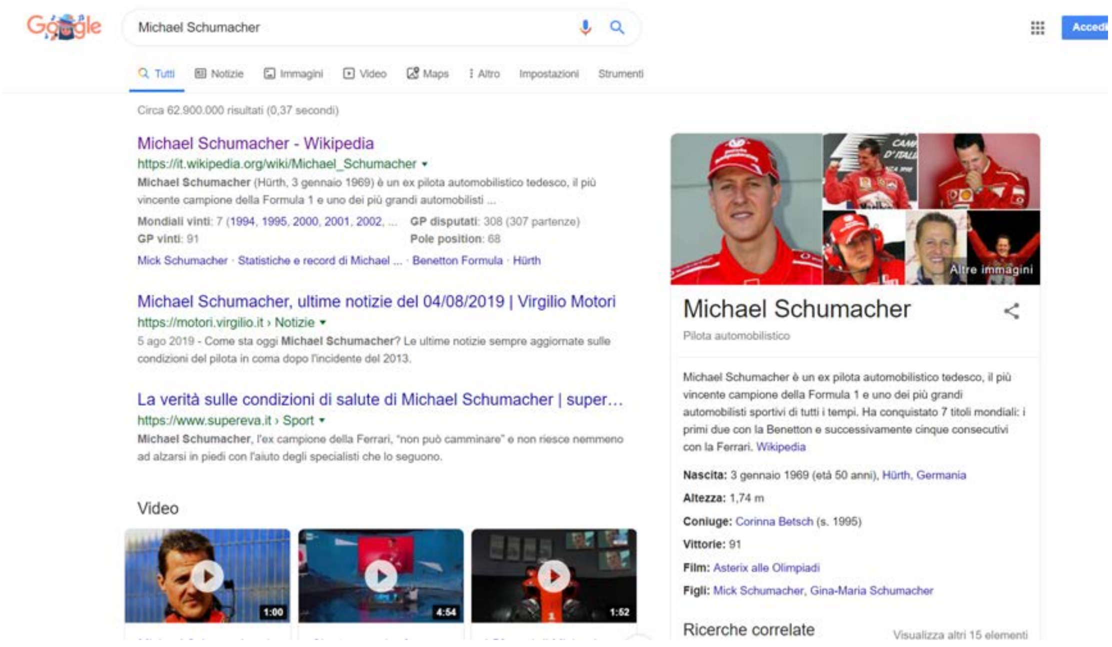
As defined by Balog in [3]: *"Entity-oriented search is the search paradigm of organizing and accessing information centered around entities, and their attributes and relationships"* and *"an entity is an object or concept in the real world that can be distinctly identified"*. In the last years the importance of entity search has become even greater due to the fact that 40% of the queries made by users mention specific entities [19, 27, 37]. Usually a user formulates a query in natural language and keywords searching for a specific thing, person or fact that can be logically gathered in the concept of entity, an example of this type of search is represented in Figure 2.3 where in the right side of the page it is possible to see the returned *Entity Card* associated to the entity searched *"Michael Schumacher"*. Entities are connected to each other by relationships and are described by means of their properties.

2.2.1 Entity

From a machine perspective an entity can be seen as [3]: *"a uniquely identifiable object or thing characterized by its name(s), type(s) and relationships to other entities"*. For information retrieval purposes we consider as entities only those present in a set called *entity catalog*. Giving a more technical definition, from [3]: *"An entity catalog is a collection of entries, where each entry is identified by a unique ID and contains the name of the corresponding entry"*.

We can identify two types of entity:

- **Named entities**: it is an object that can be defined by its proper noun.
- **Concepts:** it is an abstract object that represents a concept in a specific field such as physics, math, philosophy or others.


Michael Schumacher

Tutti · Notizie · Immagini · Video · Maps · Altro · Impostazioni · Strumenti

Accedi

Circa 62.900.000 risultati (0,37 secondi)

Michael Schumacher - Wikipedia
https://it.wikipedia.org/wiki/Michael_Schumacher ▾
Michael Schumacher (Hürth, 3 gennaio 1969) è un ex pilota automobilistico tedesco, il più vincente campione della Formula 1 e uno dei più grandi automobilisti ...
Mondiali vinti: 7 (1994, 1995, 2000, 2001, 2002, ... **GP disputati**: 308 (307 partenze)
GP vinti: 91 **Pole position**: 68
Mick Schumacher · Statistiche e record di Michael ... · Benetton Formula · Hürth

Michael Schumacher, ultime notizie del 04/08/2019 | Virgilio Motori
https://motori.virgilio.it › Notizie ▾
5 ago 2019 - Come sta oggi **Michael Schumacher**? Le ultime notizie sempre aggiornate sulle condizioni del pilota in coma dopo l'incidente del 2013.

La verità sulle condizioni di salute di Michael Schumacher | super...
https://www.supereva.it › Sport ▾
Michael Schumacher, l'ex campione della Ferrari, "non può camminare" e non riesce nemmeno ad alzarsi in piedi con l'aiuto degli specialisti che lo seguono.

Video

1:00 4:54 1:52

Altre immagini

Michael Schumacher
Pilota automobilistico

Michael Schumacher è un ex pilota automobilistico tedesco, il più vincente campione della Formula 1 e uno dei più grandi automobilisti sportivi di tutti i tempi. Ha conquistato 7 titoli mondiali: i primi due con la Benetton e successivamente cinque consecutivi con la Ferrari. Wikipedia

Nascita: 3 gennaio 1969 (età 50 anni), Hürth, Germania
Altezza: 1,74 m
Coniuge: Corinna Betsch (s. 1995)
Vittorie: 91
Film: Asterix alle Olimpiadi
Figli: Mick Schumacher, Gina-Maria Schumacher

Ricerche correlate Visualizza altri 15 elementi

Figure 2.3: Illustration of an entity search in Google. It is possible to see the classic sites result and on the right the entity card associated to the entity searched: Michael Schumacher.

Usually an entity is characterized by a set of properties that specifies some of its aspects. These properties are:

- **Entity ID**: that uniquely identifies the entity which we are referring to.
- **Name(s)**: common name in the real world associated to that entity. It does not uniquely identified the entity, in fact in the same catalog there can be multiple different entities with the same name. These names are called *surface forms* or *aliases*. The presence of this kind of situations can became a problem for the machine, which cannot easily resolve the ambiguity.
- **Type(s)**: category to which the entity belong. These categories can be considered as a set grouping together all the entities that has similar properties. An example of type can be "doctor".
- **Attributes**: attributes related to the entity that give a more specific and complete vision of the object we are considering.
- **Relationships**: connections between our entity and others. These are usually represented by a verb that specify the type of connection that exists between the two considered entities.

2.2.1.1 Entity property representation

All the information about entities has to be represented somehow in order to take all the advantages this knowledge can brought to our search system. This information can be represented in a semi-structured or structured form, in particular we have two type of representation: *knowledge repository* and *knowledge base.*

From [3] a **Knowledge repository (KR)**: *"is a catalog of entities that contains entity type information, and (optionally) descriptions or properties of entities, in a semi-structured or structured format"*. In order to maintain this information in a structured way a knowledge representation model is used, this model is called Resource Description Framework (RDF) [3]. This model allow us to describe entities with their properties and in particular relationships thorough triples. Each triple is represented by a subject (the first entity), a predicate (the relationship) and an object (the second entity). We will discuss RDF more in details in Section 2.2.2. The RDF triples represent some facts or assertions and in particular all the triples involving an entity represents the entity itself. All the assertions can be organized in a structure called *Knowledge base.*

From [3] a **Knowledge Base (KB)**: *"is a structured knowledge repository that contains a set of facts (assertions) about entities"*. Thanks to this definition and the Figure 2.4 we can observe that a KB is a KR but the reverse is not true. The representation of the entities through this kind of structure allow us to see each entity like node and each relationship between entities like edges of a graph that takes the name of **Knowledge Graph (KG)**. Another structure involved in entities representation, connection and description is the *Ontology.* This may seem similar to the KB so we will give an explanation of the difference between the two of them. While the KB aims to give a detailed description of all the entities in our dataset (bottom-up approach), the Ontology gives a representation of the world of interest, in particular focusing on concepts and relationship between instances (top-down approach) [3]. From this derives that the instance level, the one on which our objects rely, may not be involved in ontology, while it is fully described in the KB. Ontologies are based on blocks that are: classes, objects, relations, attributes, restrictions on relations and rules/axioms [3].

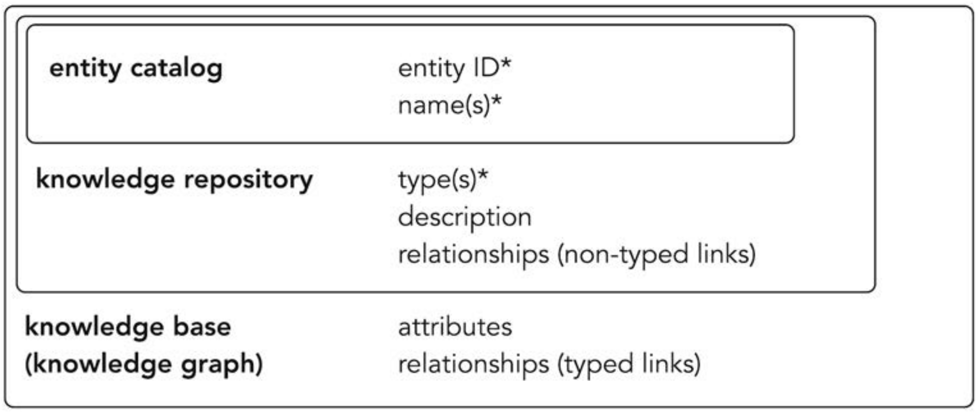
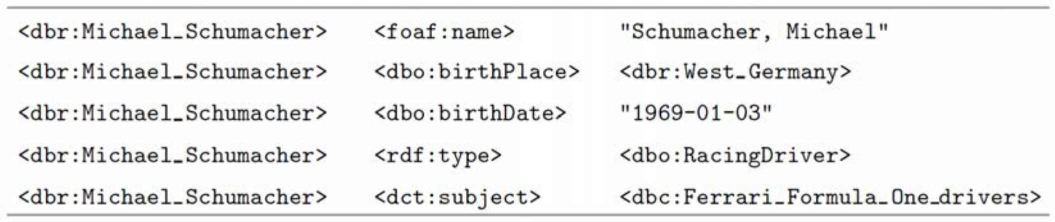


Figure 2.4: Relationship between knowledge repository and knowledge base taken from [3]. The entity properties marked with * are mandatory.



<dbr:Michael_Schumacher>	<foaf:name>	"Schumacher, Michael"
<dbr:Michael_Schumacher>	<dbo:birthPlace>	<dbr:West_Germany>
<dbr:Michael_Schumacher>	<dbo:birthDate>	"1969-01-03"
<dbr:Michael_Schumacher>	<rdf:type>	<dbo:RacingDriver>
<dbr:Michael_Schumacher>	<dct:subject>	<dbc:Ferrari_Formula_One_drivers>

Figure 2.5: Examples of RDF triples taken from [3]. In the leftmost column the subjects, in the middle column the predicates, in the rightmost column the objects.

2.2.2 Resource Description Framework

In relational model there is a clear separation between data and metadata but sometimes it would be useful to merge these two components maintaining a format that results machine readable and flexible [46]. The RDF language was created to accomplish with these requests. It is a representational language based on directed labeled graph. It allows us to represent resources that could be entities (objects), entity types (class) or relations [3]. Each entity is identified through an Uniform Resource Identifier (URI) and it is contained in one or more triples in the form: *Subject - Predicate - Object*. The subject is the resource, the predicate is the relationship or the property while the object is another resource (URI) or a literal as shown in Figure 2.5. The triples represents the factual statements of the Knowledge Base and, like mentioned before, this base could be seen as a large, directed, labeled graph called *RDF Graph* as shown in Figure 2.6. In this case each subject/object is represented as a node while each predicate is represented as an edge. In RDF also the

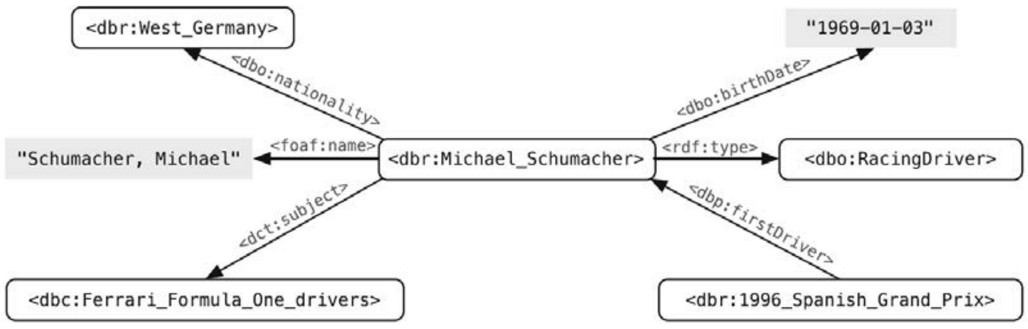


Figure 2.6: Excerpt of the RDF graph related to the entity Michael Schumacher taken from [3]. URIs are represented by rounded rectangles, literals by shaded rectangles.

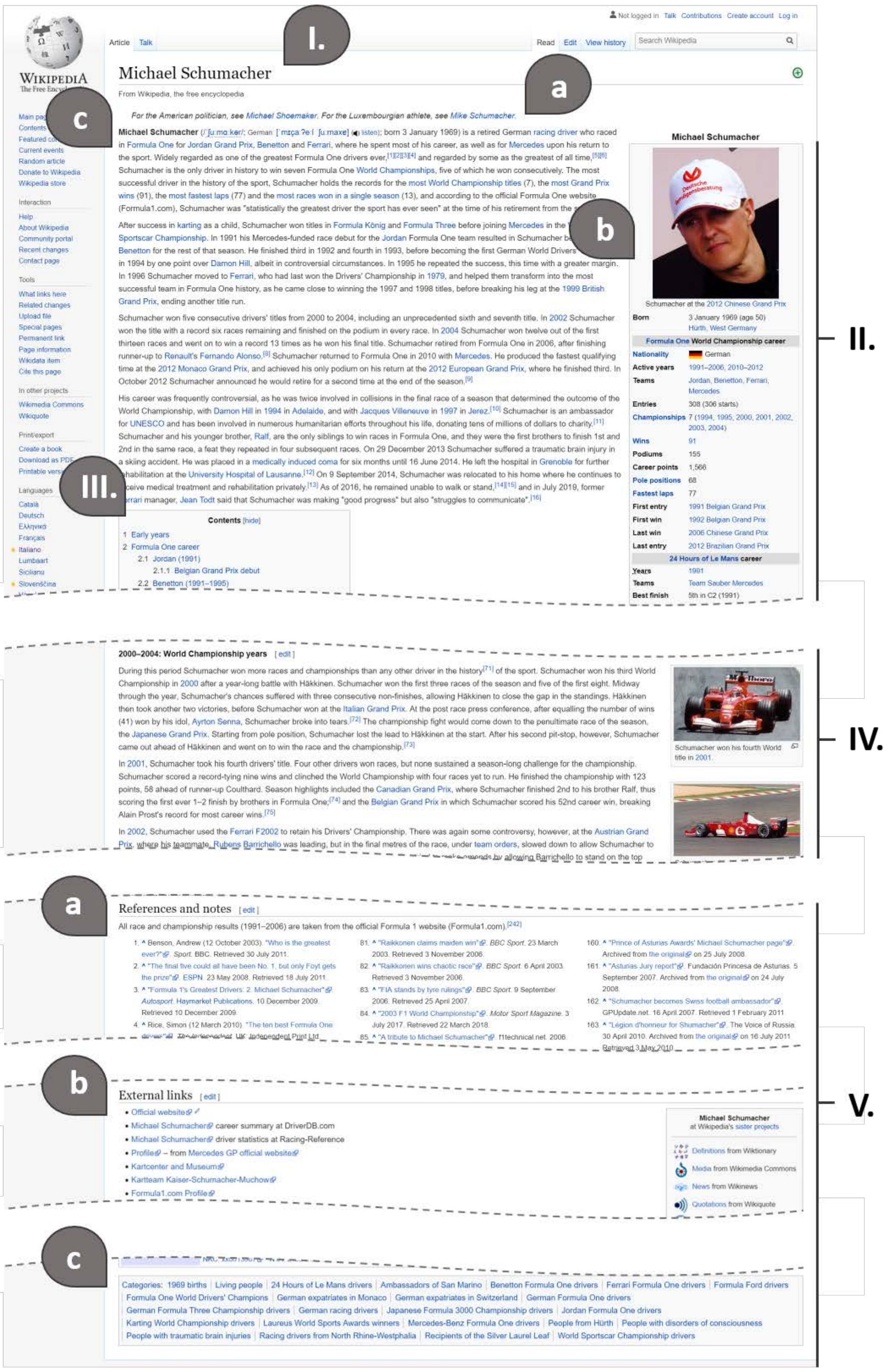
statements themselves are resources so graph edges can connect not only nodes but also edges, generating an hyper-graph. The RDF base, according to [46], can be formally described as a tuple $< I, P >$, where $I \subseteq U$ is a set of individuals in the infinite set of resources U, $P \subseteq I \times U \times I$ is a set of properties and $I \cap \{\text{URI}(p) \mid p \in P\} = \emptyset$. In order to control, handle, describe the RDF structure the W3C introduced the RDF Schema (RDFS). This provides a vocabulary of constructs that allows the classification and definition of all the information related to RDF data. In particular it defines classes for different instances, constraints for resources properties, differentiation between property and non-property resources, etc. [46].

2.2.3 DBpedia

DBpedia is a database version of Wikipedia; it is the results of a continuous process, supported by a community of people, that aims to retrieve all the information contained in Wikipedia pages storing them in RDF format. This database has some interesting feature such as the fact that new versions are periodically released and that it is available in many different languages. In order to describe how DBpedia is created and which is its structure we have to describe firstly Wikipedia pages parts and content. A *Wikipedia page* is composed mainly by five different parts [3] as shown in Figure 2.7:

I. Title: this is the title of the page and it represents also the name of the corresponding DBpedia entity in the database. In particular the triples generated are characterized by the prefix *dbr* reported in Figure 2.8.

II. Lead section: this is the top part of the page which contains in turn:



Michael Schumacher

From Wikipedia, the free encyclopedia

For the American politician, see Michael Shoemaker. For the Luxembourgian athlete, see Mike Schumacher.

Michael Schumacher (/ˈʃuːmɑːkər/; German: [ˈmɪçaːʔeːl ˈʃuːmaxɐ] listen); born 3 January 1969) is a retired German racing driver who raced in Formula One for Jordan Grand Prix, Benetton and Ferrari, where he spent most of his career, as well as for Mercedes upon his return to the sport. Widely regarded as one of the greatest Formula One drivers ever,[1][2][3][4] and regarded by some as the greatest of all time,[5][6] Schumacher is the only driver in history to win seven Formula One World Championships, five of which he won consecutively. The most successful driver in the history of the sport, Schumacher holds the records for the most World Championship titles (7), the most Grand Prix wins (91), the most fastest laps (77) and the most races won in a single season (13), and according to the official Formula One website (Formula1.com), Schumacher was "statistically the greatest driver the sport has ever seen" at the time of his retirement from the s

After success in karting as a child, Schumacher won titles in Formula König and Formula Three before joining Mercedes in the W Sportscar Championship. In 1991 his Mercedes-funded race debut for the Jordan Formula One team resulted in Schumacher be Benetton for the rest of that season. He finished third in 1992 and fourth in 1993, before becoming the first German World Drivers in 1994 by one point over Damon Hill, albeit in controversial circumstances. In 1995 he repeated the success, this time with a greater margin. In 1996 Schumacher moved to Ferrari, who had last won the Drivers' Championship in 1979, and helped them transform into the most successful team in Formula One history, as he came close to winning the 1997 and 1998 titles, before breaking his leg at the 1999 British Grand Prix, ending another title run.

Schumacher won five consecutive drivers' titles from 2000 to 2004, including an unprecedented sixth and seventh title. In 2002 Schumacher won the title with a record six races remaining and finished on the podium in every race. In 2004 Schumacher won twelve out of the first thirteen races and went on to win a record 13 times as he won his final title. Schumacher retired from Formula One in 2006, after finishing runner-up to Renault's Fernando Alonso.[8] Schumacher returned to Formula One in 2010 with Mercedes. He produced the fastest qualifying time at the 2012 Monaco Grand Prix, and achieved his only podium on his return at the 2012 European Grand Prix, where he finished third. In October 2012 Schumacher announced he would retire for a second time at the end of the season.[9]

His career was frequently controversial, as he was twice involved in collisions in the final race of a season that determined the outcome of the World Championship, with Damon Hill in 1994 in Adelaide, and with Jacques Villeneuve in 1997 in Jerez.[10] Schumacher is an ambassador for UNESCO and has been involved in numerous humanitarian efforts throughout his life, donating tens of millions of dollars to charity.[11] Schumacher and his younger brother, Ralf, are the only siblings to win races in Formula One, and they were the first brothers to finish 1st and 2nd in the same race, a feat they repeated in four subsequent races. On 29 December 2013 Schumacher suffered a traumatic brain injury in a skiing accident. He was placed in a medically induced coma for six months until 16 June 2014. He left the hospital in Grenoble for further rehabilitation at the University Hospital of Lausanne.[12] On 9 September 2014, Schumacher was relocated to his home where he continues to receive medical treatment and rehabilitation privately.[13] As of 2016, he remained unable to walk or stand,[14][15] and in July 2019, former Ferrari manager, Jean Todt said that Schumacher was making "good progress" but also "struggles to communicate".[16]

Contents [hide]

1 Early years
2 Formula One career
2.1 Jordan (1991)
2.1.1 Belgian Grand Prix debut
2.2 Benetton (1991–1995)

Michael Schumacher

Schumacher at the 2012 Chinese Grand Prix

Born	3 January 1969 (age 50) Hürth, West Germany
Formula One World Championship career	
Nationality	German
Active years	1991–2006, 2010–2012
Teams	Jordan, Benetton, Ferrari, Mercedes
Entries	308 (306 starts)
Championships	7 (1994, 1995, 2000, 2001, 2002, 2003, 2004)
Wins	91
Podiums	155
Career points	1,566
Pole positions	68
Fastest laps	77
First entry	1991 Belgian Grand Prix
First win	1992 Belgian Grand Prix
Last win	2006 Chinese Grand Prix
Last entry	2012 Brazilian Grand Prix
24 Hours of Le Mans career	
Years	1991
Teams	Team Sauber Mercedes
Best finish	5th in C2 (1991)

2000–2004: World Championship years [edit]

During this period Schumacher won more races and championships than any other driver in the history[71] of the sport. Schumacher won his third World Championship in 2000 after a year-long battle with Häkkinen. Schumacher won the first three races of the season and five of the first eight. Midway through the year, Schumacher's chances suffered with three consecutive non-finishes, allowing Häkkinen to close the gap in the standings. Häkkinen then took another two victories, before Schumacher won at the Italian Grand Prix. At the post race press conference, after equalling the number of wins (41) won by his idol, Ayrton Senna, Schumacher broke into tears.[72] The championship fight would come down to the penultimate race of the season, the Japanese Grand Prix. Starting from pole position, Schumacher lost the lead to Häkkinen at the start. After his second pit-stop, however, Schumacher came out ahead of Häkkinen and went on to win the race and the championship.[73]

Schumacher won his fourth World title in 2001.

In 2001, Schumacher took his fourth drivers' title. Four other drivers won races, but none sustained a season-long challenge for the championship. Schumacher scored a record-tying nine wins and clinched the World Championship with four races yet to run. He finished the championship with 123 points, 58 ahead of runner-up Coulthard. Season highlights included the Canadian Grand Prix, where Schumacher finished 2nd to his brother Ralf, thus scoring the first ever 1–2 finish by brothers in Formula One;[74] and the Belgian Grand Prix in which Schumacher scored his 52nd career win, breaking Alain Prost's record for most career wins.[75]

In 2002, Schumacher used the Ferrari F2002 to retain his Drivers' Championship. There was again some controversy, however, at the Austrian Grand Prix, where his teammate, Rubens Barrichello was leading, but in the final metres of the race, under team orders, slowed down to allow Schumacher to ... make amends by allowing Barrichello to stand on the top

References and notes [edit]

All race and championship results (1991–2006) are taken from the official Formula 1 website (Formula1.com).[242]

1. ^ Benson, Andrew (12 October 2003). "Who is the greatest ever?". *Sport*. BBC. Retrieved 30 July 2011.
2. ^ "The final five could all have been No. 1, but only Foyt gets the prize". ESPN. 23 May 2008. Retrieved 18 July 2011.
3. ^ "Formula 1's Greatest Drivers: 2. Michael Schumacher". *Autosport*. Haymarket Publications. 10 December 2009. Retrieved 10 December 2009.
4. ^ Rice, Simon (12 March 2010). "The ten best Formula One driver". *The Independent*. UK: Independent Print Ltd.

81. ^ "Raikkonen claims maiden win". *BBC Sport*. 23 March 2003. Retrieved 3 November 2006.
82. ^ "Raikkonen wins chaotic race". *BBC Sport*. 6 April 2003. Retrieved 3 November 2006.
83. ^ "FIA stands by tyre rulings". *BBC Sport*. 9 September 2006. Retrieved 25 April 2007.
84. ^ "2003 F1 World Championship". *Motor Sport Magazine*. 3 July 2017. Retrieved 22 March 2018.
85. ^ "A tribute to Michael Schumacher". f1technical.net. 2006.

160. ^ "Prince of Asturias Awards' Michael Schumacher page". Archived from the original on 25 July 2008.
161. ^ "Asturias Jury report". Fundación Princesa de Asturias. 5 September 2007. Archived from the original on 24 July 2008.
162. ^ "Schumacher becomes Swiss football ambassador". GPUpdate.net. 16 April 2007. Retrieved 1 February 2011.
163. ^ "Légion d'honneur for Shumacher". The Voice of Russia. 30 April 2010. Archived from the original on 16 July 2011. Retrieved 3 May 2010.

External links [edit]

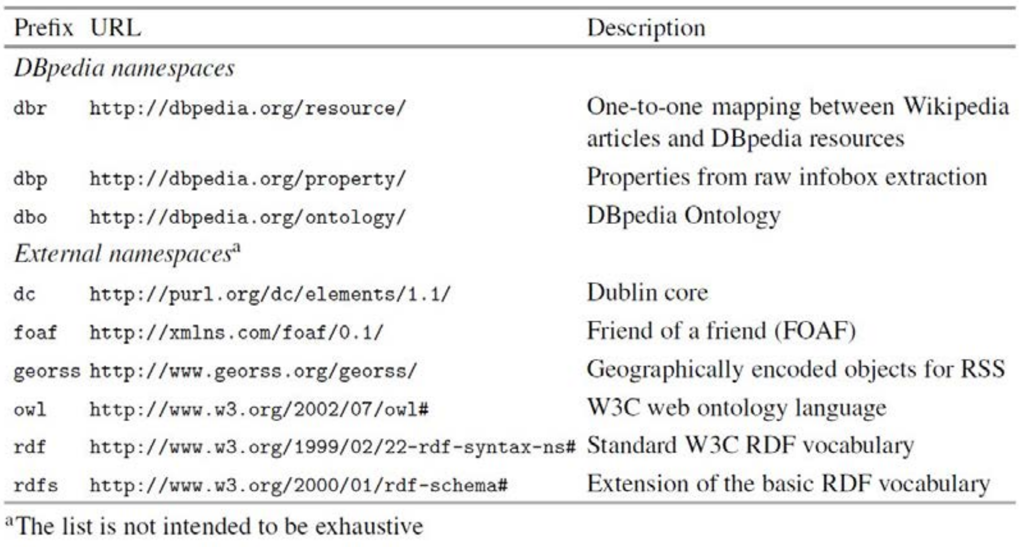
- Official website
- Michael Schumacher career summary at DriverDB.com
- Michael Schumacher driver statistics at Racing-Reference
- Profile – from Mercedes GP official website
- Kartcenter and Museum
- Kartteam Kaiser-Schumacher-Muchow
- Formula1.com Profile

Michael Schumacher at Wikipedia's sister projects

Definitions from Wiktionary
Media from Wikimedia Commons
News from Wikinews
Quotations from Wikiquote

Categories: 1969 births | Living people | 24 Hours of Le Mans drivers | Ambassadors of San Marino | Benetton Formula One drivers | Ferrari Formula One drivers | Formula Ford drivers | Formula One World Drivers' Champions | German expatriates in Monaco | German expatriates in Switzerland | German Formula One drivers | German Formula Three Championship drivers | German racing drivers | Japanese Formula 3000 Championship drivers | Jordan Formula One drivers | Karting World Championship drivers | Laureus World Sports Awards winners | Mercedes-Benz Formula One drivers | People from Hürth | People with disorders of consciousness | People with traumatic brain injuries | Racing drivers from North Rhine-Westphalia | Recipients of the Silver Laurel Leaf | World Sportscar Championship drivers

Figure 2.7: Wikipedia page structure taken from [3].



Prefix	URL	Description
DBpedia namespaces		
`dbr`	`http://dbpedia.org/resource/`	One-to-one mapping between Wikipedia articles and DBpedia resources
`dbp`	`http://dbpedia.org/property/`	Properties from raw infobox extraction
`dbo`	`http://dbpedia.org/ontology/`	DBpedia Ontology
External namespaces[a]		
`dc`	`http://purl.org/dc/elements/1.1/`	Dublin core
`foaf`	`http://xmlns.com/foaf/0.1/`	Friend of a friend (FOAF)
`georss`	`http://www.georss.org/georss/`	Geographically encoded objects for RSS
`owl`	`http://www.w3.org/2002/07/owl#`	W3C web ontology language
`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	Standard W3C RDF vocabulary
`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`	Extension of the basic RDF vocabulary

[a]The list is not intended to be exhaustive

Figure 2.8: URI namespaces mostly used in DBpedia taken from [3].

(a) Disambiguation links: a link that refers to a disambiguation page that the user can consult in case the entity currently displayed is not the one he/she wanted;

(b) Infobox: a table containing a summary of the main information related to the instance that is described in the page. It is an important part of the page because it provides us most of the information regarding the relationship between entities;

(c) Introductory text: a first brief introduction to the page that gives some general information.

III. Table of contents: this is an index of the content of the section of the page.

IV. Body content: this is the main part of the page containing the real information about the instance described.

V. Appendices and bottom matter: this is the lower part containing the links to references (a), to other external links (b) and to all the categories to which the instance of the page is associated (c). These category links give the user the possibility to navigate through other similar or related pages.

For DBpedia creation some extractors have been created. They can be categorized into four types [3, 24]:

- Raw infobox extraction: in infobox all the entity information is represented as property-value pairs and these pairs are translated into DBpedia predicates. In particular the triples generated are characterized by the prefix *dbp* as shown in Figure 2.8.
- Mapping-based infobox extraction: this type of extraction wants to overcome the inconsistency problem of infoboxes. In Wikipedia pages these tables have often different templates one from the others and it could happen that they use different terms to express the same type of concept. The mapping-based extraction tries to standardize the predicates generated to enforce consistency. The triples generated are characterized by the prefix *dbo* as shown in Figure 2.8.
- Feature extraction: this is a set of extractors, each one specialized in extracting a specific single feature from the Wikipedia page like: homepage, external links, abstracts, categories, etc.
- Statistical extraction: these are a group of extractors that uses statistical tools to get information useful for computational linguistics tasks.

Being DBpedia based in RDF format, it has its manually created ontology based on Wikipedia infoboxes. It is organized in a hierarchy of classes each one described by a set of properties that can assume some specific already defined values, an excerpt from the DBpedia ontology is represented in Figure 2.9.

2.2.4 Evaluation

When we talk about Evaluation in Information Retrieval we commonly refer to the Cranfield paradigm explained in Section 2.1.1.3. This approach is the de-facto standard paradigm for evaluation. As said by Blanco et al. in [9]: *"Cranfield methodology measures retrieval effectiveness by the means of a fixed documents collection, a set of queries, a set of relevance assessments denoting which documents are (not) relevant for a query, and a number of well-understood and defined metrics, such as precision and recall"*. This system is also used for evaluation in entity search task, indeed the retrieval process remains the same as in textual information retrieval, with the difference that in this case the documents used are created in such a way as to contain all the information related to the entities.

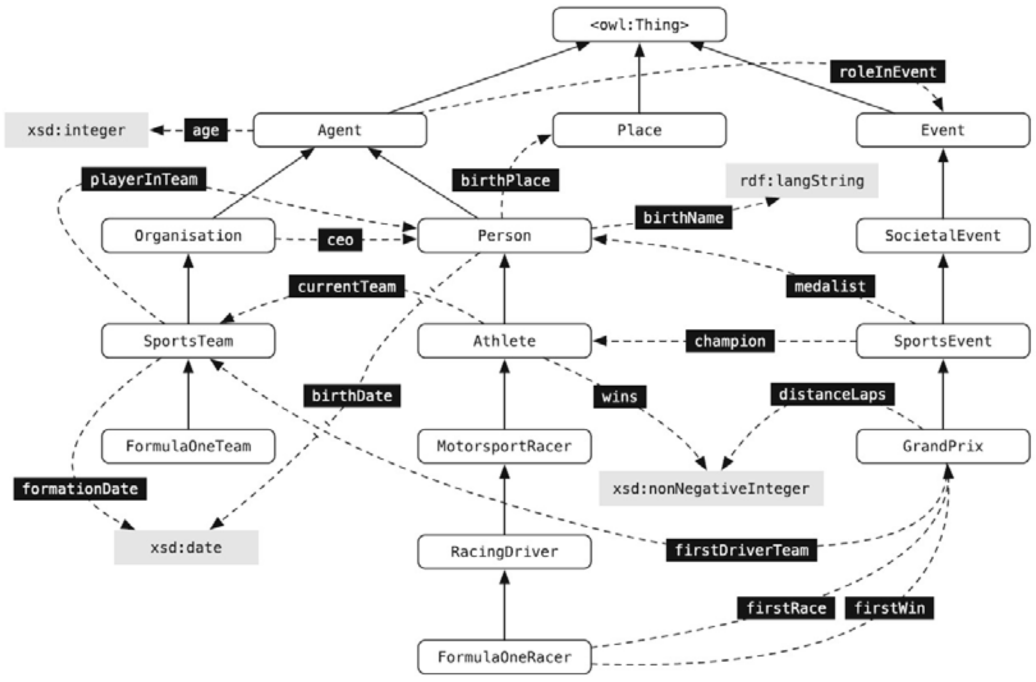


Figure 2.9: Excerpt from DBpedia ontology taken from [3]. Classes are represented by rounded rectangles where arrows with solid lines indicate subclass relationships (from subclass to superclass). Properties are denoted by the dashed arrows with black labels. Value types are shown in gray boxes. Unless specifically indicated, classes/values are in the dbo namespace, see Figure 2.8.

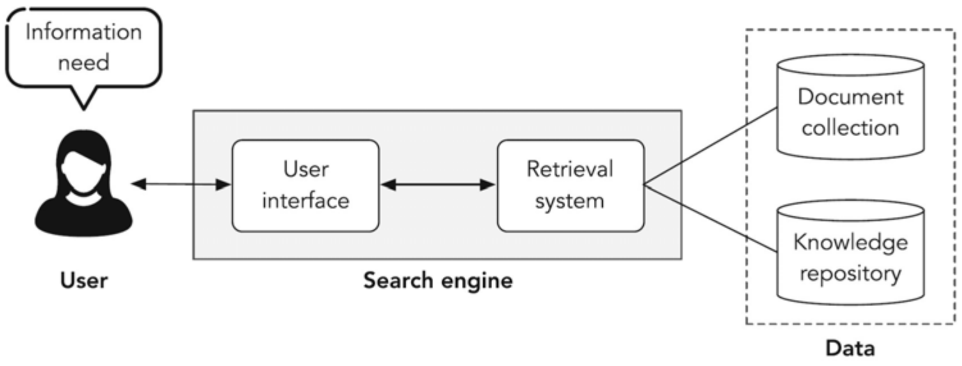


Figure 2.10: Structure of entity-search process taken from [3].

2.2.4.1 Tasks

In this thesis we focus on the task of **ad-hoc entity retrieval** which aims to answer information needs relayed to a particular entity expressed in unconstrained natural language and resolved using a collection of structured data [32]. In order to do this the retrieval systems has to identify the entities described by the natural language query of the user, identify the entities in the documents of the collection and find a match between these two elements. The structure of this process and the elements involved are represented in Figure 2.10. During the execution of this process three main challenges are encountered [4]:

1. Information need representation: the user query is formulated in natural language so the system has only an implicit description of the entities it has to find in its collection of documents.

2. Entity representation: often in entity search we have two different type of data: structured and unstructured. The unstructured part is composed of the content written as text in the documents, while the structured part is that contained in the knowledge base and that describe relationships between entities and their properties. The challenge is to find the best way to integrate this two different type of information.

3. Matching: once we have query and documents entities we have to match them, so we have to ensure that the two representations coincide or at least get closer in order to obtain an efficient retrieval.

Another problem that arises from the use of this new approach is the lack of human-readable description of relationship and properties of the entities retrieved [5]. Despite these difficulties, the introduction of entities in the information retrieval world it is very useful because it can enrich the traditional document retrieval with entity information or associations, it can help the user to express its information need and it can be use to provide context information and recommendation [3].

2.2.4.2 Dataset and Ground Truth

In order to examine how documents are generated for ad-hoc entity retrieval we first have to describe which are the most used datasets currently and how they are structured. In most cases researcher used the "Billion Triple Challenge 2009" (BTC) [21] dataset created for the Semantic Web Challenge [8] in 2009 that contains 1.14 billion RDF statements [32]. Here the data are encoded in NQuads format [21] which can be represented as:

<subject> <predicate> <object> <context>

Where:

- Subject: that represents the entity.
- Predicate: that represents the relationship between subject and object.
- Object: that represents a property or another entity. In case of a property we call this component *literal*.
- Context: that represents the URI of the graph that provides context information of other fields.

This concept is similar to the one described by RDF triples with the difference of the additional field: *context*. Also the DBpedia dataset described in Section 2.2.3 is becoming increasingly used in the research word.

Starting from these we create entity representation, as described in the next Section 2.2.4.3, and relevant assessments. Relevant assessments are generally constructed with a manual approach. In the case of the BTC dataset they are supplied together with the data, but in some cases they are generated from scratch as in [28, 9]. In [28] they are been made using a pooling strategy, in particular they take from the resulting runs the top 5 documents of each and merge together creating a set of different documents. Than they judge each document in the set giving an assessment that can take the assessment: relevant or not relevant.

It is interesting to see instead the approach used in [9]. Blanco et al. exploit the Amazon Mechanical Turk (MTurk) crowdsourcing marketplace [1]. This marketplace hires workers distributed allover the world to perform tasks, processes, jobs virtually commissioned by applicants. The researchers select 50 queries from the Yahoo! Search Tiny Sample v1.0 dataset that contains thousands of real queries from Yahoo's US query log provided by Yahoo! Webscope program [47]. Each query is randomly selected, it has been asked by at least three different users and it has been manually checked that at least 1 real answer for that query exists. At this point at each worker has been presented the query and a human-readable version of the RDF triples identifying an entity, without incorporating the explicit name of it, and he/she were asked to decide if the representation is irrelevant, somewhat relevant or relevant for that specific query. After collecting all the results researchers assigned an assessments for each document basing on the majority of the votes. It has also been incorporated some gold-win and gold-loose case, that was known a-priori from the authors if they are relevant or not, in order to measure the quality of the assessments. If a worker miss the majority of the gold-win case its assessments were rejected being deemed unreliable.

2.2.4.3 Entity Representations

While with the textual information retrieval we associate a portion of text to a document, with entity retrieval we have to define how this document is created and this usually depends on the original dataset we are using. There are several state-of-the-art methods applied for Entity Search task [6, 11, 32, 10]. The general approach plans to build documents, called *Pseudo Documents* or *Virtual Documents*, from the chosen data set which is usually in RDF format. From this dataset all the triples with the same subject are identified and concatenated forming the new document used for the indexing phase of the retrieval process. In this way the built document will represent a specific entity enclosing all the information correlated to it and we will have one document for each entity. In the process of document production many different strategies could be chosen such as the choice to insert just the triple with a specific predicate or a specific object.

Starting from the datasets presented in Section 2.2.4.2 we have now several ways to construct our documents, but in all cases we have to obtain a meaningful representation of the entity in order to achieve good performance in the retrieval phase. The most used methods can be divided into two main categories: Text-based approaches and Structured-based approaches. The Text-based approaches plan to create a document concatenating all the triples or NQuads belonging to the same entity, i.e. having the same subject, in a Bag-Of-Words approach.

The Structured-based approach instead create a fielded representation for each entity where different fields are associated to different predicates. This choice for fields could led to management problems when there are many different predicates, so sometimes these are grouped into bigger categories based on predicates types [34] or importance [39]. This reduction makes field weights more tractable even if it limits the semantic expressiveness of the document [32]. In both the approaches a filtering phase could be implemented before the document creation, deciding which information we want to insert in our representation. Lets see we some specific examples how these representations are generated:

- In [28] the authors created a profile for each entity and associated this profile with a document. In particular they used the text-based approach merging all the lines that has the same subject field, so the same URI of the entity.
- In [9] the authors report several methods used by different research groups. Most of them created documents using the text-based approach while one group used a structured-based approach based on the context of the entity. In this latter case documents are created using a labeled tree data model with the entity as the root and the attributes and values as the nodes on quads that has the same subject and context. This paper also integrates entity representations with information about their relationships obtained through the analysis of the total graph.
- In [10] Blanco and other researchers used a structured-based approach. They inserted in the same entity profile all the quads sharing the same subject and associated a field to each predicate assigning different weights to each one of them. If multiple values with same subject and predicate

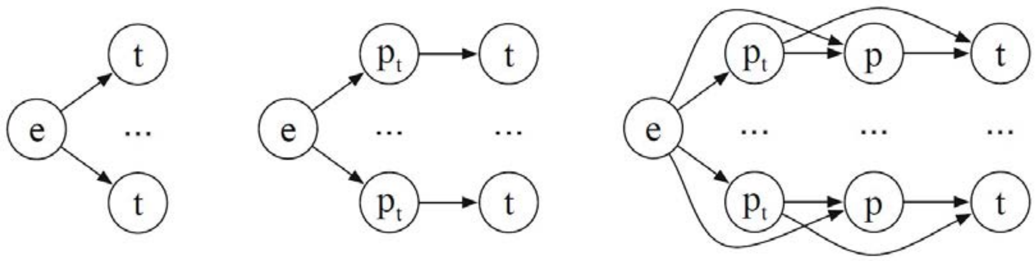


Figure 2.11: Graphical representations of entity models: (Left) Unstructured Entity Model, (Middle) Structured Entity Model using Predicate-folding, (Right) Hierarchical Entity Model [32].

were encountered they were simply concatenated in the same document. This allowed them to use a fielded version of the retrieval model BM25 called *BM25F* that weights query terms differently depending on the field considered [33].

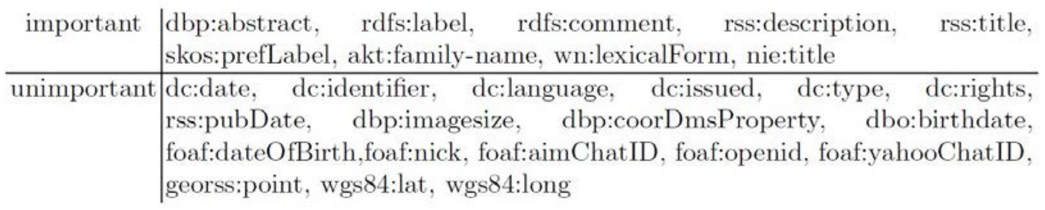
- Neumayer et al. in [32] tried to overcome the problem of loss of expressiveness which occurs with the use of categories in the structured-based approach considering a 2-level hierarchy entity representation. In particular they associated to the first level all the predicates types and to the the second level all the individual predicates of the type of the first level as shown in Figure 2.11.

2.2.5 Systems

In this section we present the state-of-the-art systems for the ad-hoc entity search task illustrating how they are implemented and which are their performances. In particular, we focus on three papers that are [10, 32, 6]. When describing these works we consider five fundamental features that are: the dataset, the document creation process, the retrieval model, the queries and the evaluation measures.

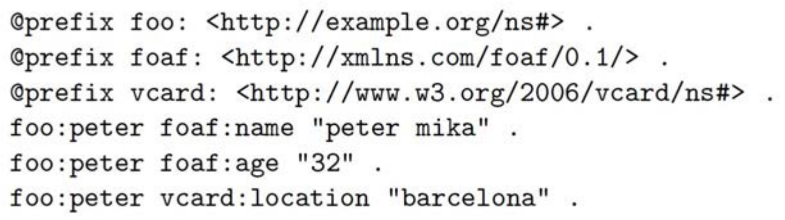
2.2.5.1 Effective and Efficient Entity Search in RDF Data

In [10] the authors presents a new method for entity search over RDF data. This method is based on a variant of the BM25F ranking function and exploit the advantages given by a set of new indexes structures for the retrieval and ranking of documents.



important	dbp:abstract, rdfs:label, rdfs:comment, rss:description, rss:title, skos:prefLabel, akt:family-name, wn:lexicalForm, nie:title
unimportant	dc:date, dc:identifier, dc:language, dc:issued, dc:type, dc:rights, rss:pubDate, dbp:imagesize, dbp:coorDmsProperty, dbo:birthdate, foaf:dateOfBirth,foaf:nick, foaf:aimChatID, foaf:openid, foaf:yahooChatID, georss:point, wgs84:lat, wgs84:long

Figure 2.12: Manually selected list of important and unimportant properties. URIs are abbreviated using known prefixes provided by the prefix.cc web service [10].



```
@prefix foo: <http://example.org/ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
foo:peter foaf:name "peter mika" .
foo:peter foaf:age "32" .
foo:peter vcard:location "barcelona" .
```

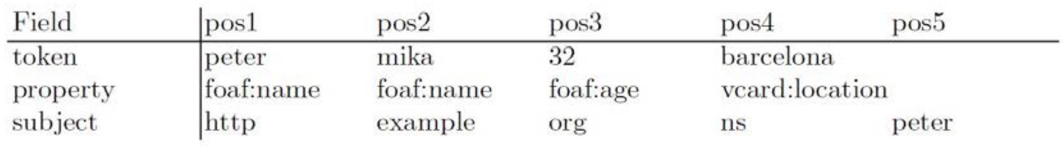
Figure 2.13: Representation of a sample of data in RDF format [10].

Dataset: as dataset they used the Billion Triple Challenge 2009 dataset described in Section 2.2.4.2.

Document creation: they considered all the datatype-properties sharing the same subject URI as a virtual document. As datatype-properties they meant all the quads having literals in the object position. If multiple values for the same subject and predicate existed they were merged together in the document. This document was then divided in fields considering each unique predicate as a separate field. This approach, however, led to an excessive number of parameters in practice so the authors decided to manually classify the properties into three categories: important, neutral and unimportant as illustrated in Figure 2.12.

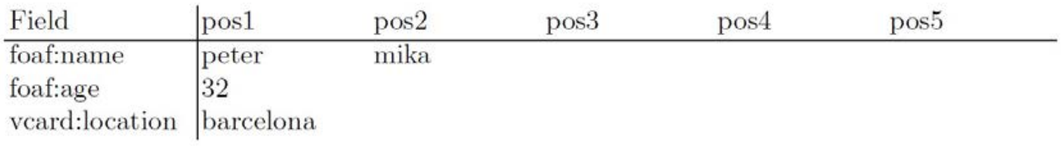
Retrieval: in the indexing phase they removed stop-words using a list of 389 common English terms and replaced delimiters with white spaces. They used three different index structures: a horizontal index (see Figure 2.14), a vertical index (see Figure 2.15) and a r-vertical index (see Figure 2.16).

The first one simply represented RDF triples; there are three fields: one for the subject, one for the predicate and one for the object or token. The second one used a field for each property occurring in the data. For ranking proposal we want to distinguish fields through their weights so the third index uses as fields the weights values: important, neutral, unimportant. In this case then they used three fields, one for each relevance weight. Regarding the



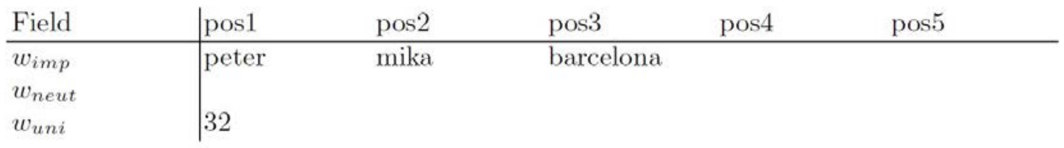
Field	pos1	pos2	pos3	pos4	pos5
token	peter	mika	32	barcelona	
property	foaf:name	foaf:name	foaf:age	vcard:location	
subject	http	example	org	ns	peter

Figure 2.14: Representation of the horizontal index of the data in Figure 2.13. On the left column the three fields corresponding to the three components of RDf triples [10].



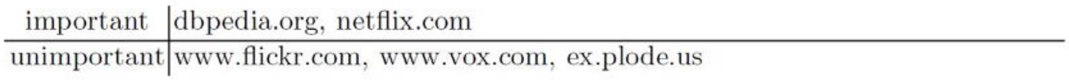
Field	pos1	pos2	pos3	pos4	pos5
foaf:name	peter	mika			
foaf:age	32				
vcard:location	barcelona				

Figure 2.15: Representation of the vertical index of the data in Figure 2.13. On the left column the fields corresponding to the various types of properties of RDf triples [10].



Field	pos1	pos2	pos3	pos4	pos5
w_{imp}	peter	mika	barcelona		
w_{neut}					
w_{uni}	32				

Figure 2.16: Representation of a r-vertical index of the data in Figure 2.13. On the left column the three fields corresponding to the three assessment values: unimportant, neutral and important [10].



important	dbpedia.org, netflix.com
unimportant	www.flickr.com, www.vox.com, ex.plode.us

Figure 2.17: Manually selected list of important and unimportant domains [10].

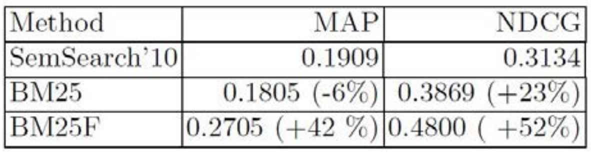
retrieval model Blanco et al. used a modified version of the classic BM25F [39]. While the standard BM25F requires information of all fields lengths for its calculation, the proposed version simplifies this aspect using as the length of all fields the size of the document; moreover due to the fact that usually the normalization component promotes short objects in RDF collections they mitigated the problem selecting a threshold beyond which all documents assume the same max length of 10. Finally they taken into account the class to which each document belong (unimportant, neutral, important) considering the document weight into the computation of the final retrieval score as:

$$\text{score}(Q, D) = \omega_D \cdot \text{score}^{\text{BM25F}}(Q, D)$$

where ω_D is the document weight, $\text{score}^{\text{BM25F}}$ is the score returned by the BM25F model depending on the query Q and the document D. In practice the weight was not assigned to each individual document but the authors manually classify a small number of domains into the tree before mentioned classes as shown in Figure 2.17 where all other domains are considered neutral. Moreover they assigned a separate weight to the *subject* because it has a special role as the identifier of the resource. Concluding they executed the following pipeline:

- Two rounds parameter tuning: the first round choosing a default configuration and tuning the performance of each one of the features individually. The second round, given an ordered list of the parameters they checked the model performance adding a new parameter at a time. This allowed them to determine the best configuration.

- 2-fold cross validation: they split the query set into two halves both used for tuning parameters. The first one through a linear search and the second one through the promising directions algorithm [39]. This algorithm plans to execute a linear search over each parameter starting with an initial set of parameter values, then it computes the



Method	MAP	NDCG
SemSearch'10	0.1909	0.3134
BM25	0.1805 (-6%)	0.3869 (+23%)
BM25F	0.2705 (+42 %)	0.4800 (+52%)

Figure 2.18: Cross-validated results comparing Blanco et al. ranking function against the BM25 baseline and the best performing submission at SemSearch 2010 (percentage improvements are relative to SemSearch 2010) [10].

promising direction identifying the vector that goes from the initial set of parameters to the best found values. The algorithm finally explores the parameter space over this vector and repeat this process until convergence to a local minimum or when a maximum number of iteration is executed.

The **query** set: consist of 92 queries selected from the query logs of Microsoft Live Search and Yahoo! Search (see [20] for details). To this set a spell corrector was used in order to fix any user error.
As **evaluation measures**: they used MAP and nDCG. They compare their results with standard BM25 and the best performing submission at SemSearch 2010 [20], checking for statistical significant differences against the baseline using Wilcoxons signed rank test (significance level set to 0.01), obtaining the ones represented in Figure 2.18.

2.2.5.2 On the Modeling of Entities for Ad-Hoc Entity Search in the Web of Data

In [32] the authors developed a new way of representing entities considering predicates grouped into two level of hierarchy in order to exploit both information regarding predicates types and individual predicates. They then compared this new hierarchical entity model with the standard one: unstructured and structured.
Dataset: as dataset they used the Billion Triple Challenge 2009 dataset described in Section 2.2.4.2.
Document creation: they considered the approach based on two-level hierarchical model described in Section 2.2.4.3 and executed a preprocessing on entity triples. This preprocessing consist of replacing each object containing an entity URI within the collection with the name of the entity itself, otherwise

it just replaces it with terms extracted from the relative part of the URI.
Retrieval: they executed the retrieval first on the baseline methods and than compared it with the hierarchical one. For the unstructured baseline method they considered three different set of experiments: "ALL", "field type-only" and "ALL-but-field type". In the first one they considered all four predicate types: Name, Attributes, OutRelations and InRealtions. In the second they used only a single predicate at a time and in the last one they omitted one of the types in turn. For the structured baseline method they considered three different set of experiments. In the first they assigned equal weights on all types, in the second one they assigned most of the weights to Name and Attributes types in equal proportion, in the last one they assigned most of the weights to Name and Attributes types taking into account also relations. Note that using a single predicate type makes the structured model equivalent to the unstructured model. For the hierarchical model they firstly evaluated it on each of the predicate types and then on using combination of multiple field types. In the first case they also used different weighting scheme for individual predicates.
The **query** set: consist of 92 and 50 keyword queries for the years 2010 and 2011 respectively. They are the obtained from web search engine logs as described in 2.2.4.2.
As **evaluation measures**: they used the standard IR evaluation metrics: MAP, P@10 and nDCG. The obtained results for the unstructured baseline are shown in Figure 2.19, the ones obtained from the structured baseline are shown in Figure 2.20 and the ones obtained from the hierarchical model are shown in Figure 2.21 for the use of single predicate and in Figure 2.22 for the use of multiple predicates.

2.2.5.3 Example Based Entity Search in the Web of Data

In [6] Balog and Neumayer evaluated and compared a set of baseline methods for entity retrieval and executed an analysis on topics pointing out the features that makes this dataset challenging.
Dataset:they used the DBpedia dataset described in Section 2.2.3.
Document creation: they indexed all the entities with predicate of the type "label" and used both unstructured and structured approaches for the entity representations. In particular for the structured method they considered the

Predicate types	2010 MAP	2010 P@10	2010 NDCG	2011 MAP	2011 P@10	2011 NDCG
ALL	0.2069	0.3141	0.3827	**0.2072**	0.1880	**0.2947**
Name-only	0.2054†	0.3043	0.3969	0.2004	0.1900	0.3097
Attributes-only	0.2058‡	0.3391†	0.3915	0.2200†	0.1900	0.3330†
OutRelations-only	0.0866‡	0.1761‡	0.2185‡	0.0935‡	0.1020‡	0.1667‡
InRelations-only	0.0955‡	0.1967‡	0.2257‡	0.0689‡	0.0980‡	0.1540‡
ALL-but-Name	0.1568‡	0.2620‡	0.3210‡	0.1563‡	0.1440‡	0.2467‡
ALL-but-Attributes	0.1820‡	0.2859‡	0.3416‡	0.1637‡	0.1600‡	0.2419‡
ALL-but-OutRelations	0.2029‡	0.3109	0.3685‡	0.2002‡	0.1800	0.2826
ALL-but-InRelations	**0.2125**	**0.3196**	**0.3848**	0.2037‡	**0.1900**	0.2826

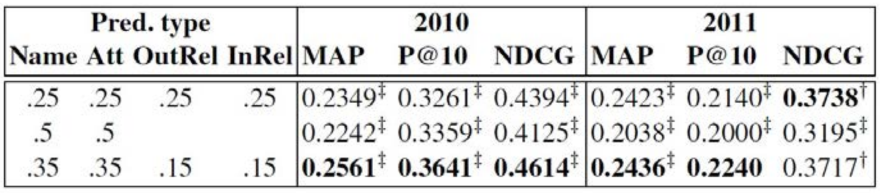
Figure 2.19: Results of the retrieval for the unstructured entity model. Significance calculated against the first line [32].

Pred. type weights				2010			2011		
Name	Attr	OutRel	InRel	MAP	P@10	NDCG	MAP	P@10	NDCG
.25	.25	.25	.25	**0.2816**‡	0.3989‡	**0.4943**‡	0.2605‡	**0.2420**‡	0.3966‡
.5	.5			0.2490‡	0.3663‡	0.4608‡	0.2506‡	0.2300‡	0.3845‡
.35	.35	.15	.15	0.2804‡	**0.4043**‡	0.4934‡	**0.2614**‡	0.2300‡	**0.3968**‡

Figure 2.20: Results of the retrieval for the structured entity model. Significance calculated against the ALL setting of the unstructured entity model of Figure 2.19 [32].

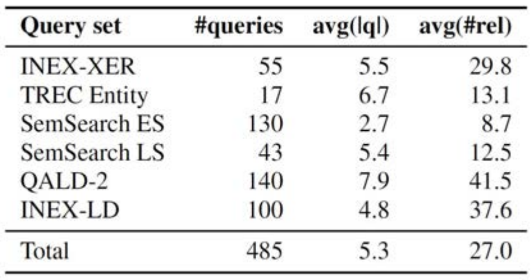
Predicate type	Predicate weighting	2010 MAP	2010 P@10	2010 NDCG	2011 MAP	2011 P@10	2011 NDCG
Attributes	Uniform	0.2350‡	0.3261	0.4403‡	0.2423‡	0.2140	0.3738
	Length	0.2116‡	0.2870‡	0.4149	0.2292‡	0.2060	0.3477
	AvgLength	0.2129‡	0.2859‡	0.4144	0.2287‡	0.2060	0.3511
	Popularity	0.2318‡	0.3109†	0.4385‡	0.2514‡	0.2140	0.3791
OutRelations	Uniform	0.1185‡	0.1783	0.2607	0.1221‡	0.1220†	0.2046‡
	Length	0.0915‡	0.1663	0.2122	0.0980‡	0.0920	0.1647
	AvgLength	0.0908‡	0.1533†	0.2131	0.0958‡	0.0940	0.1677
	Popularity	0.1067‡	0.1674	0.2439	0.1233‡	0.1260†	0.2070‡
InRelations	Uniform	0.1073‡	0.2054†	0.2387	0.0800‡	0.1160†	0.1749‡
	Length	0.0941‡	0.1793†	0.2188	0.0703	0.0940	0.1593
	AvgLength	0.0921‡	0.1696‡	0.2212	0.0776‡	0.0980	0.1751‡
	Popularity	0.1117‡	0.2022	0.2452†	0.0891‡	0.1240†	0.1903‡

Figure 2.21: Results of the retrieval for the hierarchical entity model [32].



Pred. type				2010			2011		
Name	Att	OutRel	InRel	MAP	P@10	NDCG	MAP	P@10	NDCG
.25	.25	.25	.25	0.2349‡	0.3261‡	0.4394‡	0.2423‡	0.2140‡	**0.3738**†
.5	.5			0.2242‡	0.3359‡	0.4125‡	0.2038‡	0.2000‡	0.3195‡
.35	.35	.15	.15	**0.2561**‡	**0.3641**‡	**0.4614**‡	**0.2436**‡	**0.2240**	0.3717†

Figure 2.22: Results of the retrieval for the hierarchical entity model. Significance calculated against the structured entity model of Figure 2.20 [32].



Query set	#queries	avg(|q|)	avg(#rel)
INEX-XER	55	5.5	29.8
TREC Entity	17	6.7	13.1
SemSearch ES	130	2.7	8.7
SemSearch LS	43	5.4	12.5
QALD-2	140	7.9	41.5
INEX-LD	100	4.8	37.6
Total	485	5.3	27.0

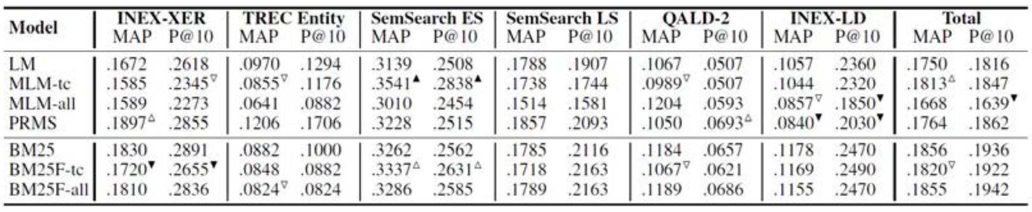
Figure 2.23: Queries used for the evaluation [6]

1000 most frequent predicates as fields and tested both a representation with a single field, where there is a single content field collapsing all the predicates, and a query-term specific field weighting.
Retrieval: For the comparison they considered two baseline methods: one based on language modeling and one based on BM25. Specifically they used: LM [50], MLM-tc with title and content fields (with weights respectively: 0.2 and 0.8) [33], MLM-all with all the fields considered with equal weight [33], PRMS: the Probabilistic Retrieval Model for Semistructured Data, BM25 with standard parameters settings ($k_1 = 1.2$, $b = 0.8$)) [39], BM25F-tc with title and content fields (with weights respectively: 0.2 and 0.8) [39] and BM25F-all considering all fields with equal weights.
The **query** set: consisted of many different queries taken from different benchmarking evaluation campaigns (see Figure 2.23).

- INEX-XER: they took 55 queries whose aim was to retrieve Wikipedia pages from a web corpus. They mapped the Wikipedia pages into the corresponding DBpedia entities.

- TREC Entity: they took 17 queries of the 20 original ones because they showed no relevant results from DBpedia.

- SemSearch ES: they took 130 queries, the ones with relevant results from



Model	INEX-XER		TREC Entity		SemSearch ES		SemSearch LS		QALD-2		INEX-LD		Total	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
LM	.1672	.2618	.0970	.1294	.3139	.2508	.1788	.1907	.1067	.0507	.1057	.2360	.1750	.1816
MLM-tc	.1585	.2345$^{\triangledown}$	.0855$^{\triangledown}$	.1176	.3541$^{\blacktriangle}$	.2838$^{\blacktriangle}$	.1738	.1744	.0989$^{\triangledown}$	.0507	.1044	.2320	.1813$^{\vartriangle}$	.1847
MLM-all	.1589	.2273	.0641	.0882	.3010	.2454	.1514	.1581	.1204	.0593	.0857$^{\triangledown}$	.1850$^{\blacktriangledown}$	.1668	.1639$^{\blacktriangledown}$
PRMS	.1897$^{\vartriangle}$	.2855	.1206	.1706	.3228	.2515	.1857	.2093	.1050	.0693$^{\vartriangle}$	.0840$^{\blacktriangledown}$	.2030$^{\blacktriangledown}$	.1764	.1862
BM25	.1830	.2891	.0882	.1000	.3262	.2562	.1785	.2116	.1184	.0657	.1178	.2470	.1856	.1936
BM25F-tc	.1720$^{\blacktriangledown}$	.2655$^{\blacktriangledown}$	.0848	.0882	.3337$^{\vartriangle}$	.2631$^{\vartriangle}$	.1718	.2163	.1067$^{\triangledown}$	.0621	.1169	.2490	.1820$^{\triangledown}$	.1922
BM25F-all	.1810	.2836	.0824$^{\triangledown}$	.0824	.3286	.2585	.1789	.2163	.1189	.0686	.1155	.2470	.1855	.1942

Figure 2.24: Results and baseline comparison. Significance for rows 2-4 is tested against row 1; for rows 6-7 tested against row 5 [6].

DBpedia, out of the total of 142 queries.

- SemSearch LS: they took 43 queries, the ones with relevant results from DBpedia, out of the original 50 queries whose aim was to retrieve a group of entities that match certain query criteria.

- QALD-2: they took 140 from the set generated collapsing both the training (100 queries) and the testing (100 queries) set of the campaign. In particular they took those queries that have answers in the DBpedia dataset.

- INEX-LD: they took all the 100 queries of the campaign mapping relevant Wikipedia pages to DBpedia entities.

For all of these queries they applied a preprocessing phase in which they removed all the additional markup, type information etc. and considered only the keyword part. They also made the relevance to be binary, they normalised every URIs to conform the one used in DBpedia dataset and filtered out URIs that are not DBpedia entities.
As **evaluation measures**: they used the standard IR evaluation metrics: MAP and P@10. They compared the results from the various different systems obtaining the ones represented in 2.24.

2.2.5.4 Comparisons

Summarizing what we saw in the previous Section 2.2.5 we would like to make some comments onto the results obtained by those methods [10, 32, 6]. Blanco et al. obtained a result that improves 50% over the classic BM25 and 42% over the best run of the SemSearch'10 using MAP as shown in Figure 2.18. Observing the queries in details they could assert that their system does poorly on:

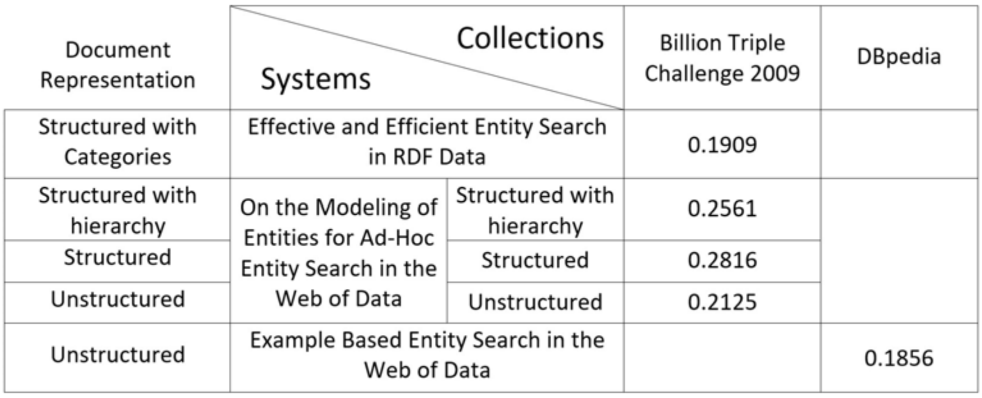
1. Long queries.
2. Queries with one relevant result that comes from a domain that the authors marked as unimportant.
3. Queries that has not directly relevant resource in the BTC dataset.

while they do well on:

1. Short queries highly selective.
2. Queries with one relevant result that they managed to find.

Neumayer et al. through their experiments stated that the combination of multiple predicates types failed in the unstructured case and that incorporating relations did not improve the system with respect to using just names or attributes alone as shown in Figure 2.19. This can be justified by the fact that there is more textual content for predicates type than for name or attribute and therefore the focus results moved from the entity itself. For the structured approach instead they asserted that relations can contribute to overall performance as shown in Figure 2.20. Both the unstructured and structured approaches suffer from a limitation: they abandon a part of the semantics behind the data. The hierarchical model proposed tries to consider this semantics and the results shows that modeling individual predicates is more effective than merging their contents into a flat representation as shown in Figure 2.21 and Figure 2.22. When they combine multiple predicates indeed they obtains better results than the unstructured entity model but they do not outperform structured entity model.

Balog et al., unlike the previous two papers, made a comparison between queries in order to evaluate each systems with respect to a specific subset of the query set as shown in Figure 2.24. They could see that different query sets exhibit different levels of difficulty. The easiest one are the SemSearch ES queries because they request a specific entity through its name. The most difficult one are those expressed in natural language such as TREC Entity, QALD-2 and INEX-LD. Others that are list-type queries are of medium difficulty because they formulate their information need by a mixture of keywords and natural language, such as INEX-XER and SemSearch LS. They also observed that there is not a specific model that outperform the others and that shows a significant improvement.



Document Representation	Collections / Systems		Billion Triple Challenge 2009	DBpedia
Structured with Categories	Effective and Efficient Entity Search in RDF Data		0.1909	
Structured with hierarchy	On the Modeling of Entities for Ad-Hoc Entity Search in the Web of Data	Structured with hierarchy	0.2561	
Structured		Structured	0.2816	
Unstructured		Unstructured	0.2125	
Unstructured	Example Based Entity Search in the Web of Data			0.1856

Figure 2.25: Summary of the results of papers [10, 32, 6]. In the first column the type of document representation use for each paper, in the second column the system considered. In the third column we have the best result for each paper that has used the BTC dataset while in the fourth column the best result for each paper that used DBpedia dataset.

We can summarize all previous results into a table, see Figure 2.25, where on columns we have the two different collections used (Billion Triple Challenge 2010 and Dbpedia) and on rows we have the three papers with their proposed methods classified by the type of document representation they use (unstructured, structured or hierarchical). Into the table we report, in terms of MAP, the best result that each paper obtained on its dataset.

2.3 Graph Embedding Techniques

Non Euclidean data structures like graphs can represent more complex elements, in particular their structure and features, with more accuracy than Euclidean one. A graph could be seen as a data model that represent real-world entities and relationships, it can be useful for many real applications like: the World Wide Web or social networks [51]. It is interesting to integrate this model with new machine learning methods, this is the reason why the Geometric Deep Learning field was born. The Geometric Deep Learning is the machine learning field that studies new ways to extend deep learning in order to use graph structures. Its aim is to exploit them in order to improve machine learning tasks for network analysis like: node classification, link prediction, visualization, etc. [51, 43]. This can be done by representing graph nodes or edges through numerical vectors. This approach moreover allows us to take advantage of all the vector space theory and operations.

One of the first neural network implemented for building word vector representations was the one by Bengio [7]. Starting from here there have been several other implementations like the one of Mikolov [30] and mainly the skip-gram model at the base of word2vec approach described in [29]. The initial idea of Bengio was to represent words into a continuous space with a reduced number of parameters in order to make them more usable and easy to manipulate. To obtain these, he developed a feedforward neural network with a linear projection layer and a non-linear hidden layer. This network allowed him to calculate a distributed representation of word, called word embeddings, together with a probability function for word sequences in term of these representation namely their statistical language model [29].

We can therefore say that graph embeddings aim to represent the topology and structure of a graph through vectors or a set of vectors. This representation is considered good if it includes all the domain knowledge related to the graph. The fact that these representations are in a compressed form allows us for better computational performance than previous graph models based on adjacency matrix which have dimension $|V| \times |V|$, with $|V|$ the number of vertices of the graph, and therefore could become very huge even in medium-sized graph making matrix multiplication too expensive and complex [31]. Moreover representing embeddings in a vector space allows us to use all vector space theory and operations which have a richer toolset of approaches [31, 18]. These methods for learning word representations have two other important features: one is that they are transductive so they can only predict instances that are already been observed in the graph at training time and the second one is that they are obtained separately from the supervised task so they can be used in different tasks [49].

2.3.1 Word2Vec

The adoption of the neural networks for the automatic learning of embeddings, after Bengio [7], become the state-of-the-art in word vector representation process. The majority of the systems nowadays is based on the skip-gram model introduced by Mikolov in [29]. This model is composed by three layer as shown in Figure 2.26:

1. Input layer
2. Hidden layer

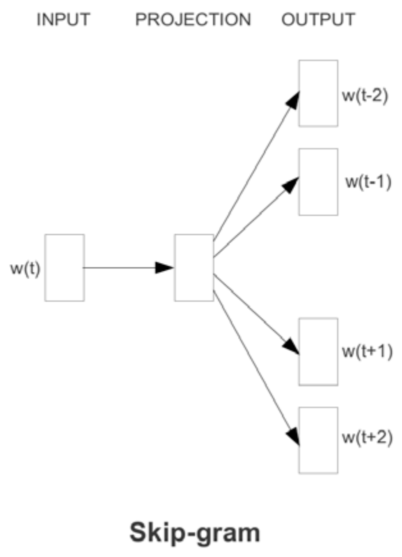


Figure 2.26: Word2Vec model architecture taken from [29].

3. Output layer

The input of the neural network is the representation of a word expressed through a one-hot vector, which has dimension $|D|$ equal to the number of words in the vocabulary of reference and contains a 1 in the corresponding position of the word itself. This input is passed through the input layer to the hidden layer. The hidden layer is composed by a matrix with as many rows as the number of words in the vocabulary and as many columns as the number of feature through which we want to represent our words. The number of features correspond to the number of neurons of the network. When multiplied by the input vector the matrix simply returns the row associated to that word, so we can look at this as a look up table (see Figure 2.27). We can therefore see that the rows are in effect the representations of the words we want to obtain. The output layer is a softmax regression classifier used to predict neighbor words in the sentence with respect to the input one. In order to optimize the network a gradient descent algorithm is executed, but this could slowdown the train in large network so two types of improvements were introduced later in [31]: a modified version of the objective function called *Negative Sampling* and a sub-sampling approach for frequent words. The first one is a modified version of the *Noise Contrastive Estimation* that turn out to be faster in training and better in creating vector representations for frequent words [31, 49]. The second one consist of a sub-sampling of the input words in relation to their frequency. This sampling aims to reduce the imbalance between frequent and rare words

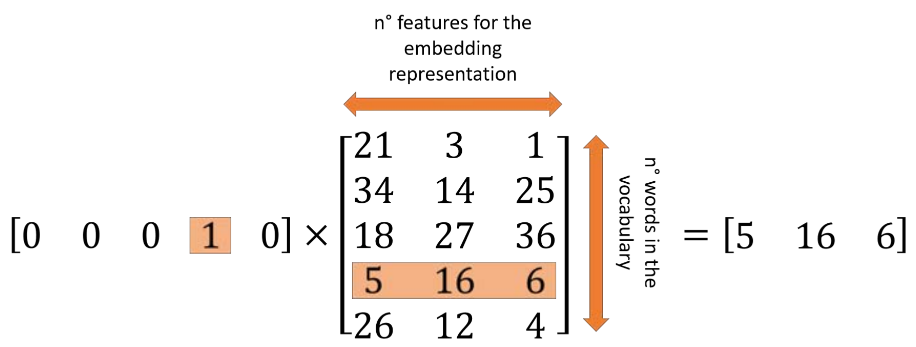

n° features for the embedding representation
n° words in the vocabulary
[0 0 0 1 0] ×
21 3 1
34 14 25
18 27 36
5 16 6
26 12 4
= [5 16 6]

Figure 2.27: Example of matrix multiplication in the hidden layer with 3 features (neurons) and 5 words in the vocabulary. We represent our word with a one-hot vector with a 1 in the position of the word in the vocabulary. We can see that this multiplication return the embedding of the input word.

considering that frequent words representations do not change significantly after training on millions of examples.

The task for which the network is trained is: given a word in input, the probability for each word in the vocabulary to be the nearby one the input. This train is done passing pairs of words taken from a context of a phrase and updating matrix weights in order to make the network predict from the first word the second one. The context is chosen setting a parameter called *Window* that identifies the number of words to consider before and after the input one (see Figure 2.28). In particular what we obtain is a distribution probability of all the words of the vocabulary with respect to the input word. This task is called *Fake task* because in our case the aim is not this but instead the embedding realization.

2.3.2 DeepWalk

With the introduction of this new approach based on neural networks researchers began to think of a way to exploit it in combination with data structures such as graphs. DeepWalk was though for this purpose, it extends the skip-gram model generating a kind of context in the input graph, similar to the one of phrases used in Word2Vec in order to predict the context of the graph. More specifically here the context is composed by random walks generated on the graph, the words representations correspond to nodes representations and the context to predict is the context of the node in the graph [49]. As Perozzi says in [35] "*DeepWalk is a novel approach for learning latent representations of vertices in a network. These latent representations*

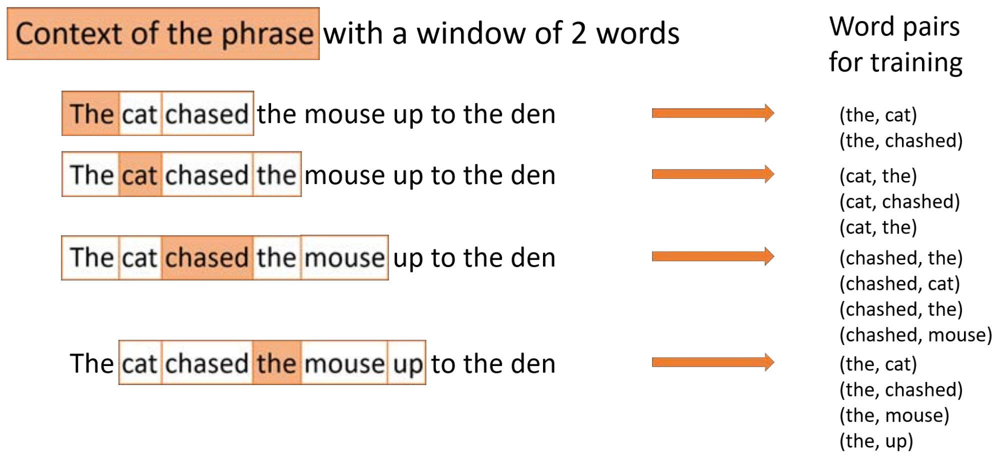


Figure 2.28: Example of an input phrase. For the word pair creation we set a window of two words near the input one from which we select the words for the input samplings. The word in orange is the input word while the words rounded by a rectangular box are the one in the window candidate for the pairs.

encode social relations in a continuous vector space, which is easily exploited by statistical models.". The target embeddings we want to obtain with DeepWalk are then *Latent representation*, namely vectors containing information about not only node connections but also neighborhood similarities and community memberships that are not explicit in the graph. Analyzing the functioning of this algorithm in more detail we are going to describe its main two components:

1. Context generation
2. Embedding computation

The main task of the first part of the algorithm is to generate a context in a graph. A context on a graph, maintaining a similarity with textual sentences, could be seen as a sequence of nodes instead of a sequence of words. This sequence is a random walk of a specific length t generated from a chosen node and can be defined, according to [35], as:

> We denote a random walk rooted at vertex v_i as W_{v_i}. It is a stochastic process with random variables $W^1_{v_i}$, $W^2_{v_i}$, ..., $W^k_{v_i}$ such that $W^{k+1}_{v_i}$ is a vertex chosen at random from the neighbors of vertex v_k.

These random walks are generated a specific number of times γ for each node in the graph. Once we have obtained random walks we can use them for embedding computation. The second part is based on skip-gram algorithm using random walk as the usual sequence of words and creating windows context for training pairs generation starting from this sequence of nodes. Here, instead of predicting words close to the input one, we are going to predict nodes that are close to the input one.

2.3.3 Line

While graph embeddings algorithms we seen until now create vectors that incorporate features based on local structures, the LINE algorithm aim to keep in consideration also global structures. In particular with classic random walk generation we take into account topological features of nodes nearby the starting one leaving out those one that are less explicit in our graph. To get a better representation of the nodes, however, we might want to consider also features related to the node neighborhood. In order to do this, Tang in [43] introduce a new objective function whose aim is to create similar embeddings for nodes that has similar context, because two nodes can be consider similar not only if they are connected one to the other but also if they share the same context. In this way also nodes that are not connected but share the same neighbors have similar embeddings.

For the objective function creation Tang consider the fact that a node can have two roles: one that is the vertex itself and one that is a specific context of other vertices [43]. This two type of proximities are combined together in this way:

1. Train the network considering 1-st order proximity.
2. Train the network considering 2-nd order proximity.
3. Concatenate, for each vertex, the two embeddings obtained from these two methods.

2.3.4 Node2Vec

The same idea to take into account also similarities between the nodes given by their context has been studied by Grover in [18] with their algorithm called

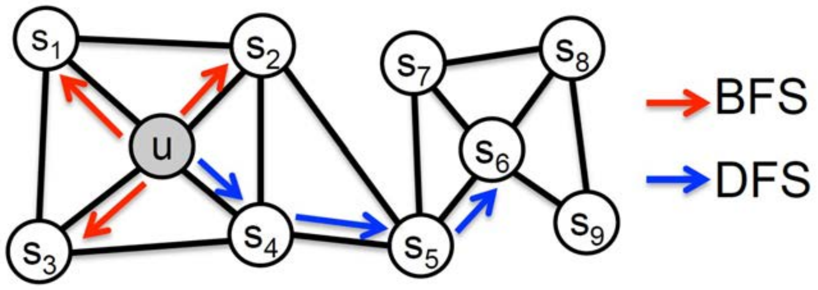


Figure 2.29: Example of a portion of a graph. In red we can see the BFS strategy while in blue we can see the DFS strategy. u is the starting node of the walk.

Node2Vec. Node2Vec is the state-of-the-art general-purpose scalable feature learning method for network analysis [51, 18]. It uses a sample strategy similar to the one of DeepWalk with the exception that in this case the random walk is generated following some probabilities related to visited and unvisited node of the graph during the walk. Using this algorithm we obtained vector representations that maximize the likelihood of preserving neighborhoods of nodes in a d-dimensional feature space [18], where as neighborhoods we consider all the nodes generated through the random walk. In addition to create a representation of nodes, this algorithm allows also for the creation of edges embeddings extending the operations to pair of nodes instead of a single node.

The approaches seen so far propose efficient algorithms that have a rigid notion of neighborhood and so they turn out to be rather insensitive to the connectivity pattern of the network. Node2Vec implementation instead wants to identify two specific kind of relationships between the nodes of the graph that it received in input:

- Homophily: with homophily we mean all those nodes that are highly connected one to each other and then belong to the same community or cluster [16, 48]. In Figure 2.29 we can see that nodes u and s_1 belongs to the same network community and so they fall into this first case.

- Structural roles: with structural roles we mean all those nodes that have similar structural roles in the network [23]. In Figure 2.29 we can see that nodes u and s_6 act as hubs of their communities and then they fall into the second case.

These aspects can be identified by two different traversal algorithm: Breadthfirst Sampling (BFS) and Depth-first Sampling (DFS). The first one, represented in red in Figure 2.29, aims to explore nodes near the starting one, in particular it explores nodes that are immediate neighbors of it. The second one, represented in blue in Figure 2.29, aims to explore consecutive nodes with increasing distances from the source one.

Node2Vec consider these two graph traversal strategies into its objective function defining it as:

$$\max_f \sum_{u \in V} \log P_r(N_S(u) \mid f(u))$$

where:

- f is a function mapping from nodes to feature representations: $f : V \to \mathbb{R}^d$ with d the number of dimensions of the feature representation.
- $u \in V$ is the source node.
- $N_S(u)$ is a network neighborhood of node u generated through the sampling strategy S.

This function, as said by Grover in [18]: aims to "*maximise the log-probability of observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation given by f*". The authors makes two assumptions in order to simplify this function [18]:

1. Conditional independence. They assume that the likelihood of observing a neighborhood node is independent of observing any other neighborhood node given the representation of the source. In this way they could factorize the likelihood into:

$$P_r(N_S(u) \mid f(u)) = \prod_{n_i \in N_S(u)} P_r(n_i \mid f(u))$$

2. Symmetry in feature space. A source node and neighborhood node have symmetric effect over each other in feature space. Thanks to this assumption we can model the conditional likelihood over every source-neighborhood node pair as a softmax unit parametrized by a dot product of their features:

$$P_r(n_i \mid f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(u) \cdot f(u))}$$

Than the objective function can be rewritten as:

$$\max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_{\mathbb{S}}(u)} f(n_i) \cdot f(u) \right]$$

In order to make the computation less expensive the authors decided also to approximate the per-node partition function $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$ using negative sampling [31] and the objective function using stochastic gradient ascent. While previous papers used a notion of neighborhood related to a sliding window on consecutive words, in Node2Vec many different neighborhood are computed from the same source u.

Let see how random walks are generated in order to detect different structural features of the nodes of the graph. Formally [18]:

> given a source node u, denoting c_i the i-th node of the walk of fixed length l starting with $c_O = u$. Nodes c_i are generated by the following distribution:
>
> $$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$
>
> where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

In Node2Vec they would like to use a mixture of both BFS and DFS strategy, so they set the unnormalized transition probability as:

$$\pi_{vx} = \alpha_{pq} \cdot \omega_{vx}$$

where :

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

with d_{tx} denoting the shortest path distance between nodes t and x. Suppose that we have just passed from node t to node v through the edge (t, v), as shown in Figure 2.30. Now the algorithm has to decide the next step evaluating π_{vx} on edges (v, x) from v. This choice is led by p and q parameters. The p parameter controls the likelihood of immediately revisiting a node in the

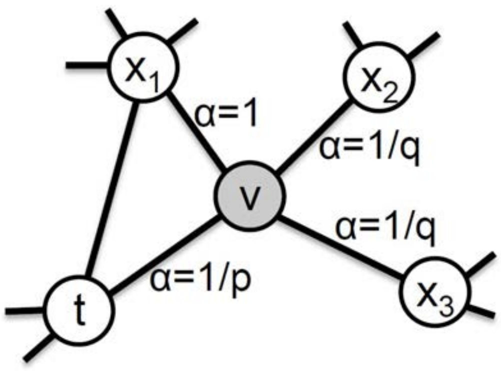


Figure 2.30: Illustration of the random walk procedure in Node2Vec. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α [18].

walk, encouraging a moderate exploration and avoiding 2-hop redundancy. The q parameters indeed allows the search to differentiate between inward and outward nodes: if $q > 1$ the walk remains close to the node t encouraging a local exploration, if $q < 1$ the walk is allowed to get away from the source encouraging the visit of nodes further away.

Although this algorithm proved to be efficient and effective for the learning of feature vectors of nodes and edges in graph data structure Zouh in [51] underlined some issues that incur when using the Node2Vec Standford implementation available in [25, 26]:

1. Their python and C++ implementation is not scalable and works well only in small graph that can fit into a single machine.

2. Their SPARK implementation is scalable and more efficient thanks to the execution of in-memory operations and caching, but has a significant overhead both in time and space and goes out of memory due to RDDs management and shuffle phase. Moreover it brought to vector representation that has poor quality because it limits the number of possible random walk to 30 for each node in order to save memory for storing the transition probabilities.

3 Solution Design

In our approach we create entity representations that consider the structure of the context in which entities are contained. As described in Section 2.3.4 we create these representations starting from a RDF graph where we consider the graph neighborhood of the entity node as its context. We think that considering also entities related to the ones we want to retrieve can enrich the representation of the single one, giving a semantic context that helps to retrieve relevant entities that are not explicitly mentioned in the user information need. For example, if we are talking about "Chefs with a show on the Food Network." we are not explicitly asking for a specific chef we know having a tv show, but our system it's able to retrieve many of the entities requested because we can assume they belong to the same subgraph and therefore sharing the same context. Our pipeline is composed of five parts as shown in Figure 3.1:

1. **Entity representations**. In this phase we generate a vector representation for each entity considering its properties and its context information. We want to obtain these vectors of features because they have mathematical properties that allow us to execute operations considering words similarity. In particular we use a graph embedding technique that produce, for each entity, a single vector representation that takes into account the entity context through an exploration of its neighborhood in the RDF graph. It is these embeddings that allow us to mathematically represent the entities semantic contexts necessary for the next clustering phase.

2. **Clustering**. In this phase we execute clustering on embedding vectors. This technique allow us to define semantic contexts by grouping together highly correlated and similar entities. This is done considering the distances between vectors. The more two entities represent different concepts the more their vectors are distant and therefore higher is the probability that they belong to two different clusters.

3. **Documents creation**. In this phase we want to bring back our approach to the classical text retrieval one based on documents retrieval. In order to do this we want to create "virtual documents" by associating each of them with a cluster of entities. In this way our documents identify

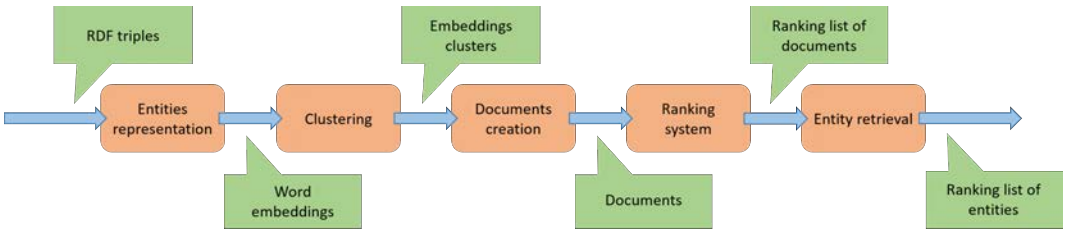


Figure 3.1: Pipeline of the thesis. From RDF triples we obtain entities representations. We use these embeddings to execute a clustering in order to group them by similarity. These clusters are used as a reference for document creation, so our document is in effect made of all the entities contained in a cluster. These documents represent then our collection for the retrieval part of the process which returns a ranked list of them. From this list we finally extract a ranked list of entities in answer to the user information need.

a specific semantic field, such as: "football", "chefs", "politicians".

4. **Ranking system**. In this phase we use the classic text retrieval technique on our collection of documents i.e. collection of clusters. In this way we obtain a ranked list of fields where we find in higher positions clusters that are more semantically close to the user request.

5. **Entity retrieval**. Through its demand, the user wants to get a list of relevant entities, it is therefore necessary to create it starting from our virtual documents ranked list. In this last phase, we implement different approaches for this purpose. In particular we create this final list starting from the ranked list of clusters and the ranked list of entities obtained through the state-of-the-art method.

Summing up: we first of all create entity vector representations through *graph embeddings techniques*, than we use these representations in a *clustering* process to obtain documents for the retrieval phase. Then, we use information *retrieval systems* and *fusion methods* with the state-of-the-art entity retrieval process to obtain a ranked list of entities.

3.1 Entity representations

The starting point of the creation of our entity representations is a RDF graph. As described in Section 2.2.2, this graph is generated considering each RDF triple of the form subject-predicate-object as a sequence of node-edge-node. In this way we associate a node to each entity and we represent

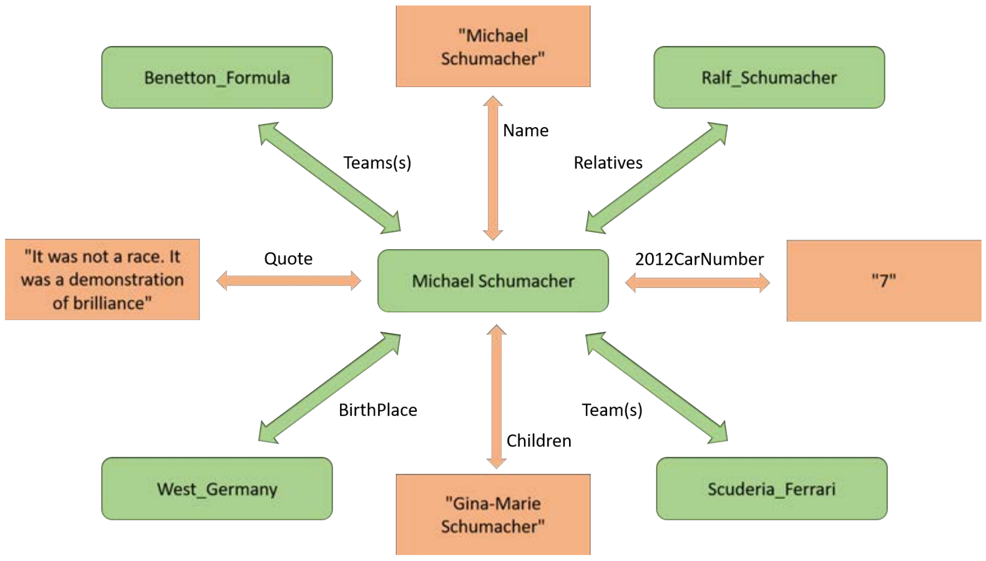


Figure 3.2: Entity subgraph, taken from DBpedia dataset, that refers to the Entity Schumacher and a part of its neighborhood. Represented by green rounded rectangles entities, by green arrows relationships between entities, by orange rectangles literals, by orange arrows predicates between entities and literals. In green the entity graph.

relationships through edges. In our approach we consider as input for the embedding algorithm a subgraph of the one just presented. Before describing our subgraph creation we firstly want to introduce what we call an "entity graph" in a classic RDF graph (see Figure 3.2). We define as *Entity graph* the subgraph constituted by all those nodes that represent entities (thus excluding literals) and all those edges that represent relationships connecting them. In Figure 3.2 the entity graph is represented in green. Our subgraph is an entity graph.

3.1.1 Entities Embeddings

The subgraph determined in the previous section is the input of the graph embedding process. For the entities vector representations we use the Node2Vec implementation presented by Elior Cohen in [13]. We choose this implementation because it added the support for big graphs that can not be found in the original Node2Vec Python version of the SNAP framework [25]. This support is obtained allowing for a parallel computation of the random walks of the algorithm. Eliorc Node2Vec accepts several different parameters, the most important ones are:

1. *Embedding dimension*: the number of features that we want to use for

our entity representation (default = 128).

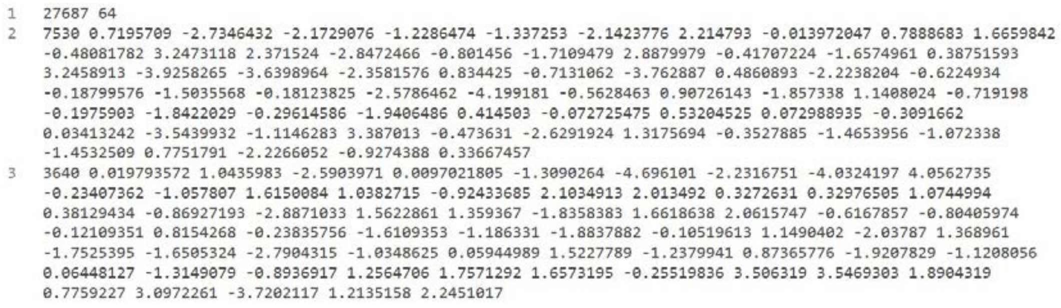
2. *Walk length*: the number of entities (nodes) that we want to traverse in each computed random walk. It represent the length of our "context" or "neighborhood" (default = 80).
3. *Number of walks*: is the number of walks that we want to generate starting from the same chosen vertex of the graph (default = 10).
4. *P*: the hyper parameter p of the algorithm (default = 1).
5. *Q*: the hyper parameter q of the algorithm (default = 1).
6. *Weight_key*: for the weighted graph it defines the key for the weight attribute (default = "weight").
7. *Workers*: the number of workers that we want to use for the parallel computation of random walks (default = 1).
8. *Temp_folder*: the path of the folder in which saving a shared memory copy of the graph. This folder is use for graphs that are too big to fit in memory during the algorithm execution.

Since Node2Vec internally recalls the Word2Vec gensim implementation ([38, 17]) it is also possible to set Word2Vec parameters such as:

1. *Window size*: the number of words to consider before and after the input one in the training phase. This is the window defined in Section 2.3.1.
2. *Min_count*: a threshold defined in order to ignore words with a frequency lower than min_count.
3. *Batch_words*: the number of words to be pass to a worker for parallel computation.

Let's remember that in our case the words of Word2Vec are node identifiers. We set the following parameters:

- Embedding dimension = 64
- Walk length = 4
- Number of walks = 20



```
27687 64
7530 0.7195709 -2.7346432 -2.1729076 -1.2286474 -1.337253 -2.1423776 2.214793 -0.013972047 0.7888683 1.6659842
-0.48081782 3.2473118 2.371524 -2.8472466 -0.801456 -1.7109479 2.8879979 -0.41707224 -1.6574961 0.38751593
3.2458913 -3.9258265 -3.6398964 -2.3581576 0.834425 -0.7131062 -3.762887 0.4860893 -2.2238204 -0.6224934
-0.18799576 -1.5035568 -0.18123825 -2.5786462 -4.199181 -0.5628463 0.90726143 -1.857338 1.1408024 -0.719198
-0.1975903 -1.8422029 -0.29614586 -1.9406486 0.414503 -0.072725475 0.53204525 0.072988935 -0.3091662
0.03413242 -3.5439932 -1.1146283 3.387013 -0.473631 -2.6291924 1.3175694 -0.3527885 -1.4653956 -1.072338
-1.4532509 0.7751791 -2.2266052 -0.9274388 0.33667457
3640 0.019793572 1.0435983 -2.5903971 0.0097021805 -1.3090264 -4.696101 -2.2316751 -4.0324197 4.0562735
-0.23407362 -1.057807 1.6150084 1.0382715 -0.92433685 2.1034913 2.013492 0.3272631 0.32976505 1.0744994
0.38129434 -0.86927193 -2.8871033 1.5622861 1.359367 -1.8358383 1.6618638 2.0615747 -0.6167857 -0.80405974
-0.12109351 0.8154268 -0.23835756 -1.6109353 -1.186331 -1.8837882 -0.10519613 1.1490402 -2.03787 1.368961
-1.7525395 -1.6505324 -2.7904315 -1.0348625 0.05944989 1.5227789 -1.2379941 0.87365776 -1.9207829 -1.1208056
0.06448127 -1.3149079 -0.8936917 1.2564706 1.7571292 1.6573195 -0.25519836 3.506319 3.5469303 1.8904319
0.7759227 3.0972261 -3.7202117 1.2135158 2.2451017
```

Figure 3.3: Output file containing embeddings. In the first line we have the number of embeddings and their dimensions (in this case 64 features). In the following lines the vector representations of three entities, in particular the ones with node ID: 7530, 3640, 564 (first number of the row).

- Workers = 50
- Window size = 10
- Min_count = 1
- Batch_words = 4

For all the other values we use the default value. Once executed Node2Vec we obtain an embedding for each node/entity of our subgraph (see Figure 3.3).

For embeddings we also try to use the SNAP SPARK implementation but we abandoned it due to memory problems described in Section 2.3.4.

3.2 Clustering

For the creation of virtual documents that exploits entity embeddings properties we use clustering techniques. The idea behind this choice is to create a document that contains more than one related and similar entities because, in this way, it is possible to return to the user also entities that are not explicitly mentioned in the user information need but that are still relevant for his/her request. In order to do this we execute clustering on those entity representations returned by the embedding phase. For this phase we use the K-MeansSort algorithm with 700 clusters because it is easy to initialize, it obtains good results and it is highly scalable. This last feature is essential because it allows us to use it with big graphs. This algorithm is a modified version of the classic K-Means. In particular it modifies the way in which the algorithm decides to which class assign points. The main aims of

K-MeansSort is to speedup this decision process. In order to do this it exploits the triangle inequality and the means sorting in order to reduce the number of comparison to be made between distances calculated between points and means [36]. Looking in detail into the algorithm, lets consider the triangle inequality $d(\mu_i, \mu_j) \leq d(p, \mu_i) + d(p, \mu_j)$ where d stands for the distance, μ_i stands for the mean of the i-th class (cluster) and p stands for the point. Manipulating this equation we can obtain the inequality $d(\mu_i, \mu_j) \geq d(\mu_i, \mu_j) - d(p, \mu_i)$. From this we can conclude that if $d(\mu_i, \mu_j) \geq 2d(p, \mu_i)$ it stands that $d(p, \mu_j) \geq d(p, \mu_i)$ without computing $d(p, \mu_j)$. The algorithm then precomputes the distance $d(\mu_i, \mu_j)$ for each pairs of means before iteration and then check if the previous condition is verified. If this is true the algorithm skip all the following calculations passing to the next cluster, otherwise it has to update its distance from the nearest cluster calculating $d(p, \mu_j)$. A further speedup is done sorting the means in order of increasing distance and then, for each point, compare the distance $d(p, \mu_c)$ of the point from the cluster with the other sorted means. If the means we are considering is far enough from the actual point distance we can skip all remaining means and continue with the next point.

For the implementation of this phase we use a framework called ELKI [42, 15] which gives tools for the implementation of data mining algorithm, in particular for Clustering methods. Indeed ELKI is an open source data mining software written in Java, it implements mostly research algorithms, in particular those related to unsupervised methods in cluster analysis and outlier detection. As said in [15]: "*This framework offers data index structures such as the R*-tree that can provide major performance gains*". This framework provides highly scalable algorithms characterized by the independence between data, parsers, database, distance functions, etc. that allows you to more easily evaluate and compare these methods. The main design goals of this framework, as reported in [15], are:

- *Extensibility*: ELKI has a modular design that allows different combinations of data types, distance functions, algorithms, input formats, index structures and evaluations methods.

- *Contributions*: the modular design described in the previous point allows people to make even small and different contributions. These contributes help ELKI to grow.

- *Completeness*: authors try to continuously improve and integrate the

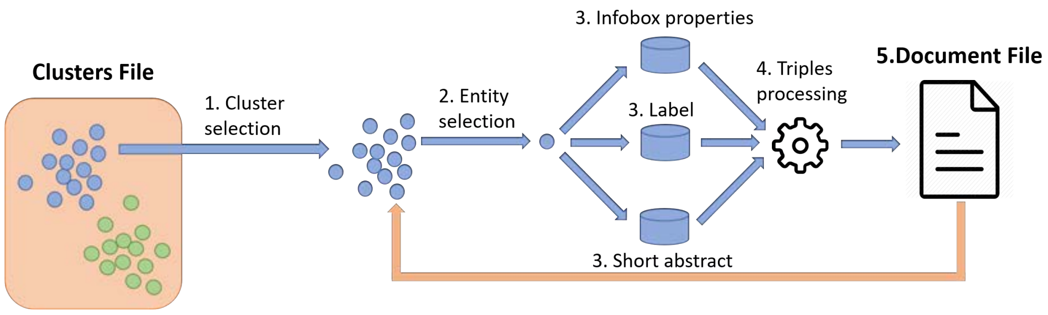


Figure 3.4: Representation of the process for document creation.

framework with as much published and credited works.

- *Fairness*: in order to obtain the most reliable comparisons possible, the contributors try to implement algorithms and other components as good as possible. This is done also making public all the source code for external improvements.
- *Performance*: the performances are improved thanks to modular architecture of ELKI and its index structures.

In our implementation we choose 700 clusters on a qualitative evaluation basis. We execute several versions of the K-MeansSort algorithm with different number of clusters, then we choose the best value looking at the total number of clusters obtained and at clusters coherence and completeness.

3.3 Documents creation

Once we get clusters, we can create all those virtual documents that become the collection for our retrieval phase. In order to do this, our approach associates to each semantic field, identified by a cluster, a document. In details the process consists of five phases (see Figure 3.4):

1. Select a cluster.
2. Take cluster entities one after the other.
3. For each entity we select the corresponding RDF triples that are all those triples that have that entity as subject or object.
4. We process the triples obtained above in order to get information in natural language by removing namespaces, replacing "_" with white space, removing "<>" of RDF triples and "" of literals.

5. The information obtained are merged in a textual file that represents the document associated to the cluster at hand.

Our documents collection is then obtained repeating the entire process for each cluster.

3.4 Ranking System

Once obtained the virtual documents we can proceed with the retrieval process. The aim of this phase is to apply the classic text retrieval technique to our virtual documents in order to obtain a ranked list of clusters. In this way we can identify which are the semantic field, i.e. clusters, that are more relevant for the user information need and then, in the next phase, execute a reranking of the entities inside these fields. For this phase we use the software Terrier [44], which is an open source search engine written in Java and developed by the University of Glasgow. This software implements state-of-the-art indexing and retrieval functionalities and provide a platform for the development and the evaluation of large-scale retrieval applications. As reported in the website [44], the main feature of this platform are:

- *Efficient*: it provides several indexing strategies such as multi-pass, large-scale single-pass, real-time.
- *Effective*: it provides state-of-the-art retrieval approaches such as divergence from randomness, BM25, support for supervised ranking models.
- *Flexible*: it allows for indexing and batch retrieval for all known TREC test collections.
- *Multi-lingual*: it can support collections written in languages other than English.
- *Extensible*: it follows a plugin architecture so it is easy to extend with new retrieval techniques and ranking features.
- *Interactive*: the results obtained from the retrieval can be viewed through a desktop search application, JSP web intarfaces or a website search application.

As described in Section 2.1.1.1 the indexing phase is composed of many different steps, in this thesis we use:

1. For the Lexical Analysis: the default English Tokeniser provided by Terrier.
2. For the Stop words removal: the default English stopword-list provided by Terrier.
3. For the Stemming: the Porter Stemmer.

We index all the virtual documents of the collection and then execute the retrieval using the BM25 model. Through this process we obtain a ranked list of documents i.e. a ranked list of clusters. The final phase uses this list in order to create a ranked list of entities to return to the user.

3.5 Entity Retrieval

Once obtained the ranked list of virtual documents we implement seven alternative methods in order to produce a ranked list of entities. In particular in three of these methods we firstly execute a reranking of the entities within each cluster and then we combine these entities for the creation of the final ranking to be returned to the user. For the reranking of the entities it is necessary to create a new collection composed of a document for each entity as it is done in the state-of-the-art methods. This because we want to rank all the entities inside a virtual documents, through the classic text retrieval method, considering each one independently. Once obtained the run with reranked entities we can reconstruct the final ranked list. This is done through the process shown in Figure 3.5 and described in Algorithm 1. All our first

Algorithm 1 Combination Algorithm

```
1: while exist clusterLine in clusterRun do
2:     rerankedEntitiesRun = openFile(topicID, clusterID)
3:     while rerankedEntitiesLine do
4:         if numEntity < maxEntity AND score > 0 then
5:             finalRun.write(rerankedEntity)
6:         end if
7:     end while
8: end while
```

three methods rely on a common procedure: we take the first row of the

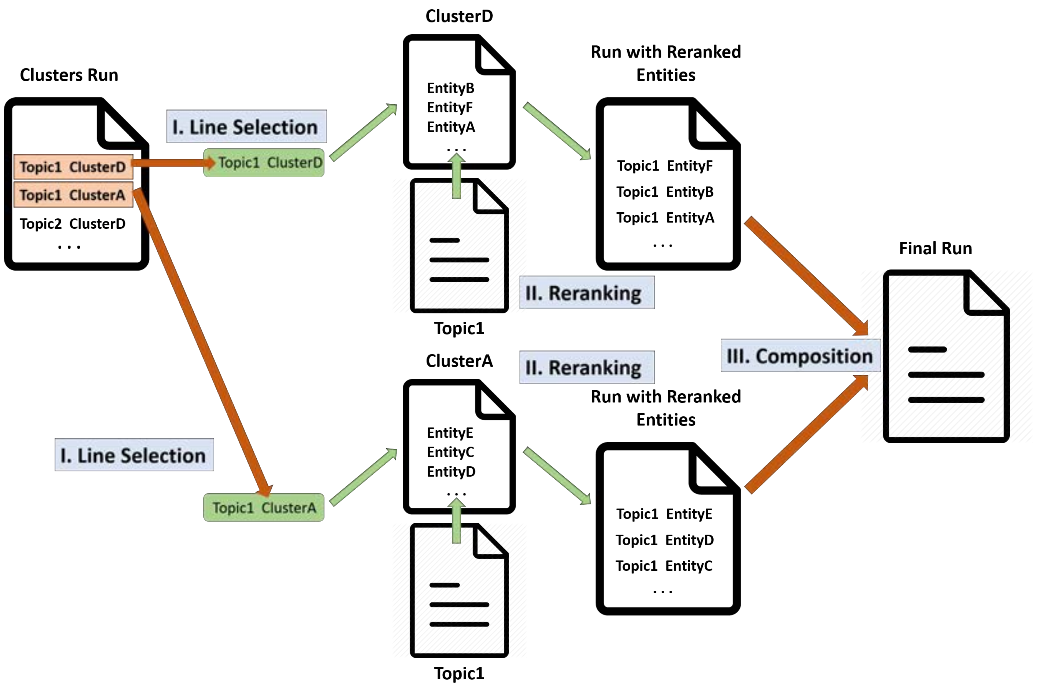


Figure 3.5: Final run creation process. In the first step we take one line from clusters run. In the second step we execute the reranking of the entities inside that cluster we selected on that topic we selected. In the third step we merge the results from each line of clusters run in order to create the final run.

cluster run together with its topic ID, cluster ID and score (line 1); we take the corresponding reranked entities run file containing the reranking of the entities in cluster ID for topic ID (line 2) and insert them into the final run (line 5). In particular, we insert *maxEntity* entities with *score* > 0, from the reranked entities run, for each line of the cluster run (line 4). The value of *maxEntity* is chosen through a qualitative evaluation basis. We execute several versions of combination algorithms with different values of *maxEntity*, then we choose the best value. What changes from one approach to another is the way in which we insert the entities into the final run. The different entities composition approaches are:

1. *Simple completion (SICO)* (see Figure 3.6): this approach takes a fixed number of entities from each reranked entities run. In particular we look at the corresponding reranked entities run and take a fixed number of entities (*maxEntity*) with *score* > 0 inserting them in the final run in the same order in which they appear in the reranked entities run. With this approach we preserve both clusters and entities ranking.

2. *Sum completion (SUMCO)* (see Figure 3.7): this approach executes the same process as the one described in SICO with two differences. The first one is that all the entities of the same topic are ordered through their scores before being insert into the final run. The second difference is that we do not use the entity score of the reranked entities run alone but we add to it the score of the cluster that contains the entity. This become the final score of the final run.

3. *Weighted completion (WECO)* (see Figure 3.8): this approach calculates the final score as:

$$\text{finalScore} = (\text{entityScore} + \text{clusterScore}) * \text{similarity}$$

   where *similarity* is the *cosine similarity* measure defined in [41]. In this case the similarity is computed between the cluster mean, that is a vector representing the centroid of the cluster to which the entity belongs, and the entity embedding vector. An entity that is positioned near the mean of the cluster to which it belongs, probably results to be very inherent to the topic associated to that cluster and highly related to other entities within the same set. In this way we consider how much an entity is representative for its cluster. Entities that are more representative for the cluster have the highest score and then are considered more important for retrieval purposes.

For comparison purposes we also implement the state-of-the-art method that does not use clustering but simply creates a document for each entity. In order to incorporate advantages from both the processes: the classic state-of-the-art method and our approach, we implement four fusion methods. These methods consist of five phases as shown in Figure 3.9 and described in Algorithm 2. As we can see from the line 1 of the algorithm, we take the first row of the run returned by the classic method together with its topic ID, entity ID and score. We compare this score with a threshold we call: *minScore* (line 2). This threshold is fixed in order to avoid the insertion of entities with low score into the final run. If this condition is satisfied we insert into the final run the entity of the classic run we are considering (line 3). In line 4 we check if the rank of the entity is less than a threshold we called *maxRank*. This threshold is fixed considering the number of entities, taken from classic run, we want to enrich

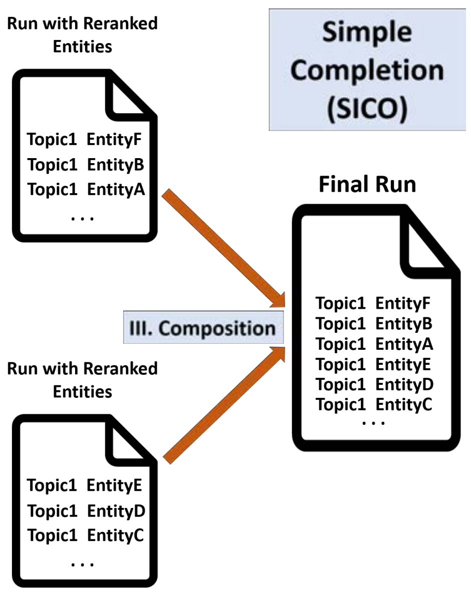


Figure 3.6: Third phase of Figure 3.5 in detail. From the reranked entities run of the clusterD we take the first 3 entities and merge them into final run. From the reranked entities run of the clusterA we take the first 3 entities and merge them into final run after the ones of clusterD in order to keep the ranking of both clusters and entities.

Algorithm 2 Fusion Algorithm

```
1: while exist classicLine in classicRun do
2:    if score > minScore then
3:       finalRun.write(classicEntity)
4:       if $rank < maxRank$ then
5:          clusterID = searchCluster(classicEntityID)
6:          rerankedEntitiesRun = openFile(classicTopicID, clusterID)
7:          while rerankedRunLine AND numEntity < maxEntity do
8:             finalRun.write(rerankedEntity)
9:          end while
10:      end if
11:   end if
12: end while
```

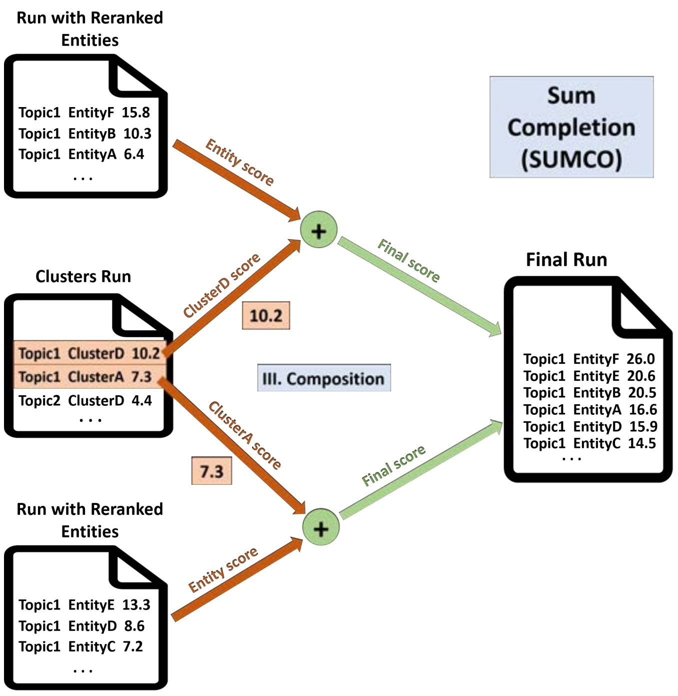


Figure 3.7: Third phase of Figure 3.5 in detail. From the reranked entities run of the clusterD we take the first 3 entities and for each one we sum its score with the score of clusterD on topic1. From the reranked entities run of the clusterA we take the first 3 entities and for each one we sum its score with the score of clusterA on topic1. After the processing of all the lines of clusters run corresponding to topic1, the entities are sorted by their new final score and then merged into the final run. The same process is repeated for each topic.

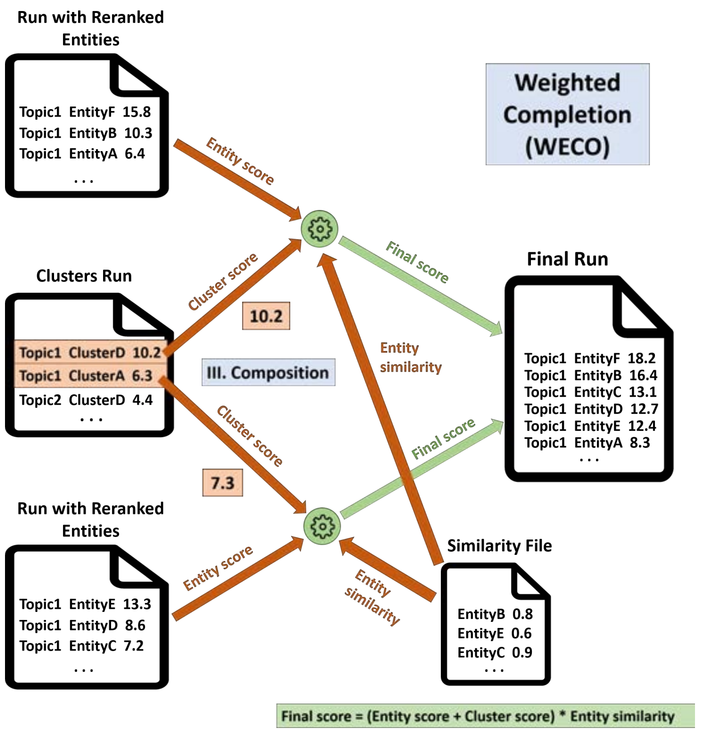


Figure 3.8: Third phase of Figure 3.5 in detail. From the reranked entities run of the clusterD we take the first 3 entities, for each entity we sum its score with the score of clusterD on topic1 and we multiply the result for the entity similarity. From the reranked entities run of the clusterA we take the first 3 entities, for each entity we sum its score with the score of clusterA on topic1. After the processing of all the lines of clusters run corresponding to topic1, the entities are sorted by their new final score and then merged in final run. The same process is repeated for each topic.

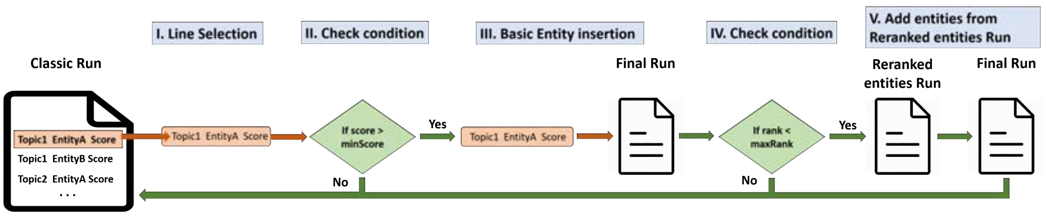


Figure 3.9: Example of a fusion process. In the phase I we take a line from the classic run file. For that line we check if $score > minScore$. If it is true we add to final run the line taken at point I and follow with point IV otherwise we restart from I. After III we check if $rank < maxRank$. If it is true we add other entities from the corresponding reranked entities run, otherwise we restart from I. After adding entities in point V we restart from point I.

with entities taken from reranked entities run. In this case we enrich all the entities with rank between 0 and *maxRank* with new entities. In line 5 and 6 we firstly retrieve the cluster corresponding to the classic entity and then we retrieve the reranked entities run of that cluster on classic topic. In line 7 we select the number of new entities we want to add to the final run and in line 8 we insert them. The values of *minScore*,*maxRank* and *maxEntity* are chosen through qualitative evaluation. We execute several versions of fusion algorithms with different values of *minScore*, *maxRank* and *maxEntity*, then we choose the best value looking at the evaluation results. The four fusion methods we implement rely on the same Algorithm described above, they only differ from the way in which they choose and merge entities from the reranked entities run into the final run. For the new entities insertion we use four different approaches:

1. *Simple fusion (SIFU)* (see Figure 3.10): we add a fixed number of entities taken from the corresponding reranked entities run. In this way we keep the ranking given by both the classical method and our reranking method.

2. *Sort fusion (SOFU)* (see Figure 3.11): we execute the same process described for SIFU with the difference that, before writing the entities into the final run, we collect all the entities of the same topic and sort them. The sorting is performed based on the score of each entity. In this way we keep only entities ranking losing classical ranking.

3. *1000 length fusion (LEFU)* (see Figure 3.12): we execute the same process described for SOFU with the difference that instead of checking if the score entity is greater than a threshold, we simply insert a total of 1000 entities in the final run for each topic.

4. *Sum fusion (SUMFU)* (see Figure 3.13): we execute the same process as described for SOFU with the difference that we use as the final score for the entity:

$$\text{finalScore} = \text{classicScore} + \text{entityRerankScore}$$

where *classicScore* is the score of the entity of the classic run and *entityRerankScore* is the score of the entity inside its reranked entities

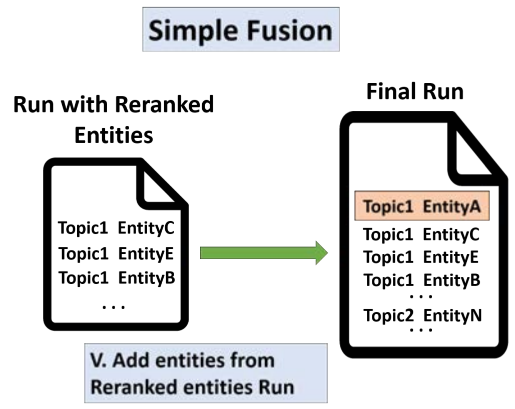


Figure 3.10: Fifth phase of Figure 3.9 in detail. From the reranked entities run of the cluster to which entityA belongs we take the first 3 entities and merge them into the final run. In this way the entities order of both classical and reranking methods is kept.

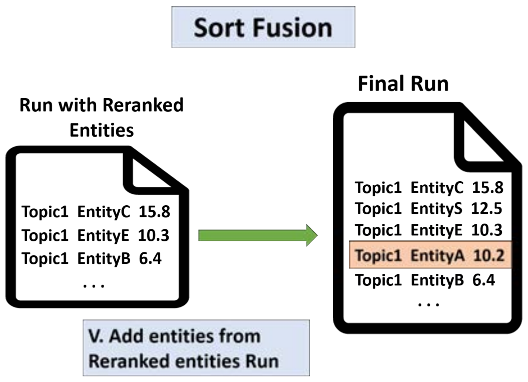


Figure 3.11: Fifth phase of Figure 3.9 in detail. From the reranked entities run of the cluster to which entityA belongs we take the first 3 entities. After the processing of all the lines of classic run corresponding to topic1, the entities are sorted by their score and then merged into the final run. The same process is repeated for each topic.

run.

We have to make two further considerations:

- During this phase we have also to process special characters inside each line due to a malfunction in the management of the encoding by Terrier.
- At the end of each process we check for duplicates inside the final run and in case of positive outcome we keep only the entity with higher score among all those identified.

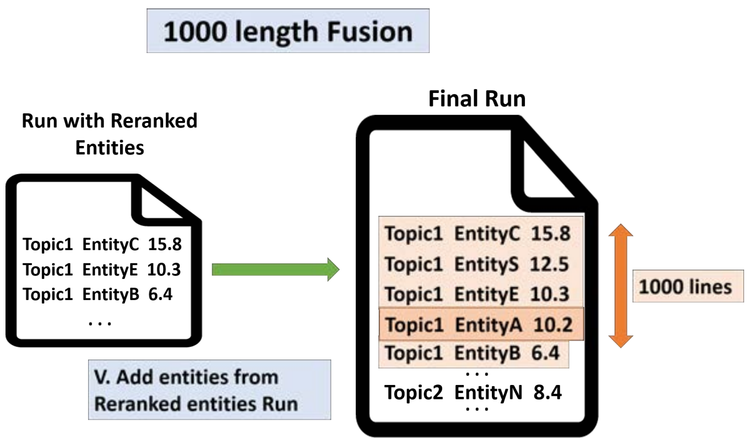


Figure 3.12: Fifth phase of Figure 3.9 in detail. From the reranked entities run of the cluster to which entityA belongs we take the first 3 entities. For each topic we select 1000 entities. After the processing of all the lines of classic run corresponding to topic1, the entities are sorted by their score and then merged into the final run. The same process is repeated for each topic.

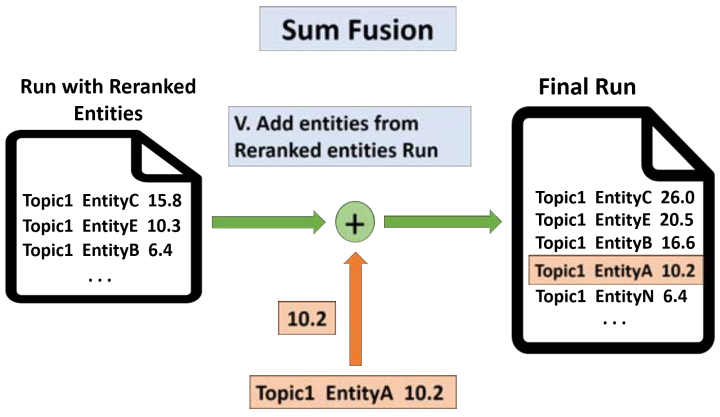


Figure 3.13: Fifth phase of Figure 3.9 in detail. From the reranked entities run of the cluster to which entityA belongs we take the first 3 entities and for each one we sum its score with the score of entityA on topic1 taken at phase I. After the processing of all the lines of classic run corresponding to topic1, the entities are sorted by their new final score and then merged in final run. The same process is repeated for each topic.

4 Evaluation

After the realization of our combination and fusion systems we evaluate the results achieved. In particular we do two types of evaluation: quantitative and qualitative (i.e. topic-based). In the quantitative evaluation we make some general considerations about the effectiveness of the proposed systems, considering measures as MAP, nDCG, P@5 and P@10. In the qualitative evaluation we look at how the effectiveness of each system is correlated to the topic (query) through a study on the form in which the it is formulated (with witch terms) and on the type and number of entities requested. In particular this evaluation can give us information about the coherence and the completeness of our virtual documents making a comparison between the ideal answer, the user would like to receive, and the real one. This analysis highlights also those problems related to the computation of the final score of each entity and their sorting. We report all the numerical values in details in the appendix in order to avoid cluttering and to increase readability.

4.1 Experimental setup

In this section we want to describe in detail the portion of dataset used, the queries, the relevance judgments and the process for documents creation.

4.1.1 Dataset

We consider the DBpedia dataset described in Section 2.2.3, in particular we use the English part of the 2015-10 version available at [14]. This dataset contains 6.2 million entities, 1.1 billion facts and an ontology of 739 types. The parts we have considered are:

1. *Infobox properties unredirected*: containing all the information described into Wikipedia infoboxes. This file allows us to reconstruct the connection between entities because it contains references to other Wikipedia pages that in our context represent entities.

2. *Labels*: containing all the titles associated to Wikipedia pages that in our context represent entities names.

3. *Short abstracts*: containing all the brief descriptions of the content of Wikipedia pages. These information allow us to better understand what is described in the entire page.

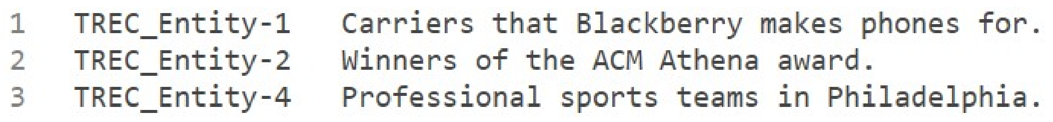
In particular we use files in Terse RDF Triple Language (turtle) format which allows us to express data represented through RDF data model.

In accordance with what was done by Hasibi et al. in [22], we take all the triples that have both a title and a short abstract (*label* and *comment* as predicates). To do this we create a database on which we firstly save the three files (infoboxes properties unredirected, labels and short abstracts) using three different tables composed all by three columns: subject, predicate and object; and subsequently we identify through SQL JOIN operations which entities have both a label and a short abstract. These entities are therefore stored in a separate table (nodes table) in which we associate each entity IRI with a numeric ID. In this way we reduce our dataset dimensions obtaining 4.6 millions of entities from the original 6 millions. As described in Section 2.2.2 we can see these triples as a RDF graph with 4.6 millions of nodes. For space and time constraints and computations limits we decide to further reduce our mini world of interest selecting a specific subgraph of the one just identified.

For the creation of this subgraph, we firstly take from the relevance judgments (*qrels*) provided by the DBpedia dataset those assessments relative to the TREC entity campaign. We choose this subset of relevance judgments because it is the smallest one, indeed it consists of 1575 assessments. The assessments are in the form:

texitCampaign-topic Q0 <dbpedia:Entity> Relevance_Judgment

and we extract, from our subset, all the entities present in the 3-rd column of qrels with any relevance judgment (0 - not relevant, 1 - partially relevant, 2 - highly relevant). To select from our database only these entities we must first perform string preprocessing in order to make the "<dbpedia:Entity>" string compatible with the entity name (IRI) "http://dbpedia.org/resource/Entity" stored in the node table. Once identified these entities in the DBpedia RDF graph, we exploit their relationships to expand our subgraph by adding another level of entities. We select the entities directly connected with the one in the qrels set. Using the example in Figure 3.2 we can suppose to have in our qrels the quadruple:



```
TREC_Entity-1   Carriers that Blackberry makes phones for.
TREC_Entity-2   Winners of the ACM Athena award.
TREC_Entity-4   Professional sports teams in Philadelphia.
```

Figure 4.1: Example of queries from the TREC entity campaign.

TREC_Entity-15 Q0 <dbpedia:Michael_Schumacher> 2

In this case we select the entity "Michael_Schumacher" as the entity of level 0 in the subgraph, then we expand this graph with the entities connected to it such as: "Benetton_Formula", "Ralf_Schumacher", "West_Germany" and "Scuderia_Ferrari". These new entities belong to the level 1 of our graph. The final subgraph is then constituted of these entities as nodes and their relationships as edges.

Someone could think that the choice of considering as a collection a subgraph of entities taken by the relevance judgments can be limiting because we rely on entities that we know are relevant for our topics but this is not true. This set of entities does not unduly affect the outcome of the retrieval process because we do not take just those entities that are marked as relevant but also those not relevant leading to a balanced subset of entities.

4.1.2 Topics and relevance judgments

As relevance judgments we use those provided by the DBpedia dataset that are relative to the TREC entity campaign. These assessments are 1575 and use as relevance weights for the entities: 0 for not relevant, 1 for partially relevant and 2 for highly relevant. As set of queries for the retrieval phase, we choose those corresponding to the relevance judgments just mentioned. These queries are 17, are expressed in natural language and focus on specific relationships between entities. An example of queries can be seen in Figure 4.1. These topics require entities without explicitly mentioning them, instead, they describe a specific relationship that uniquely identifies one or a set of target entities.:

```
<top>
<num><Topic_ID></num><title>
question
</title>
</top>
```

4.1.3 Documents creation

In this phase we create those documents necessary for the retrieval phase. In order to do this, as described in Section 3.3, we take one cluster at a time, we consider all the entities that belong to it and we take for each entity all its information contained in the database described in Section 4.1.1. Once obtained all the information, we merge them into a single virtual document.

This is the part with higher computational cost in terms of time because it requires access to the database and the execution of nested queries. These queries extract all information needed from the three tables described in Section 4.1.1, in particular we have to take the label, short abstract and infobox properties of the entities of our subgraph described in Section 4.1.1.

The generated document is serialized in TREC format:

```
<DOC>
<DOCNO><Cluster_ID></DOCNO>
text
</DOC>
```

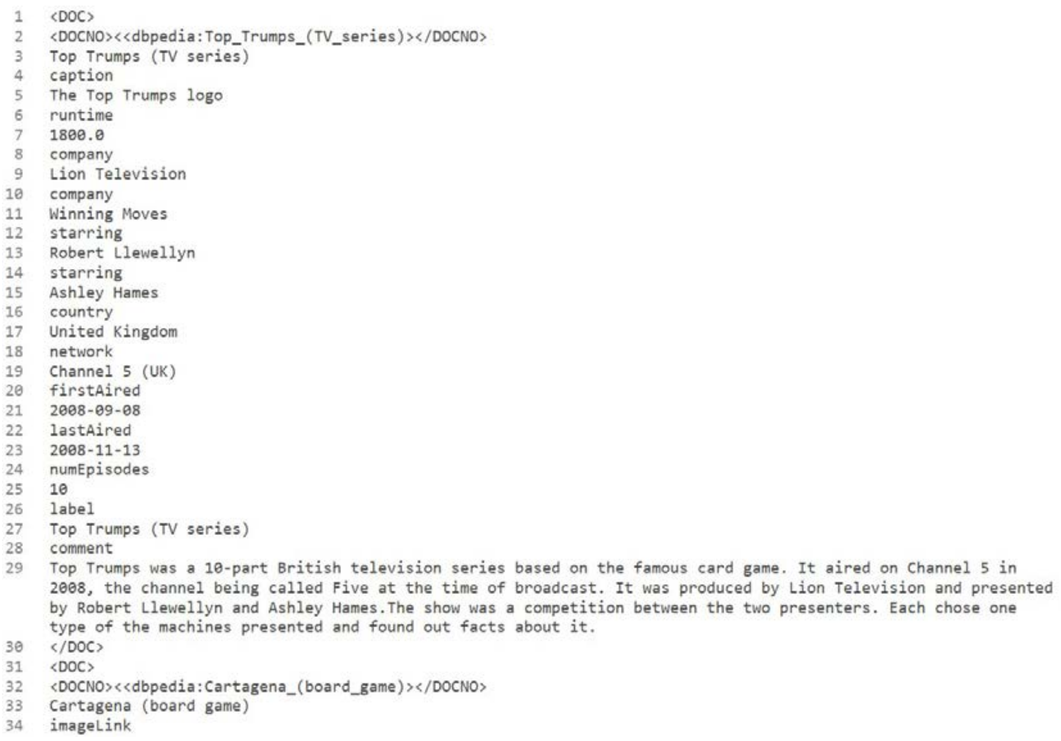
A real example can be seen in Figure 4.2.

4.2 Average evaluation

In Figure 4.3 and Figure 4.4 we represent through a box-and-whisker plot the values of MAP, average nDCG, average P@5 and average P@10 for the combination systems (SICO, SUMCO, WECO) and the fusion systems (SIFU, SOFU, LEFU, SUMFU). Each box in the two plots represents with its bottom line the value of the first quartile (namely the middle value between the smallest number and the median), with its top line the value of the third quartile (namely the middle value between the bigger value and the median), with the horizontal line inside the box the median. The two vertical lines (namely whiskers) connect the bottom line with the minimum and the top line with the maximum. Minimum and maximum are calculated through:

$$\text{minimum} = \text{Q3} + 1.5 * \text{IQR}$$
$$\text{maximum} = \text{Q1} - 1.5 * \text{IQR}$$

where Q3 and Q1 are respectively the third and the first quartile, IQR is the interquantile range (namely the difference between the third and the first



```
<DOC>
<DOCNO><<dbpedia:Top_Trumps_(TV_series)></DOCNO>
Top Trumps (TV series)
caption
The Top Trumps logo
runtime
1800.0
company
Lion Television
company
Winning Moves
starring
Robert Llewellyn
starring
Ashley Hames
country
United Kingdom
network
Channel 5 (UK)
firstAired
2008-09-08
lastAired
2008-11-13
numEpisodes
10
label
Top Trumps (TV series)
comment
Top Trumps was a 10-part British television series based on the famous card game. It aired on Channel 5 in
2008, the channel being called Five at the time of broadcast. It was produced by Lion Television and presented
by Robert Llewellyn and Ashley Hames.The show was a competition between the two presenters. Each chose one
type of the machines presented and found out facts about it.
</DOC>
<DOC>
<DOCNO><<dbpedia:Cartagena_(board_game)></DOCNO>
Cartagena (board game)
imageLink
```

Figure 4.2: Representation of the document associated to cluster 0. We can observe the first entity "Top Trumps (TV series)" with all its information in the following lines, in particular label "Top Trumps (TV series)" and comment "Top Trumps was a 10-part British television series based on the famous card game. It aired on Channel...". We can also observe the second entity of the cluster "Cartagena (board game)" concatenated after the first one.

quartile). From this type of plot we can have information on the dispersion of the data because the more the box and the whiskers are symmetric the more the probability distribution is symmetric. Considering a single box, the points distributed in vertical on its central axis represent the single value obtained for each query, namely inner points. If a point comes out from the whiskers it is called *outlier*.

From the representations of the inner points in Figure 4.3, we can notice that, while most of them are contained in the box, there are some topics (outliers) where the systems obtain very good results. In Figure 4.3 we can also observe that BM25 obtains better results in all these measures even though the performance are comparable on average. For what concern our systems we can notice that SUMCO (in gray) proved to be the better one obtaining better results in all the measures, WECO (in yellow) follows and SICO (in blue) is the worst one. From these results we can deduce that the final score used for WECO is not optimal for our retrieval purpose while the one of SUMCO works better. In general we can also said that ordering the reranked entities by score before writing them into the final run, as done by SUMCO and WECO, allow us to obtain a better ranked list with respect to the SICO method where this is not done. This can be justified by the fact that in SICO we create the final run by progressively inserting *maxEntity* entities for each cluster, maintaining their original ordering. However, it may be that the next cluster contains some entities that are more relevant than thus of the previous one; in this case these entities would be shifted lower than their ideal position leading to worse performance.

In order to improve the effectiveness of combination systems we implement the fusion systems as described in Section 3.5. Comparing Figure 4.3 with Figure 4.4 we can see that fusion systems obtain better results with respect to combination systems because we exploit BM25 run, that we know having better performance, as basis for our approaches. Looking at Figure 4.4 in detail we can observe that LEFU (in gray) turns out to be the best method for MAP and average nDCG measures, while it obtains the same performance of BM25 in average P@5 and average P@10. This behavior can be due to the choice of which entities to insert into the final run of LEFU. Resuming what described in Section 3.5, the LEFU system selects, for each entity of the classic retrieval run, a fixed number of entities from the reranked entities run of the cluster to which the classic entity belongs. In particular this system repeats

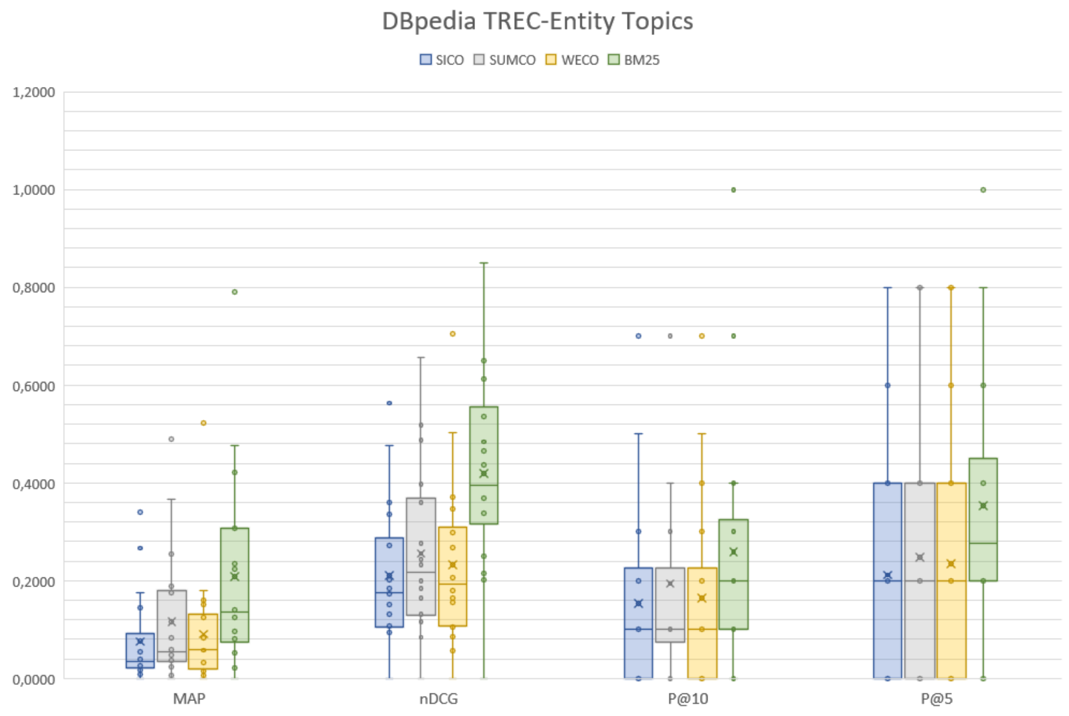


Figure 4.3: Boxplot of the MAP, average nDCG, average P@5 and average P@10 values for the combination systems. The bottom line of each box represents the first quartile, the top line represents the third quartile and the horizontal line inside the box the median. The points distributed on each system vertical axis are the specific values for each topic. We can see that BM25 outperforms our methods in all measures. SUMCO is the best systems among our approaches and SICO the worst one. The values are reported in Table A1.1 of the Appendix.

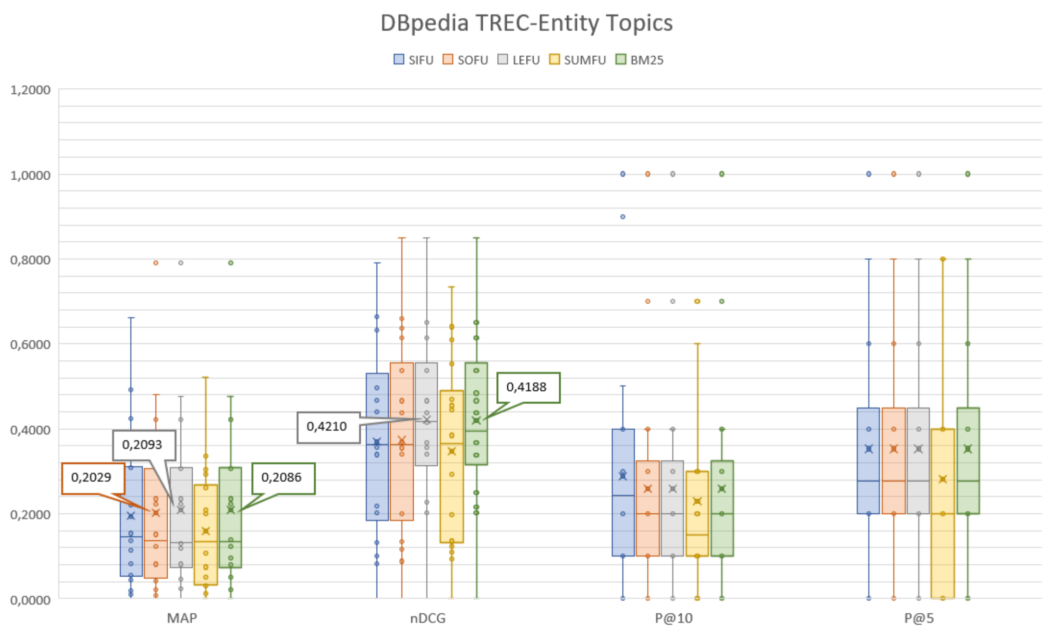


Figure 4.4: Boxplot of the MAP, average nDCG, average P@5 and average P@10 values for the fusion systems. The bottom line of each box represents the first quartile, the top line represents the third quartile and the horizontal line inside the box the median. The points distributed on each system vertical axis are the specific values for each topic. We can see that for MAP and nDCG the LEFU systems is the better one; for P@10 SIFU is the better one. The values are reported in Table A1.2 of the Appendix.

this approach until we insert 1000 entities for each topic. Let us consider the case in which:

- there is a partially relevant entity in the classic run;
- the entity above belongs to a cluster in which there is also a relevant entity.

With LEFU we obtain the partially relevant entity, we access the reranked entities run of the corresponding cluster and take the relevant entity in order to add it to the final run. With the other three methods this is not possible if the partially relevant entity has a score lower than the *minScore* threshold. In this case we do not access the corresponding cluster and thus we do not insert the relevant entity in the final run.

As we have seen in Section 3.5, in the fusion systems we select a threshold called *maxEntity* that defines the number of entities, from the reranked entities run, we want to add into the final run for each entity of the classic run. We execute some tests adding different number of entities in the fusion process

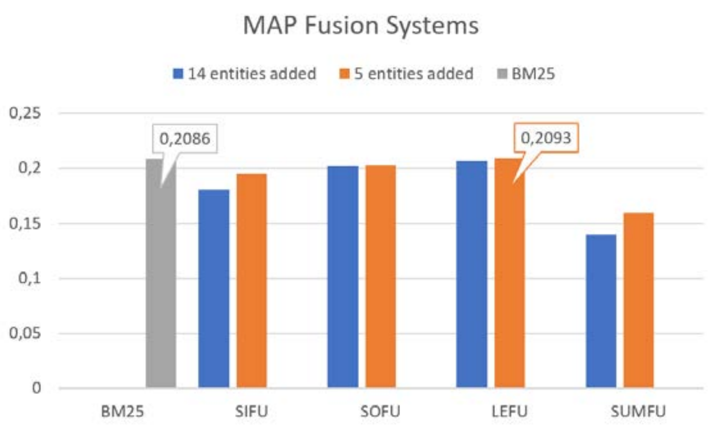


Figure 4.5: Representation of MAP for fusion systems with the addition of 14 entities or 5 entities and representation of MAP for BM25. In all the systems the addition of 5 entities turns out to be better. We can see that our LEFU method, with the addition of 5 entities, has the best performance. The values are reported in Table A1.3 of the Appendix.

in order to select the best value of $maxEntity$. In particular, we represent in Figures from 4.5 to 4.8 the MAP, average nDCG, average P@5 and average P@10 values obtained with the addition of 14 entities and 5 entities. We can see that in almost all cases adding a small number of entities turns out to be better.

An explanation to this behavior can be given through two considerations. The first consideration is that adding a big number of entities correspond into the addition of a little number of relevant ones and a big number of non relevant ones (noise). In fact the inclusion of entities from cluster run into the final run often leads to a shift of the relevant entities retrieved by BM25 in positions with a lower rank, worsening the performance of the system.

The second consideration is on the relevance judgments creation through the pooling technique. This technique relies on the judgment of a portion of retrieved entities, the ones that constitutes the pool. The pool is constituted by the retrieved entities that belong to the top-k rank of the runs retrieved by classic systems. Many entities that can be considered relevant are not considered in the pooling process and therefore result non relevant at the end of the evaluation process even though they may be. In particular the DBpedia dataset contains a big number of entities that are very similar one to each other, as the ones shown in Figure 4.9. In this Figure we can observe that in DBpedia we have a single different entity for each football team belonging to a

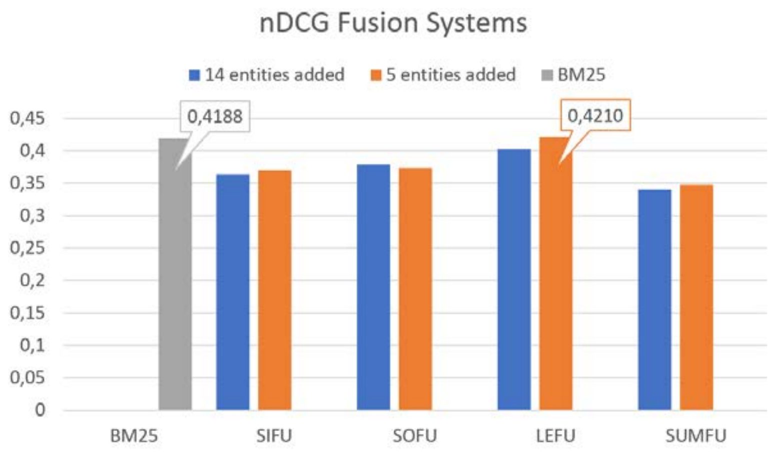


Figure 4.6: Representation of average nDCG for fusion systems with the addition of 14 entities or 5 entities and representation of average nDCG for BM25. In all systems, except for SOFU, the addition of 5 entities turns out to be better. We can see that our LEFU method, with the addition of 5 entities, has the best performance. The values are reported in Table A1.3 of the Appendix.

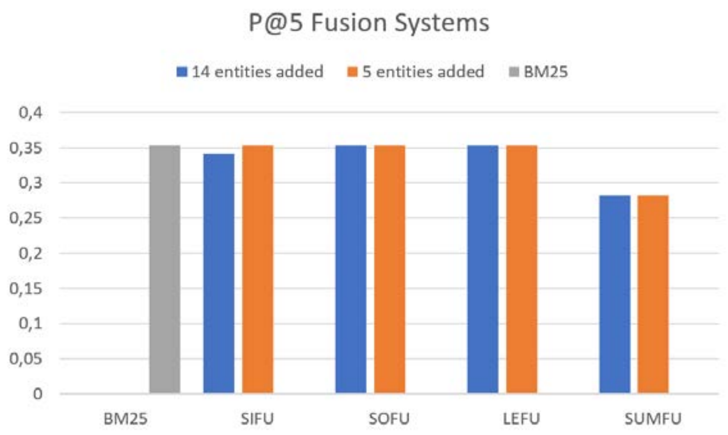


Figure 4.7: Representation of average P@5 for fusion systems with the addition of 14 entities or 5 entities and representation of average P@5 for BM25. In all systems, except for SIFU, we obtain the same performance for both the addition of 14 and 5 entities. The values are reported in Table A1.3 of the Appendix.

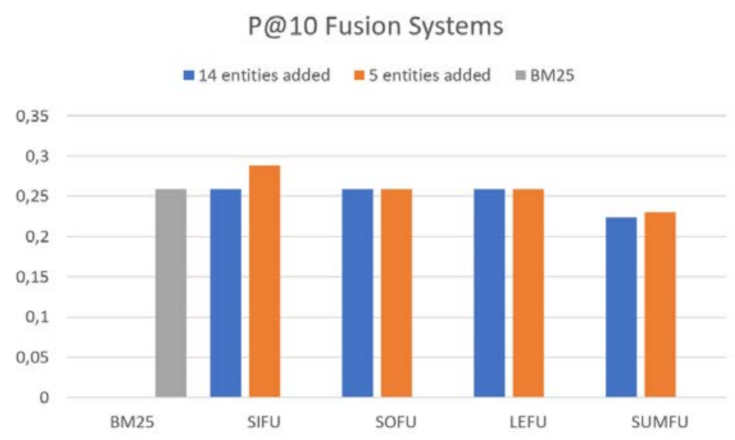


Figure 4.8: Representation of average P@10 for fusion systems with the addition of 14 entities or 5 entities and representation of average P@10 for BM25. In SOFU and LEFU we obtain the same results both adding 14 or 5 entities. SUMFU obtains the worst results while SIFU with 5 entities obtains the better result. The values are reported in Table A1.3 of the Appendix.

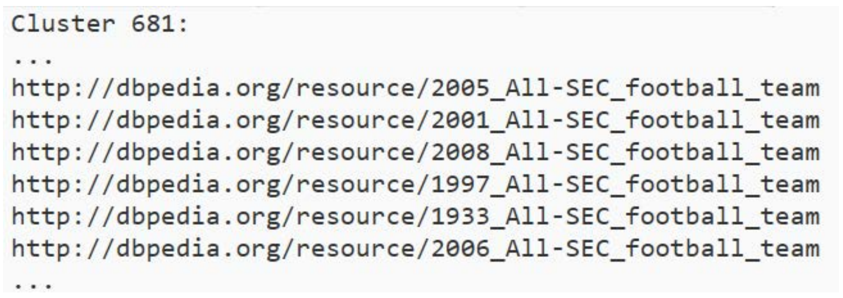

```
Cluster 681:
...
http://dbpedia.org/resource/2005_All-SEC_football_team
http://dbpedia.org/resource/2001_All-SEC_football_team
http://dbpedia.org/resource/2008_All-SEC_football_team
http://dbpedia.org/resource/1997_All-SEC_football_team
http://dbpedia.org/resource/1933_All-SEC_football_team
http://dbpedia.org/resource/2006_All-SEC_football_team
...
```

Figure 4.9: Example of similar entities present in DBpedia. In particular these entities are taken from cluster 681.

specific year. In this case only few of these entities have been considered during pooling and then judged, even though we maybe want to consider as relevant also the other teams of other years. As a real example of this consideration, in Figure 4.18 we can observe that only a small subset of the entities in Figure 4.9 have been judged in the pooling process. One way to overcome this problem can be to consider these entities like as one, creating a new unified entity that contains all the relevant information. This approach however could not be used without changing even the relevance judgment because the new created entities would not have an equivalent in the assessed entities. We therefore deduce that it would be necessary also to aggregate the same entities in the relevance judgment leading to a complete reconstruction of the set.

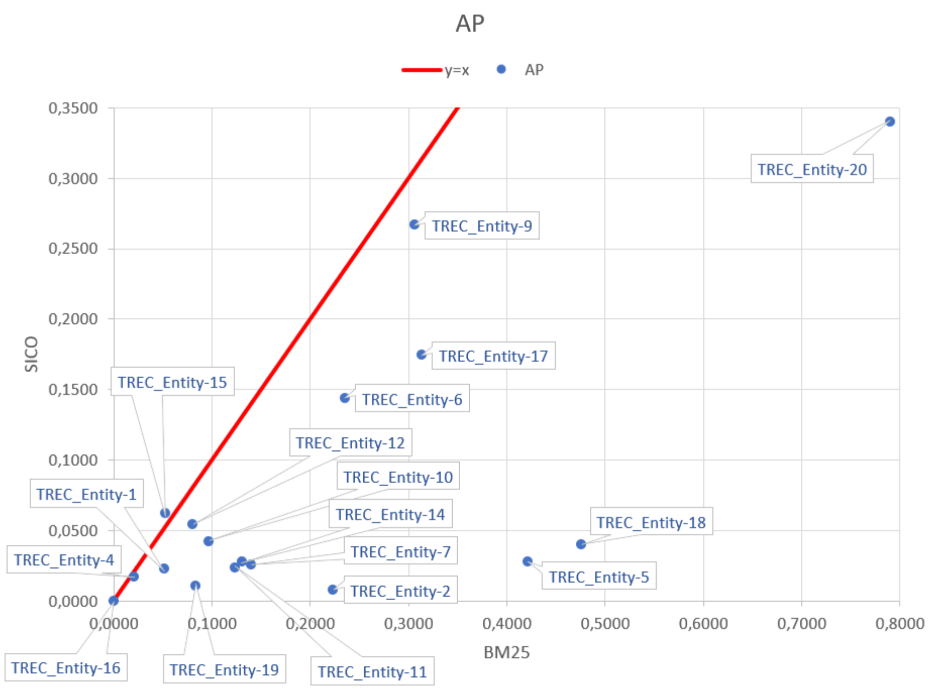


Figure 4.10: Representation of AP for SICO and BM25. Every point represents the AP value of a specific topic. We can see that the only topic in which our system performs better is TREC_Entity-15. The AP values are reported in the Tables from A1.4 to A1.20 of the Appendix.

4.3 Topic-based evaluation

In order to execute a more specific study on our systems results, we perform a topic-based evaluation. In this evaluation we can observe some interesting aspects of the retrieval in relation to the topic on which it is performed. In Figures from 4.10 to 4.12 we can see a representation of the AP measure obtained for each topic in the case of combination systems and BM25. We can observe that, in mostly of the topics, BM25 obtains better results than our methods, but there are still some topics in which we perform better such as TREC_Entity-15 with SICO and WECO, TREC_Entity-9 with SUMCO and TREC_Entity-1 with WECO.

The same study has been done for fusion systems which results are represented in Figures from 4.13 to 4.16. In these cases we have that our systems obtain results very close to the ones of BM25 for most of the topics with the exception of TREC_Entity-11, TREC_Entity-14 and TREC_Entity-18 for SIFU: TREC_Entity-14 for SOFU; TREC_Entity-12, TREC_Entity-14 and TREC_Entity-6 for SUMFU and TREC_Entity-10 for LEFU. Even if

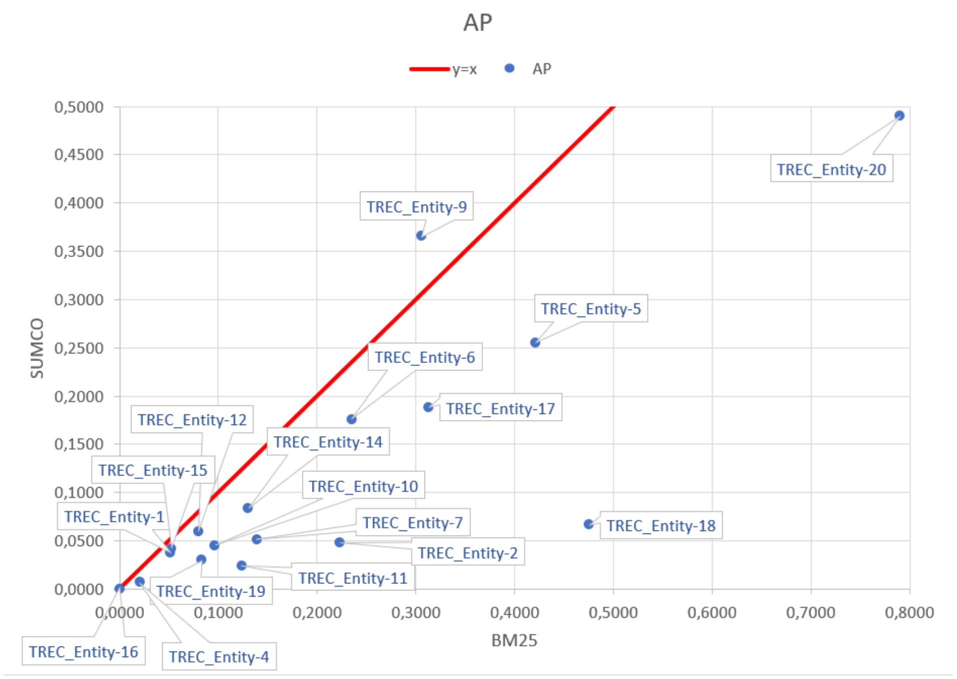


Figure 4.11: Representation of AP for SUMCO and BM25. Every point represents the AP value of a specific topic. We can see that the only topic in which our system performs better is TREC_Entity-9. The AP values are reported in the Tables from A1.4 to A1.20 of the Appendix.

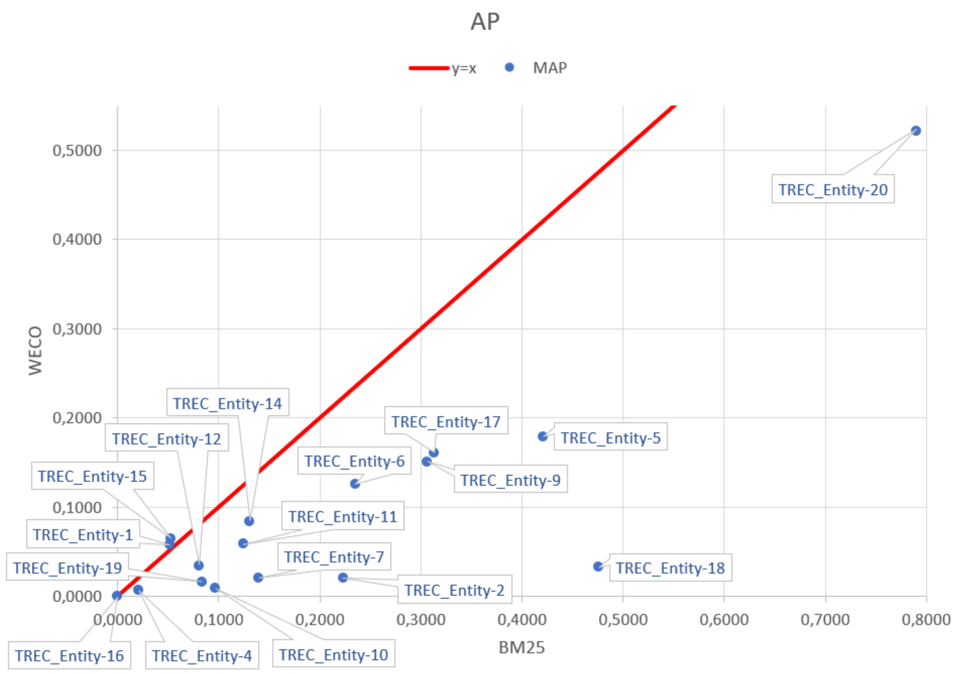


Figure 4.12: Representation of AP for WECO and BM25. Every point represents the AP value of a specific topic. We can see that the only two topics in which our system performs better are TREC_Entity-1 and TREC_Entity-15. The AP values are reported in the Tables from A1.4 to A1.20 of the Appendix.

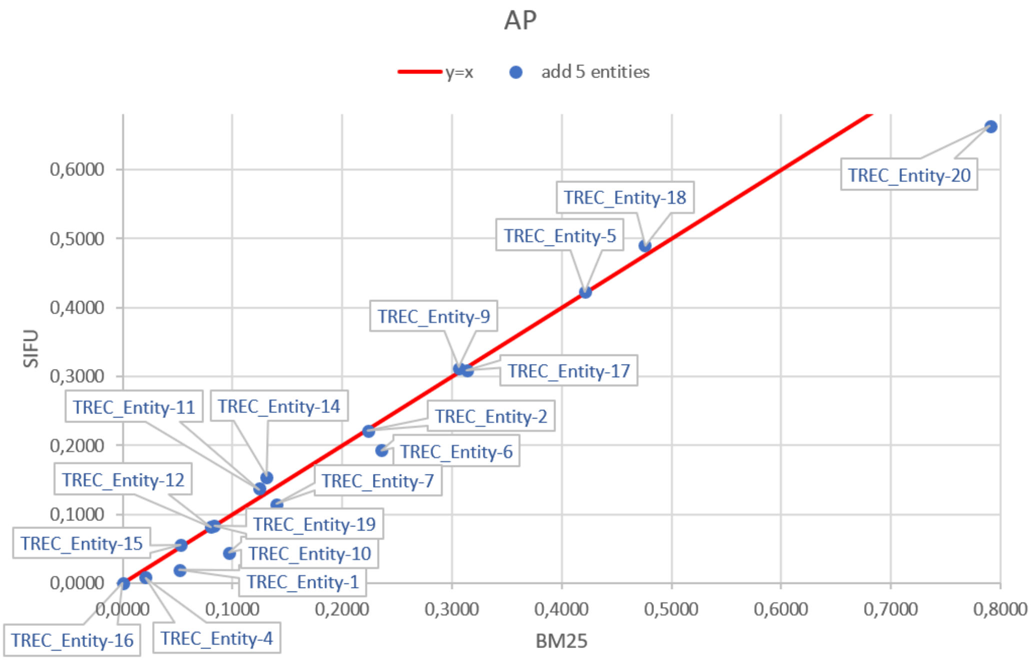


Figure 4.13: Representation of AP for SIFU and BM25. Every point represents the AP value of a specific topic. The AP values are reported in the Tables from A1.21 to A1.37 of the Appendix.

SUMFU results to be the worst system in average measures we can see that it performs well for the three specific topic mentioned above, obtaining results that distance themselves from those of BM25 more than the ones of the other methods.

As described in the Section above it may occur that some of the entities retrieved by our systems were not retrieved in the pooling process and therefore result not relevant for a topic. In Figure 4.17 we can see a representation obtained for TREC_Entity-15 topic. This topic ask for "*Universities that are members of the SEC conference for football.*" and it is an example in which we find the situation presented above. For this query we can see that the relevant entities from the relevance judgments are the ones shown in Figure 4.18 but it reasonable to say that even entities like the ones in Figure 4.9 should be considered as relevant. The run returned by SUMFU can be seen in Figure 4.19 where we can notice that entities like:

<dbpedia:2007_All-SEC_football_team>

are retrieved but not considered relevant and therefore this worsens the performance of SUMFU. Making a comparison with the run returned by BM25,

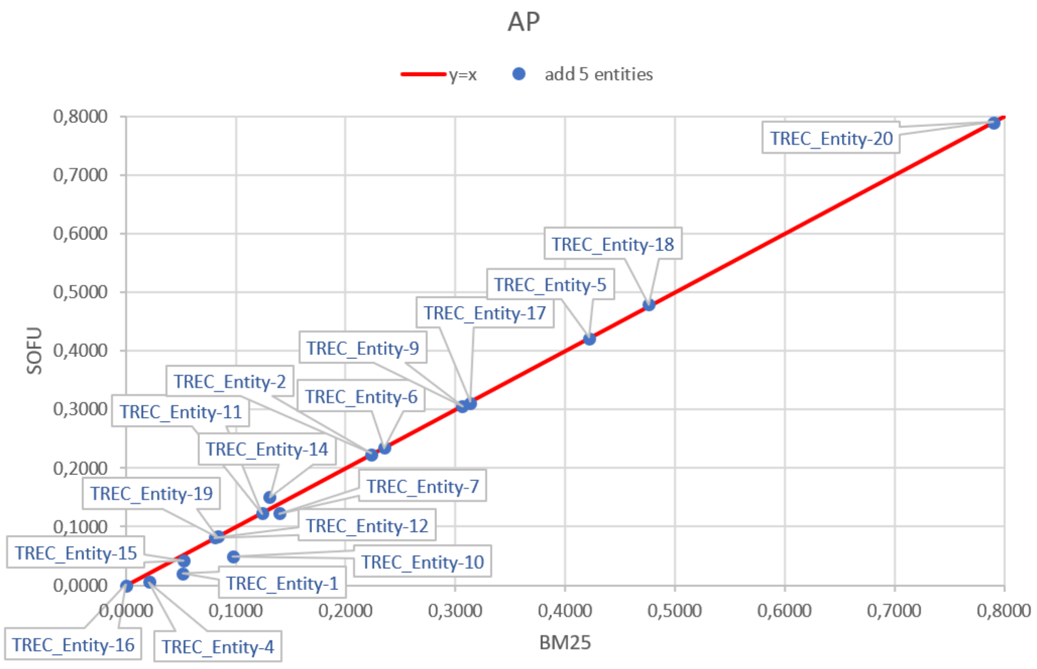


Figure 4.14: Representation of AP for SOFU and BM25. Every point represents the AP value of a specific topic. The AP values are reported in the Tables from A1.21 to A1.37 of the Appendix.

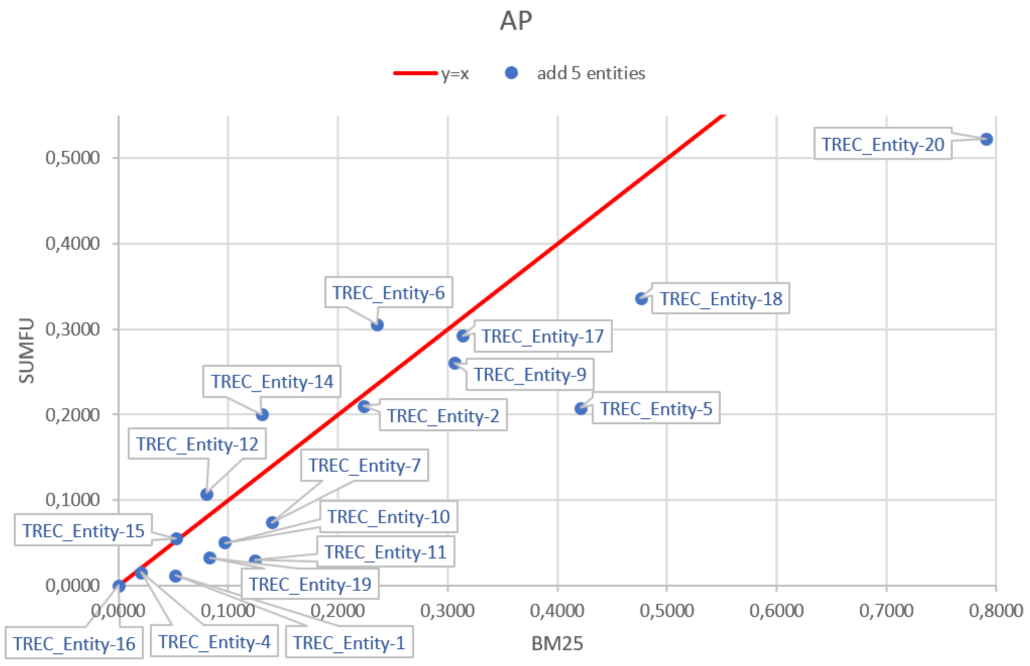


Figure 4.15: Representation of AP for SUMFU and BM25. Every point represents the AP value of a specific topic. The AP values are reported in the Tables from A1.21 to A1.37 of the Appendix.

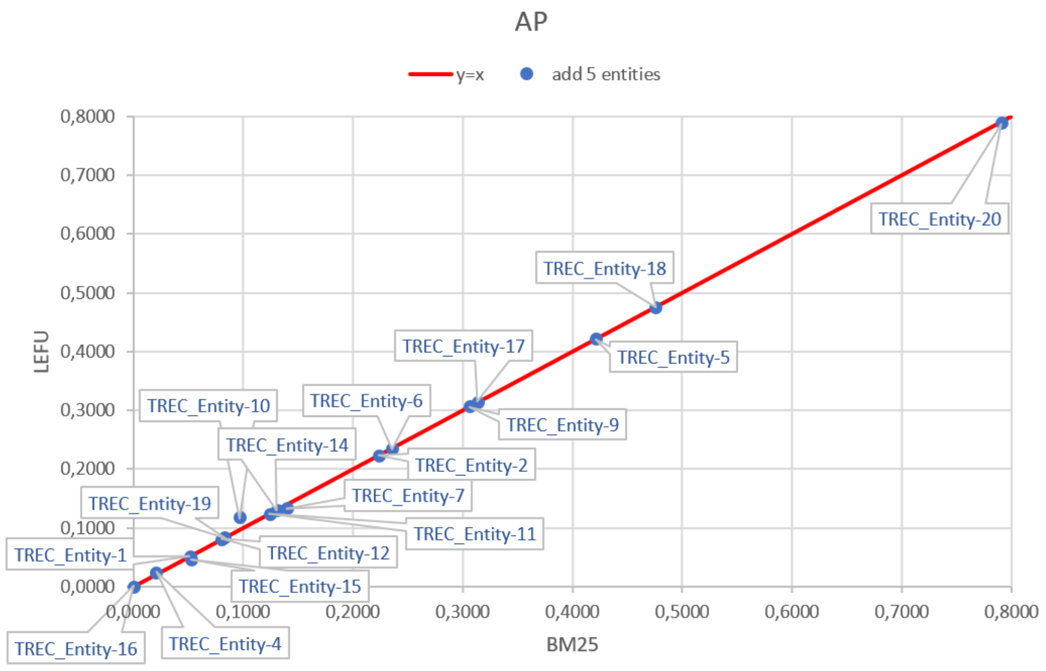


Figure 4.16: Representation of AP for LEFU and BM25. Every point represents the AP value of a specific topic. We can see that our system obtains the same results of BM25 for most of the topics with the exception of the topic 10 in which we obtain better results. The AP values are reported in the Tables from A1.21 to A1.37 of the Appendix.

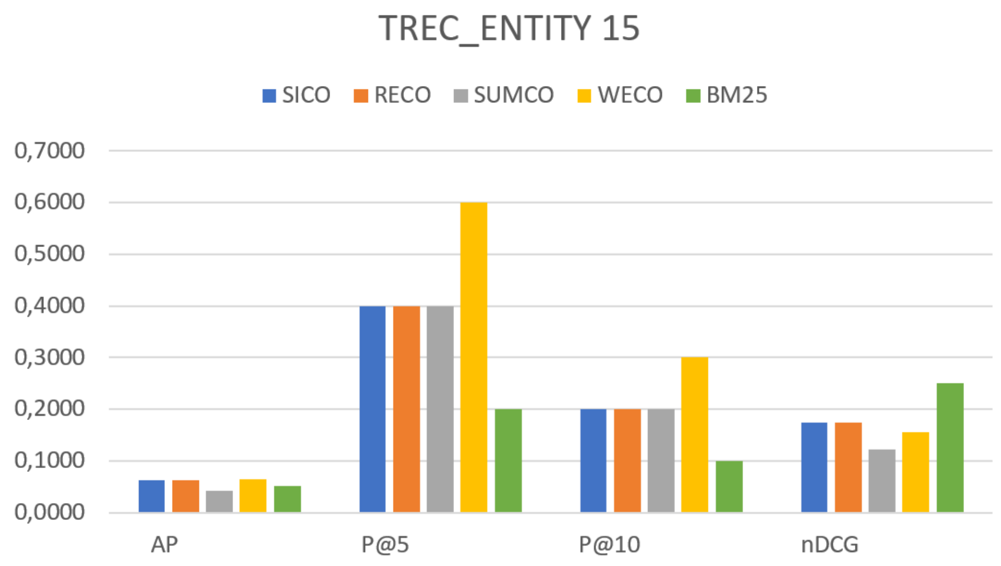
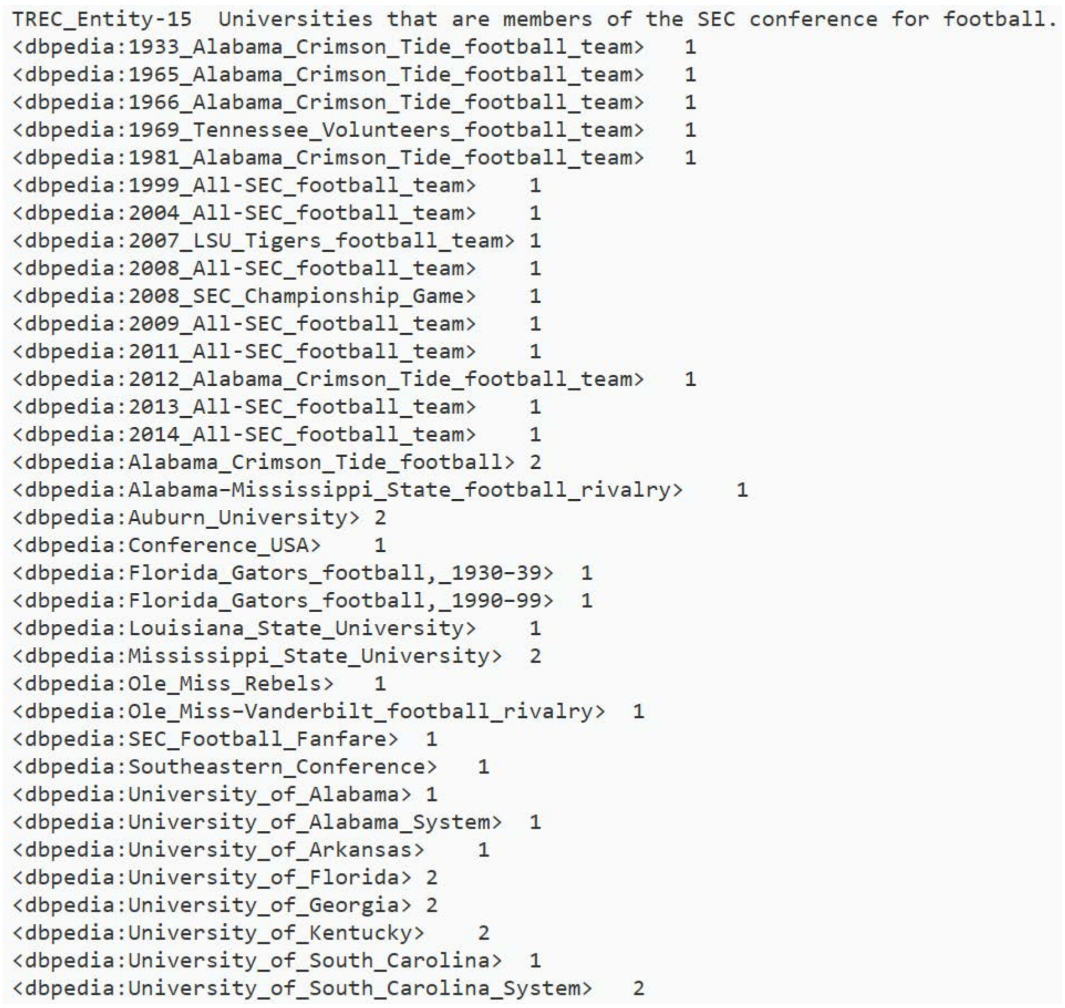


Figure 4.17: Representation of AP, P@5, P@10 and nDCG for combination systems for topic 15. We can see that we obtain good results in P@5 and in P@10. Our systems perform better also in AP while BM25 performs better in nDCG. The AP values are reported in the Table A1.15 of the Appendix.

shown in Figure 4.20, we can see that it retrieved only one relevant entity in the top 15 ones but it also retrieved many entities that could be considered as relevant.

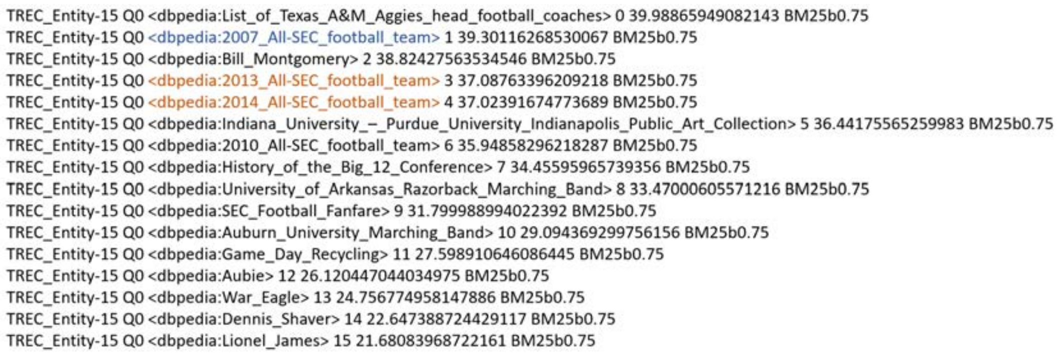
Continuing the analysis of individual topics, from Figure 4.21 and Figure 4.22 we can see that for TREC_Entity-16 the results obtained from all the systems are equal to zero. This happens because the relevant entities for query: "*Sponsors of the Mancuso quilt festivals.*" are just two: "*Brother_Industries*" and "*Janome*" (see Figure 4.23). Therefore it results very difficult for our systems to retrieve those specific entities among all the others in the collection. This task is made even more difficult by the fact that in "*Janome*" and "*Brother_Industries*" documents there is no reference to any of the words of the query and therefore systems such as BM25, that works on words matching, fails to retrieve the relevant entities. For TREC_Entity-16 not even our methods succeed in retrieval because both "*Janome*" and "*Brother_Industries*" entities are contained into two clusters that rely on "*Companies*" scope and therefore do not contain references to "*quilt festival*" which turns out to be a topic weakly connected to ours.

Other considerations can be made for TREC_Entity-4 whose performance are shown in Figure 4.24 for combination systems. We can see that in particular



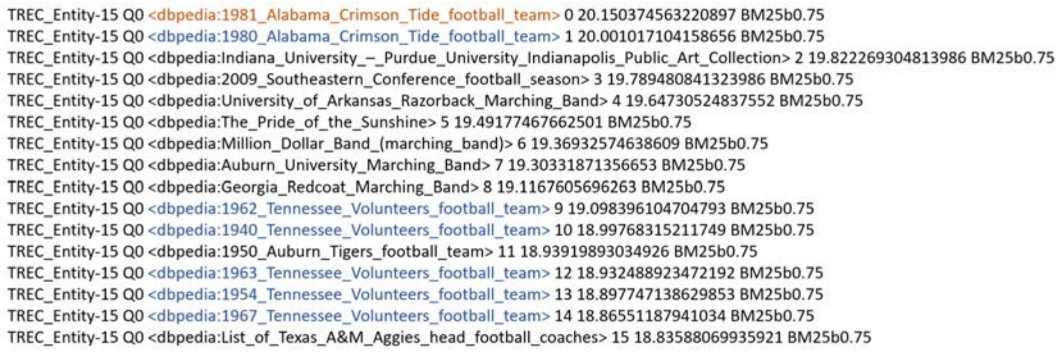
```
TREC_Entity-15  Universities that are members of the SEC conference for football.
<dbpedia:1933_Alabama_Crimson_Tide_football_team>   1
<dbpedia:1965_Alabama_Crimson_Tide_football_team>   1
<dbpedia:1966_Alabama_Crimson_Tide_football_team>   1
<dbpedia:1969_Tennessee_Volunteers_football_team>   1
<dbpedia:1981_Alabama_Crimson_Tide_football_team>   1
<dbpedia:1999_All-SEC_football_team>    1
<dbpedia:2004_All-SEC_football_team>    1
<dbpedia:2007_LSU_Tigers_football_team> 1
<dbpedia:2008_All-SEC_football_team>    1
<dbpedia:2008_SEC_Championship_Game>    1
<dbpedia:2009_All-SEC_football_team>    1
<dbpedia:2011_All-SEC_football_team>    1
<dbpedia:2012_Alabama_Crimson_Tide_football_team>   1
<dbpedia:2013_All-SEC_football_team>    1
<dbpedia:2014_All-SEC_football_team>    1
<dbpedia:Alabama_Crimson_Tide_football> 2
<dbpedia:Alabama-Mississippi_State_football_rivalry>    1
<dbpedia:Auburn_University> 2
<dbpedia:Conference_USA>    1
<dbpedia:Florida_Gators_football,_1930-39>  1
<dbpedia:Florida_Gators_football,_1990-99>  1
<dbpedia:Louisiana_State_University>    1
<dbpedia:Mississippi_State_University>  2
<dbpedia:Ole_Miss_Rebels>   1
<dbpedia:Ole_Miss-Vanderbilt_football_rivalry>  1
<dbpedia:SEC_Football_Fanfare>  1
<dbpedia:Southeastern_Conference>   1
<dbpedia:University_of_Alabama> 1
<dbpedia:University_of_Alabama_System>  1
<dbpedia:University_of_Arkansas>    1
<dbpedia:University_of_Florida> 2
<dbpedia:University_of_Georgia> 2
<dbpedia:University_of_Kentucky>    2
<dbpedia:University_of_South_Carolina>  1
<dbpedia:University_of_South_Carolina_System>   2
```

Figure 4.18: Portion of the relevance assessments of the topic 15. In particular it shows only the name and weight of relevant entities.



```
TREC_Entity-15 Q0 <dbpedia:List_of_Texas_A&M_Aggies_head_football_coaches> 0 39.98865949082143 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:2007_All-SEC_football_team> 1 39.30116268530067 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Bill_Montgomery> 2 38.82427563534546 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:2013_All-SEC_football_team> 3 37.08763396209218 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:2014_All-SEC_football_team> 4 37.02391674773689 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Indiana_University_–_Purdue_University_Indianapolis_Public_Art_Collection> 5 36.44175565259983 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:2010_All-SEC_football_team> 6 35.94858296218287 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:History_of_the_Big_12_Conference> 7 34.45595965739356 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:University_of_Arkansas_Razorback_Marching_Band> 8 33.47000605571216 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:SEC_Football_Fanfare> 9 31.799988994022392 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Auburn_University_Marching_Band> 10 29.094369299756156 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Game_Day_Recycling> 11 27.598910646086445 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Aubie> 12 26.120447044034975 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:War_Eagle> 13 24.756774958147886 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Dennis_Shaver> 14 22.647388724429117 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Lionel_James> 15 21.68083968722161 BM25b0.75
```

Figure 4.19: Top 15 entities retrieved from the SUMFU system for the topic 15. In orange we highlighted the entities marked as relevant in the relevance assessments, in blue an entity that could be considered relevant but that it is not.



```
TREC_Entity-15 Q0 <dbpedia:1981_Alabama_Crimson_Tide_football_team> 0 20.150374563220897 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1980_Alabama_Crimson_Tide_football_team> 1 20.001017104158656 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Indiana_University_–_Purdue_University_Indianapolis_Public_Art_Collection> 2 19.822269304813986 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:2009_Southeastern_Conference_football_season> 3 19.789480841323986 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:University_of_Arkansas_Razorback_Marching_Band> 4 19.64730524837552 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:The_Pride_of_the_Sunshine> 5 19.49177467662501 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Million_Dollar_Band_(marching_band)> 6 19.36932574638609 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Auburn_University_Marching_Band> 7 19.30331871356653 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:Georgia_Redcoat_Marching_Band> 8 19.1167605696263 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1962_Tennessee_Volunteers_football_team> 9 19.098396104704793 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1940_Tennessee_Volunteers_football_team> 10 18.99768315211749 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1950_Auburn_Tigers_football_team> 11 18.93919893034926 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1963_Tennessee_Volunteers_football_team> 12 18.932488923472192 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1954_Tennessee_Volunteers_football_team> 13 18.897747138629853 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:1967_Tennessee_Volunteers_football_team> 14 18.86551187941034 BM25b0.75
TREC_Entity-15 Q0 <dbpedia:List_of_Texas_A&M_Aggies_head_football_coaches> 15 18.83588069935921 BM25b0.75
```

Figure 4.20: Top 15 entities retrieved from the BM25 system for the topic 15. In orange we highlighted the entities marked as relevant in the relevance assessments, in blue the entities that could be considered relevant but that they are not.

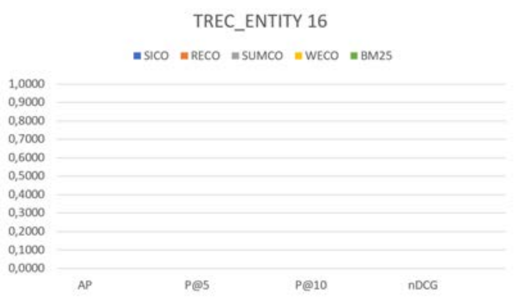


Figure 4.21: Representation of AP, P@5, P@10 and nDCG for combination systems for topic 16. The values are reported in Table A1.16 of the Appendix, but in this case we can see that all the systems do not achieve any result in all measures.

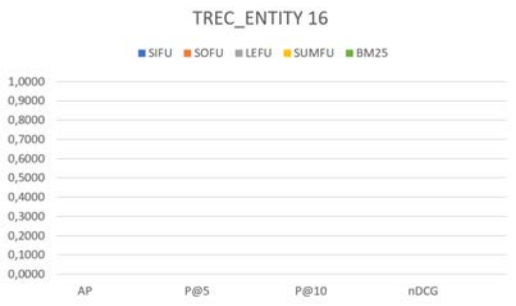
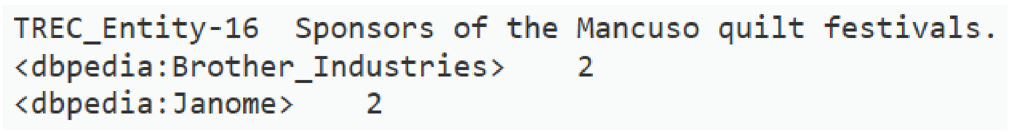


Figure 4.22: Representation of AP, P@5, P@10 and nDCG for fusion systems for topic 16. The values are reported in Table A1.33 of the Appendix, but in this case we can see that all the systems do not achieve any result in all measures.



```
TREC_Entity-16  Sponsors of the Mancuso quilt festivals.
<dbpedia:Brother_Industries>    2
<dbpedia:Janome>    2
```

Figure 4.23: Portion of the relevance assessments of the topic 15. In particular it shows only the name and weight of relevant entities.

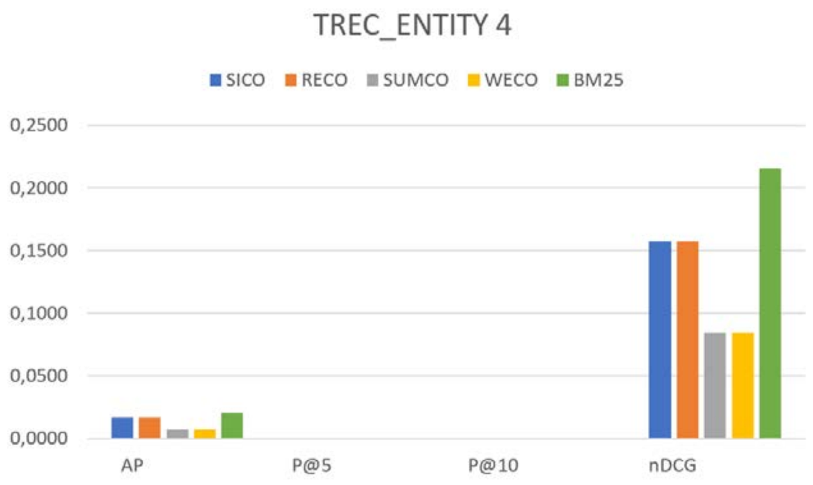


Figure 4.24: Representation of AP, P@5, P@10 and nDCG for combination systems for topic 4. We can see that all the systems obtain zero results in P@5 and P@10, while BM25 obtains the best nDCG result. The values are reported in Table A1.6 of the Appendix.

for AP, P@5 and P@10, we obtain very poor results. This could be due to the fact that the query "*Professional sports teams in Philadelphia.*" is very general and therefore it becomes difficult for our systems to understand whose are the entities to retrieve. In studying this topic we can also observe that, for how the fusion process is structured, our systems highly relies on BM25 ranked list and therefore our fusion runs present a high number of entities which contain both the word "*sport*" and "*Philadelphia*" when the information need of the user should be mainly focused on "*teams*".

Observing Figure 4.25 for TREC_Entity-17 we can say that SUMFU method obtains very good results in measures P@5 and P@10, and obtains comparable results in AP and nDCG. Looking at the run in Figure 4.26 in detail, we observe that there are many relevant entities (in orange) such as "*Sugar_Rush_(Food_Network)*", "*Duff_Goldman*" and "*Aarti_Sequeira*", that thanks to our clustering approach are not only retrieved but also ranked in high positions. In blue we can see also the entities "*Richard_Blais*" and "*Michael_Allemeier*", that are not judged as relevant in the pooling phase but that are anyway relevant for this query. Making a comparison with the run obtained through BM25 shown in Figure 4.27, we can notice that with our fusion methods we retrieved many relevant entities (in green) such as "*Nadia_Giosia*", "*Chef_at_Home*", "*Martie_Duncan*" and "*The_Food_Network_Adward*", that are not present in the BM25 run while

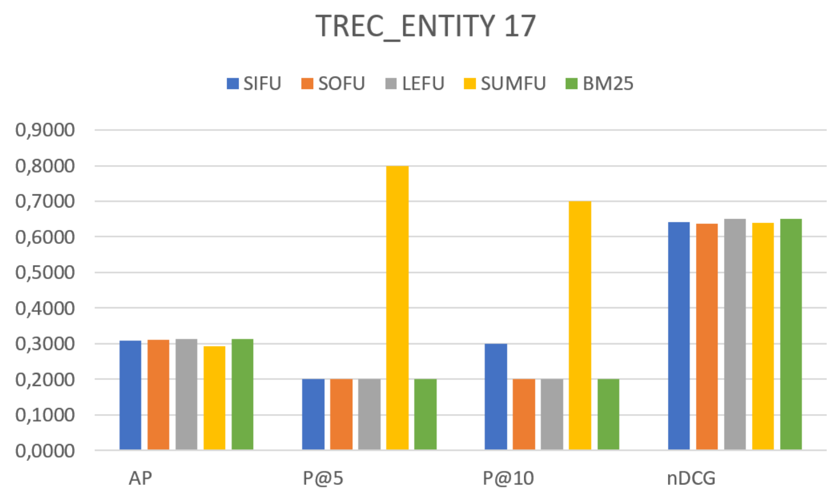
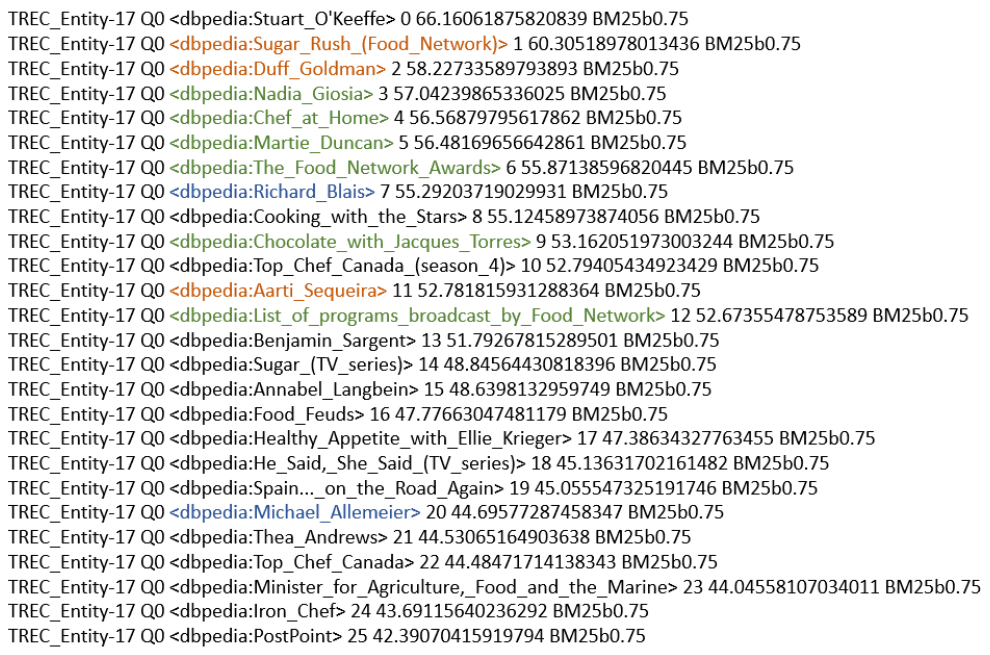


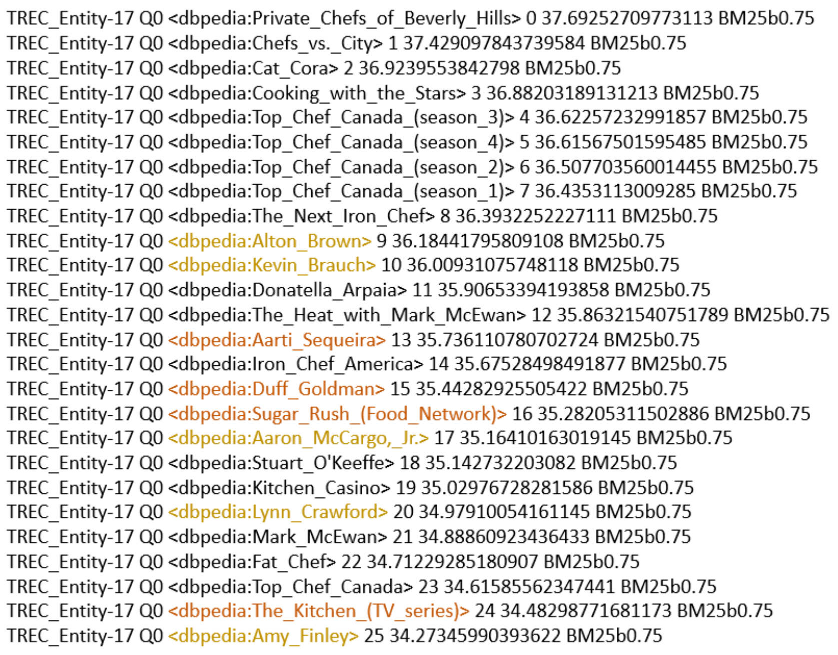
Figure 4.25: Representation of AP, P@5, P@10 and nDCG for fusion systems for topic 17. We can see that SUMFU performs very well in P@5 and P@10 while all the systems obtain similar results for AP and nDCG. The values are reported in Table A1.34 of the Appendix.

all the entities retrieved by the classic method are all present in our SUMFU run (in Figure 4.26 we display only the first 25 entities, but the run contains also the other entities of BM25 at lower rank).



```
TREC_Entity-17 Q0 <dbpedia:Stuart_O'Keeffe> 0 66.16061875820839 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Sugar_Rush_(Food_Network)> 1 60.30518978013436 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Duff_Goldman> 2 58.22733589793893 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Nadia_Giosia> 3 57.04239865336025 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Chef_at_Home> 4 56.56879795617862 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Martie_Duncan> 5 56.48169656642861 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:The_Food_Network_Awards> 6 55.87138596820445 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Richard_Blais> 7 55.29203719029931 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Cooking_with_the_Stars> 8 55.12458973874056 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Chocolate_with_Jacques_Torres> 9 53.162051973003244 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada_(season_4)> 10 52.79405434923429 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Aarti_Sequeira> 11 52.781815931288364 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:List_of_programs_broadcast_by_Food_Network> 12 52.67355478753589 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Benjamin_Sargent> 13 51.79267815289501 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Sugar_(TV_series)> 14 48.84564430818396 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Annabel_Langbein> 15 48.6398132959749 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Food_Feuds> 16 47.77663047481179 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Healthy_Appetite_with_Ellie_Krieger> 17 47.38634327763455 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:He_Said,_She_Said_(TV_series)> 18 45.13631702161482 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Spain..._on_the_Road_Again> 19 45.055547325191746 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Michael_Allemeier> 20 44.69577287458347 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Thea_Andrews> 21 44.53065164903638 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada> 22 44.48471714138343 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Minister_for_Agriculture,_Food_and_the_Marine> 23 44.04558107034011 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Iron_Chef> 24 43.69115640236292 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:PostPoint> 25 42.39070415919794 BM25b0.75
```

Figure 4.26: Top 25 entities retrieved from the SUMFU system for the topic 17. In orange we highlighted the entities marked as relevant in the relevance assessments, in blue the entities that could be considered relevant but that they are not and in green the relevant entities that are not retrieved from the BM25 run shown in Figure 4.27.



```
TREC_Entity-17 Q0 <dbpedia:Private_Chefs_of_Beverly_Hills> 0 37.69252709773113 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Chefs_vs._City> 1 37.429097843739584 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Cat_Cora> 2 36.9239553842798 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Cooking_with_the_Stars> 3 36.88203189131213 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada_(season_3)> 4 36.62257232991857 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada_(season_4)> 5 36.61567501595485 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada_(season_2)> 6 36.507703560014455 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada_(season_1)> 7 36.4353113009285 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:The_Next_Iron_Chef> 8 36.3932252227111 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Alton_Brown> 9 36.18441795809108 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Kevin_Brauch> 10 36.00931075748118 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Donatella_Arpaia> 11 35.90653394193858 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:The_Heat_with_Mark_McEwan> 12 35.86321540751789 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Aarti_Sequeira> 13 35.736110780702724 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Iron_Chef_America> 14 35.67528498491877 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Duff_Goldman> 15 35.44282925505422 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Sugar_Rush_(Food_Network)> 16 35.28205311502886 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Aaron_McCargo,_Jr.> 17 35.16410163019145 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Stuart_O'Keeffe> 18 35.142732203082 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Kitchen_Casino> 19 35.02976728281586 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Lynn_Crawford> 20 34.97910054161145 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Mark_McEwan> 21 34.88860923436433 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Fat_Chef> 22 34.71229285180907 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Top_Chef_Canada> 23 34.61585562347441 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:The_Kitchen_(TV_series)> 24 34.48298771681173 BM25b0.75
TREC_Entity-17 Q0 <dbpedia:Amy_Finley> 25 34.27345990393622 BM25b0.75
```

Figure 4.27: Top 25 entities retrieved from the BM25 system for the topic 17. In orange we highlighted the entities marked as relevant in the relevance assessments, in yellow the relevant entities that are not retrieved from the SUMFU run shown in Figure 4.26.

5 Discussion and Conclusion

In this thesis we proposed a new method for the Entity Search task, whose aim is to retrieve specific entities starting from the user's information need. In particular, as a collection we used the entities of the DBpedia dataset. In order to execute retrieval and obtain entities, we firstly generated virtual documents starting from the dataset. For the document creation we exploited graph embeddings and clustering. Through graph embeddings we represented each entity and all the information related to it with a numeric vector of features. These numeric vectors were then used for the next phase of clustering. The idea was to aggregate similar and highly correlated entities inside the same cluster and than use those clusters as documents for the Entity Retrieval. For this phase we use the BM25 model obtaining a ranked list of clusters. From this list we finally want to extract the individual entities to return to the user, therefore we implemented seven methods for this purpose. Our approaches can be divided into two main groups: combination systems and fusion systems. The combination systems are those methods that simply manipulate entities inside each cluster in order to generate the final ranked list of entities. The fusion systems are those methods that exploit both the ranked list of clusters and the ranked list returned by the state-of-the-art approach in order to generate the final ranked list of entities.

In the evaluation of our approaches we have seen that combination systems obtains results that are always worse than the ones of BM25 even though there are some topics in which they perform better. Fusion systems, on the other hand, obtains results that are very similar to those of BM25. In particular LEFU system turns out to be the best in both MAP and nDCG measures, while SIFU turns out to be the best in P@10. Analyzing these results we notice that the performances are highly affected by the cluster creation. If these sets are too generic, we will have many non relevant entities that during the composition or fusion phase will be included into the final run. If the clusters are too specific, we will consequently have a loss of relevant entities. These entities will not appear in our cluster and therefore they will not be included into the final run. Also the way in which we assembles entities it is fundamental for the success of the retrieval because it affects the order in which entities appear. In combination systems we have to insert a number of entities that facilitates both the insertion of correlated entities and the preservation of clusters rank. If we insert too many entities it is very likely that some of them are not relevant to the topic and therefore we will have a shift of the relevant entities of the second cluster to a lower rank. If we insert too few entities it is very likely that we miss to insert some relevant entities into the final run. The reordering in some of the combination systems was made for this purpose, in order to obtain a ranking based on the entities scores. Despite this, the systems still have poor performances and obtain results comparable with the ones of BM25 only in the fusion systems.

In fusion approaches we have seen that we obtain results very similar the ones of BM25 because we highly rely on its run during the fusion phase. We maintain good results in some specific topic while in other we obtain exactly the same values as the ones of BM25. This is due to the fact that if we insert many entities with a very low rank they are shifted down into the ranked list obtaining a final run that for the higher ranks is the same of BM25. In Chapter 4 we also see that in many cases we retrieve entities that are actually very related to the ones marked as relevant in the relevance judgment but that aren't considered as such, leading to a deterioration in performances.

In conclusion, we have seen that our approaches turns out to be promising in Entity Search task because they are able to enrich the retrieval with entities that are generally not detected by state-of-the-art systems and that are relevant and highly correlated to the topic.

5.1 Future Work

Further research should be carried out to improve our combination and fusion systems through a more detailed analysis on the choice of parameter settings, explore the use of fuzzy clustering, explore the effects of the aggregation of similar entities and improve combination and fusion phases.

Regarding the first point, further studies are needed to perform an in depth analysis on the setting of both graph embeddings and clustering parameters. The first in order to obtain better entities representations through a more effective traversal of nodes and edges of the graph. The second in order to obtain better clusters both as regards their completeness and consistency. Regarding the second point, it would be interesting to explore fuzzy clustering because it would allow for a more faithfully representation of the reality. It is in fact natural to think that an entity can belong to more than one cluster. Making an example, if we have a chef holding a TV show it would be reasonable to think that this entity belongs in both "*chef*" cluster and "*TV personality*" cluster. Regarding the third point, aggregate similar entities into a new one could overcome the pooling problem described in the previous Chapter. In this way we would in fact obtain a new entity containing all the useful information of the other similar ones. This approach requires a corresponding change in the set of relevance judgments, which must be subjected to the same process in order to obtain a match between entities in the evaluation phase. Regarding the fourth point, in order to improve our systems performances, it would be also useful to implement more sophisticated ways to add entities into the final run. In particular, further studies are needed to better understand how, how many and which entities to insert into the final run considering their rank, score, and belonging cluster.

References

[1] Amazon Mechanical Turk. Overview. https://www.mturk.com, 2019. Accessed: 2019-08-29.

[2] M. Angelini, N. Ferro, G. Santucci, and G. Silvello. Virtue: A visual tool for information retrieval performance evaluation and failure analysis. *Journal of Visual Languages & Computing*, 25(4):394–413, 2014.

[3] K. Balog. *Entity-Oriented Search*, volume 39 of *The Information Retrieval Series*. Springer International Publishing, 1 edition, 2018.

[4] K. Balog et al. *People search in the enterprise.* Universiteit van Amsterdam [Host], 2008.

[5] K. Balog, E. Meij, and M. De Rijke. Entity search: building bridges between two worlds. In *Proceedings of the 3rd International Semantic Search Workshop*, page 9. ACM, 2010.

[6] K. Balog and R. Neumayer. A test collection for entity search in dbpedia. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 737–740. ACM, 2013.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[8] C. Bizer and P. Mika. The semantic web challenge, 2009. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 2010.

[9] R. Blanco, H. Halpin, D. M. Herzig, P. Mika, J. Pound, H. S. Thompson, and T. T. Duc. Entity search evaluation over structured web data. In *Proceedings of the 1st international workshop on entity-oriented search workshop (SIGIR 2011), ACM, New York*, 2011.

[10] R. Blanco, P. Mika, and S. Vigna. Effective and efficient entity search in rdf data. In *International Semantic Web Conference*, pages 83–97. Springer, 2011.

[11] M. Bron, K. Balog, and M. De Rijke. Example based entity search in the web of data. In *European Conference on Information Retrieval*, pages 392–403. Springer, 2013.

[12] J. Coffman and A. C. Weaver. An empirical performance evaluation of relational keyword search techniques. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):30–42, 2012.

[13] E. Cohen. Node2vec. https://github.com/eliorc/node2vec, 2018. Accessed: 2019-09-03.

[14] DBpedia dataset. Dbpedia version 2015-10. http://downloads.dbpedia.org/2015-10. Accessed: 2019-09-03.

[15] ELKI Data Mining Framework. Overview. https://elki-project.github.io. Accessed: 2019-09-04.

[16] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[17] Gensim. Word2vec. https://radimrehurek.com/gensim/models/word2vec.html. Accessed: 2019-09-03.

[18] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[19] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2009.

[20] H. Halpin, D. M. Herzig, P. Mika, R. Blanco, J. Pound, H. Thompon, and D. T. Tran. Evaluating ad-hoc object retrieval. In *IWEST@ ISWC*, 2010.

[21] A. Harth. Billion Triples Challenge data set. Downloaded from http://km.aifb.kit.edu/projects/btc-2009/, 2009.

[22] F. Hasibi, F. Nikolaev, C. Xiong, K. Balog, S. E. Bratsberg, A. Kotov, and J. Callan. Dbpedia-entity v2: a test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1265–1268. ACM, 2017.

[23] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM, 2012.

[24] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[25] J. Leskovec. Node2vec. http://snap.stanford.edu/node2vec. Accessed: 2019-09-01.

[26] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[27] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects. In *Proceedings of the 21st international conference on World wide web*, pages 589–598. ACM, 2012.

[28] X. Liu and H. Fang. A study of entity search in semantic search workshop. In *Proc. of the 3rd Intl. Semantic Search Workshop*, 2010.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[30] T. Mikolov, J. Kopecky, L. Burget, O. Glembek, et al. Neural network based language models for highly inflective languages. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4725–4728. IEEE, 2009.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[32] R. Neumayer, K. Balog, and K. Nørvåg. On the modeling of entities for ad-hoc entity search in the web of data. In *European Conference on Information Retrieval*, pages 133–145. Springer, 2012.

[33] P. Ogilvie, J. Callan, and J. Callan. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 143–150. ACM, 2003.

[34] J. R. Pérez-Agüera, J. Arroyo, J. Greenberg, J. P. Iglesias, and V. Fresno. Using bm25f for semantic search. In *Proceedings of the 3rd international semantic search workshop*, page 2. ACM, 2010.

[35] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[36] S. J. Phillips. Acceleration of k-means and related clustering algorithms. In *Workshop on Algorithm Engineering and Experimentation*, pages 166–177. Springer, 2002.

[37] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th international conference on World wide web*, pages 771–780. ACM, 2010.

[38] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[39] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[40] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR94*, pages 232–241. Springer, 1994.

[41] G. Salton. *Automatic information organization and retrieval.* McGraw-Hill, 1968.

[42] E. Schubert and A. Zimek. ELKI: A large open-source library for data analysis - ELKI release 0.7.5 "heidelberg". *CoRR*, abs/1902.03616, 2019.

[43] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[44] Terrier IR Platform. Overview. http://terrier.org. Accessed: 2019-09-04.

[45] Text REtrieval Conference. Overview. https://trec.nist.gov/overview.html, 2019-05-08. Accessed: 2019-08-18.

[46] Y. Velegrakis. Relational technologies, metadata and rdf. In *Semantic Web Information Management*, pages 41–66. Springer, 2010.

[47] Yahoo US query log. Datasets. http://webscope.sandbox.yahoo.com, 2019. Accessed: 2019-08-29.

[48] J. Yang and J. Leskovec. Overlapping communities explain core–periphery organization of networks. *Proceedings of the IEEE*, 102(12):1892–1902, 2014.

[49] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

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

[51] D. Zhou, S. Niu, and S. Chen. Efficient graph computation for node2vec. *arXiv preprint arXiv:1805.00280*, 2018.

Appendix

A1 Tables

In this section we report all the tables containing the resulting values of combination and fusion system for all the measures MAP, nDCG, P@5, P@10 and all the topics.

	MAP	nDCG	P@5	P@10
SICO	0,0757	0,2104	0,2118	0,1529
SUMCO	0,1159	0,2557	0,2471	0,1941
WECO	0,0907	0,2321	0,2353	0,1647
BM25	0,2086	0,4188	0,3529	0,2588

Table A1.1: Numeric results for MAP, average nDCG, average P@5 and average P@10 measures for combination systems.

	MAP	nDCG	P@5	P@10
SIFU	0,1950	0,3703	0,3529	0,2882
SOFU	0,2029	0,3732	0,3529	0,2588
LEFU	0,2093	0,4210	0,3529	0,2588
SUMFU	0,1596	0,3480	0,2824	0,2294
BM25	0,2086	0,4188	0,3529	0,2588

Table A1.2: Numeric results for MAP, average nDCG, average P@5 and average P@10 measures for fusion systems and BM25.

	SIFU		SOFU		LEFU		SUMFU	
	14	5	14	5	14	5	14	5
MAP	0,1805	0,1950	0,2022	0,2029	0,2070	0,2093	0,1399	0,1596
nDCG	0,3635	0,3703	0,3787	0,3732	0,4029	0,4210	0,3407	0,3480
P@5	0,3412	0,3529	0,3529	0,3529	0,3529	0,3529	0,2824	0,2824
P@10	0,2588	0,2882	0,2588	0,2588	0,2588	0,2588	0,2235	0,2294

Table A1.3: Numeric results for MAP, average nDCG, average P@5 and average P@10 measures for fusion systems with the addition of 14 entities or 5 entities.

TREC_Entity-1	SICO	SUMCO	WECO	BM25
AP	0,0225	0,0377	0,0579	0,0516
P@5	0,0000	0,0000	0,2000	0,2000
P@10	0,1000	0,1000	0,1000	0,1000
nDCG	0,2012	0,2325	0,2673	0,3685

Table A1.4: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 1.

TREC_Entity-2	SICO	SUMCO	WECO	BM25
AP	0,0080	0,0480	0,0202	0,2232
P@5	0,0000	0,2000	0,0000	0,2000
P@10	0,0000	0,1000	0,0000	0,1000
nDCG	0,0949	0,1724	0,1078	0,4660

Table A1.5: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 2.

TREC_Entity-4	SICO	SUMCO	WECO	BM25
AP	0,0169	0,0072	0,0071	0,0208
P@5	0,0000	0,0000	0,0000	0,0000
P@10	0,0000	0,0000	0,0000	0,0000
nDCG	0,1574	0,0843	0,0848	0,2159

Table A1.6: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 4.

TREC_Entity-5	SICO	SUMCO	WECO	BM25
AP	0,0281	0,2550	0,1792	0,4214
P@5	0,0000	0,2000	0,4000	0,4000
P@10	0,0000	0,2000	0,2000	0,4000
nDCG	0,2716	0,4865	0,5024	0,6134

Table A1.7: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 5.

TREC_Entity-6	SICO	SUMCO	WECO	BM25
AP	0,1440	0,1759	0,1253	0,2352
P@5	0,4000	0,4000	0,4000	0,6000
P@10	0,3000	0,3000	0,4000	0,4000
nDCG	0,3606	0,3975	0,3707	0,5363

Table A1.8: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 6.

TREC_Entity-7	SICO	SUMCO	WECO	BM25
AP	0,0255	0,0516	0,0206	0,1397
P@5	0,2000	0,2000	0,0000	0,4000
P@10	0,1000	0,1000	0,1000	0,3000
nDCG	0,1519	0,2761	0,1806	0,4840

Table A1.9: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 7.

TREC_Entity-9	SICO	SUMCO	WECO	BM25
AP	0,2667	0,3655	0,1510	0,3063
P@5	0,4000	0,4000	0,2000	0,6000
P@10	0,3000	0,4000	0,1000	0,3000
nDCG	0,4756	0,5187	0,2979	0,4375

Table A1.10: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 9.

TREC_Entity-10	SICO	SUMCO	WECO	BM25
AP	0,0424	0,0448	0,0091	0,0969
P@5	0,2000	0,2000	0,0000	0,4000
P@10	0,1000	0,1000	0,0000	0,3000
nDCG	0,1079	0,1154	0,0561	0,3409

Table A1.11: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 10.

TREC_Entity-11	SICO	SUMCO	WECO	BM25
AP	0,0232	0,0241	0,0587	0,1241
P@5	0,0000	0,0000	0,0000	0,2000
P@10	0,0000	0,0000	0,1000	0,1000
nDCG	0,1851	0,1852	0,2384	0,3400

Table A1.12: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 11.

TREC_Entity-12	SICO	SUMCO	WECO	BM25
AP	0,0540	0,0591	0,0340	0,0804
P@5	0,2000	0,2000	0,2000	0,4000
P@10	0,1000	0,1000	0,1000	0,2000
nDCG	0,1772	0,1997	0,1555	0,3380

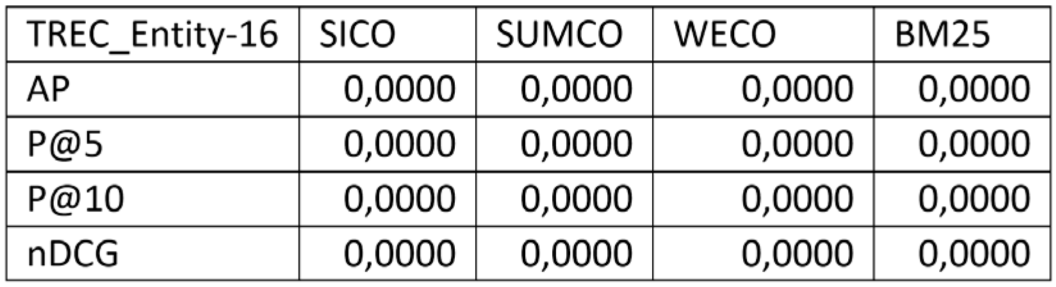
Table A1.13: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 12.

TREC_Entity-14	SICO	SUMCO	WECO	BM25
AP	0,0279	0,0833	0,0833	0,1307
P@5	0,2000	0,2000	0,2000	0,2000
P@10	0,1000	0,1000	0,1000	0,1000
nDCG	0,0970	0,1641	0,1641	0,3712

Table A1.14: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 14.

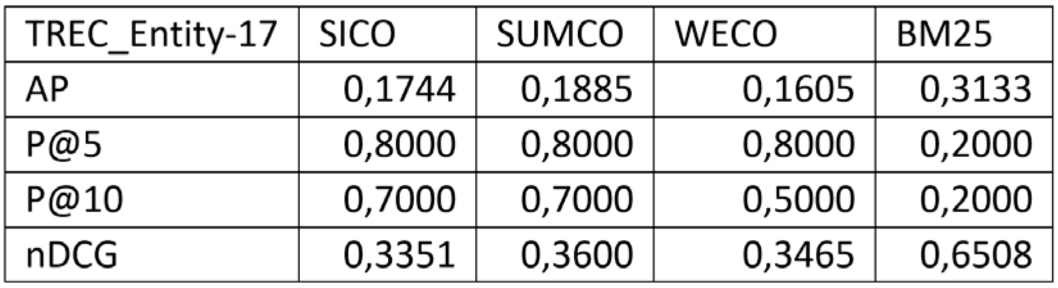
TREC_Entity-15	SICO	SUMCO	WECO	BM25
AP	0,0623	0,0419	0,0644	0,0525
P@5	0,4000	0,4000	0,6000	0,2000
P@10	0,2000	0,2000	0,3000	0,1000
nDCG	0,1735	0,1232	0,1566	0,2499

Table A1.15: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 15.



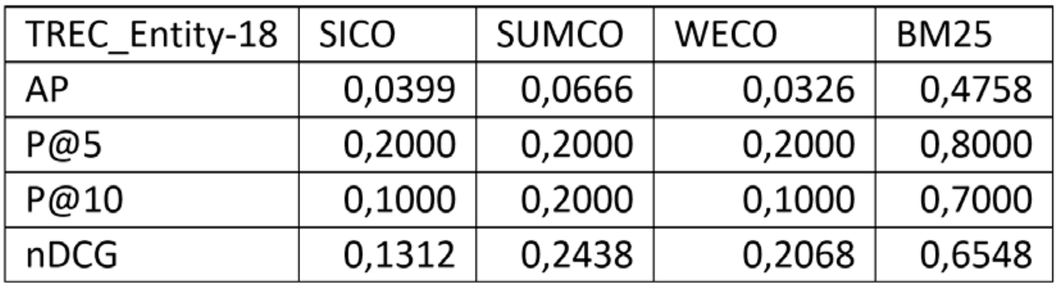
TREC_Entity-16	SICO	SUMCO	WECO	BM25
AP	0,0000	0,0000	0,0000	0,0000
P@5	0,0000	0,0000	0,0000	0,0000
P@10	0,0000	0,0000	0,0000	0,0000
nDCG	0,0000	0,0000	0,0000	0,0000

Table A1.16: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 16.



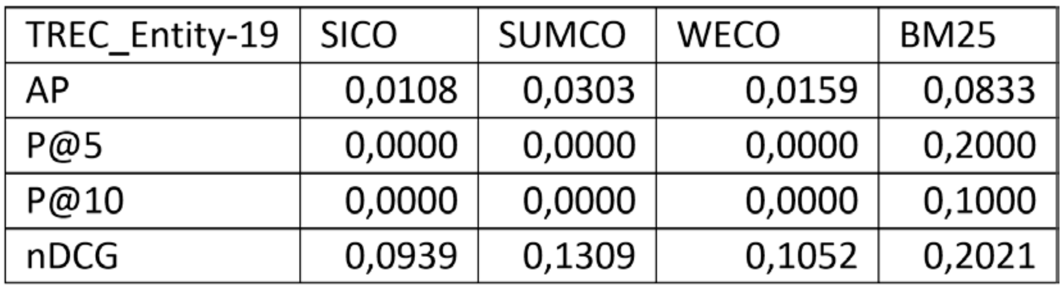
TREC_Entity-17	SICO	SUMCO	WECO	BM25
AP	0,1744	0,1885	0,1605	0,3133
P@5	0,8000	0,8000	0,8000	0,2000
P@10	0,7000	0,7000	0,5000	0,2000
nDCG	0,3351	0,3600	0,3465	0,6508

Table A1.17: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 17.



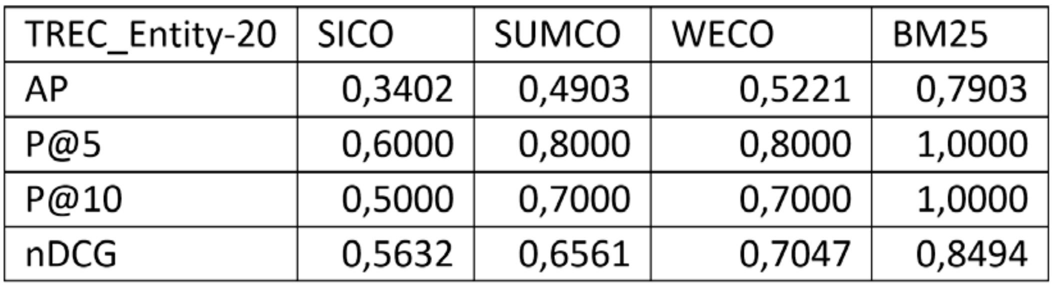
TREC_Entity-18	SICO	SUMCO	WECO	BM25
AP	0,0399	0,0666	0,0326	0,4758
P@5	0,2000	0,2000	0,2000	0,8000
P@10	0,1000	0,2000	0,1000	0,7000
nDCG	0,1312	0,2438	0,2068	0,6548

Table A1.18: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 18.



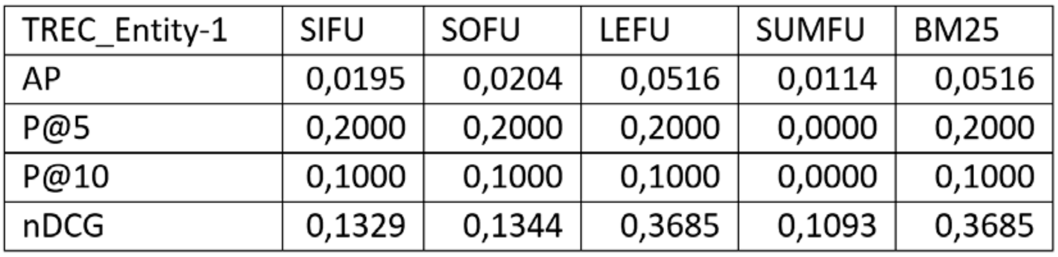
TREC_Entity-19	SICO	SUMCO	WECO	BM25
AP	0,0108	0,0303	0,0159	0,0833
P@5	0,0000	0,0000	0,0000	0,2000
P@10	0,0000	0,0000	0,0000	0,1000
nDCG	0,0939	0,1309	0,1052	0,2021

Table A1.19: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 19.



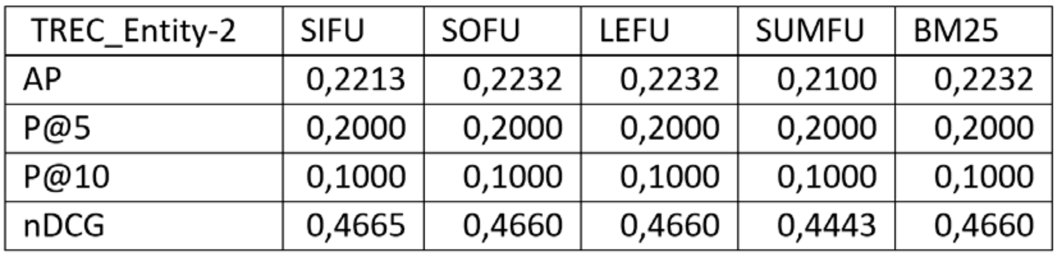
TREC_Entity-20	SICO	SUMCO	WECO	BM25
AP	0,3402	0,4903	0,5221	0,7903
P@5	0,6000	0,8000	0,8000	1,0000
P@10	0,5000	0,7000	0,7000	1,0000
nDCG	0,5632	0,6561	0,7047	0,8494

Table A1.20: Numeric results for AP, P@5, P@10 and nDCG measures for combination systems for topic 20.



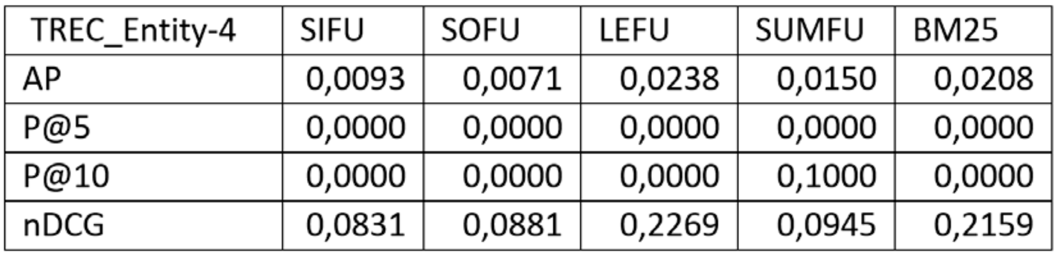
TREC_Entity-1	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0195	0,0204	0,0516	0,0114	0,0516
P@5	0,2000	0,2000	0,2000	0,0000	0,2000
P@10	0,1000	0,1000	0,1000	0,0000	0,1000
nDCG	0,1329	0,1344	0,3685	0,1093	0,3685

Table A1.21: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 1.



TREC_Entity-2	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,2213	0,2232	0,2232	0,2100	0,2232
P@5	0,2000	0,2000	0,2000	0,2000	0,2000
P@10	0,1000	0,1000	0,1000	0,1000	0,1000
nDCG	0,4665	0,4660	0,4660	0,4443	0,4660

Table A1.22: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 2.



TREC_Entity-4	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0093	0,0071	0,0238	0,0150	0,0208
P@5	0,0000	0,0000	0,0000	0,0000	0,0000
P@10	0,0000	0,0000	0,0000	0,1000	0,0000
nDCG	0,0831	0,0881	0,2269	0,0945	0,2159

Table A1.23: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 4.

TREC_Entity-5	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,4238	0,4214	0,4214	0,2081	0,4214
P@5	0,4000	0,4000	0,4000	0,2000	0,4000
P@10	0,5000	0,4000	0,4000	0,2000	0,4000
nDCG	0,6329	0,6134	0,6134	0,4684	0,6134

Table A1.24: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 5.

TREC_Entity-6	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,1929	0,2352	0,2352	0,3052	0,2352
P@5	0,6000	0,6000	0,6000	0,8000	0,6000
P@10	0,4000	0,4000	0,4000	0,6000	0,4000
nDCG	0,4966	0,5363	0,5363	0,5527	0,5363

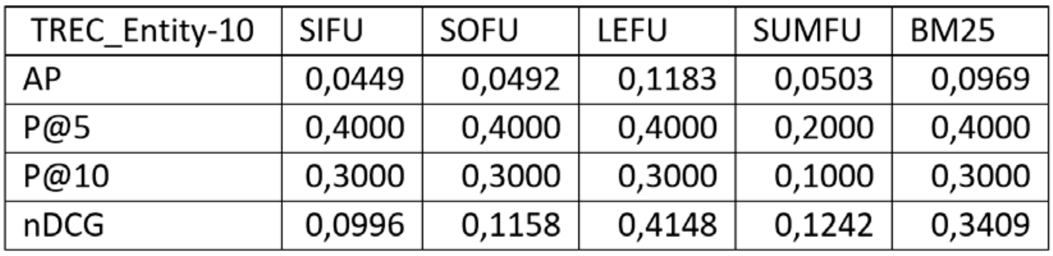
Table A1.25: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 6.

TREC_Entity-7	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,1151	0,1226	0,1332	0,0746	0,1397
P@5	0,4000	0,4000	0,4000	0,0000	0,4000
P@10	0,4000	0,3000	0,3000	0,1000	0,3000
nDCG	0,3392	0,3405	0,4380	0,2928	0,4840

Table A1.26: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 7.

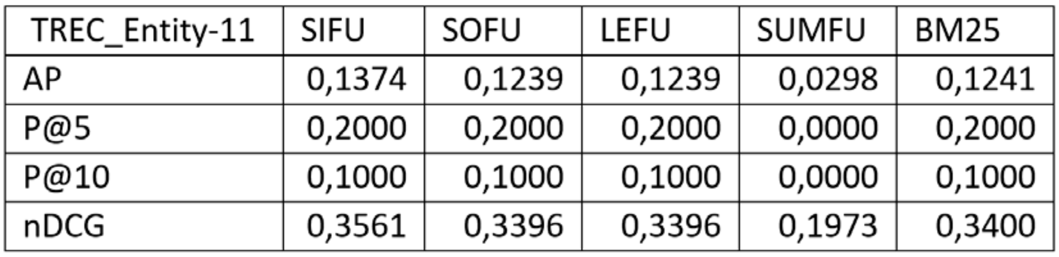
TREC_Entity-9	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,3125	0,3064	0,3064	0,2603	0,3063
P@5	0,6000	0,6000	0,6000	0,4000	0,6000
P@10	0,3000	0,3000	0,3000	0,2000	0,3000
nDCG	0,4410	0,4376	0,4376	0,4702	0,4375

Table A1.27: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 9.



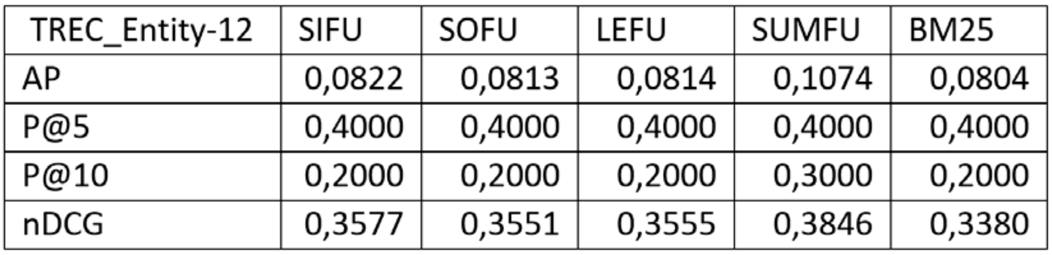
TREC_Entity-10	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0449	0,0492	0,1183	0,0503	0,0969
P@5	0,4000	0,4000	0,4000	0,2000	0,4000
P@10	0,3000	0,3000	0,3000	0,1000	0,3000
nDCG	0,0996	0,1158	0,4148	0,1242	0,3409

Table A1.28: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 10.



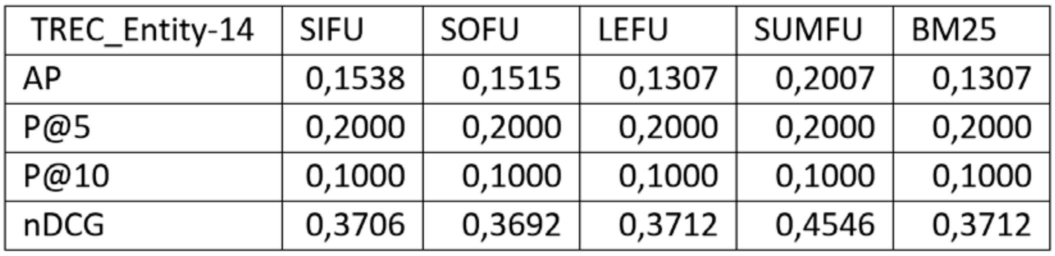
TREC_Entity-11	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,1374	0,1239	0,1239	0,0298	0,1241
P@5	0,2000	0,2000	0,2000	0,0000	0,2000
P@10	0,1000	0,1000	0,1000	0,0000	0,1000
nDCG	0,3561	0,3396	0,3396	0,1973	0,3400

Table A1.29: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 11.



TREC_Entity-12	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0822	0,0813	0,0814	0,1074	0,0804
P@5	0,4000	0,4000	0,4000	0,4000	0,4000
P@10	0,2000	0,2000	0,2000	0,3000	0,2000
nDCG	0,3577	0,3551	0,3555	0,3846	0,3380

Table A1.30: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 12.



TREC_Entity-14	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,1538	0,1515	0,1307	0,2007	0,1307
P@5	0,2000	0,2000	0,2000	0,2000	0,2000
P@10	0,1000	0,1000	0,1000	0,1000	0,1000
nDCG	0,3706	0,3692	0,3712	0,4546	0,3712

Table A1.31: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 14.

TREC_Entity-15	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0560	0,0421	0,0458	0,0553	0,0525
P@5	0,2000	0,2000	0,2000	0,4000	0,2000
P@10	0,1000	0,1000	0,1000	0,3000	0,1000
nDCG	0,2191	0,2000	0,2319	0,2032	0,2499

Table A1.32: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 15.

TREC_Entity-16	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0000	0,0000	0,0000	0,0000	0,0000
P@5	0,0000	0,0000	0,0000	0,0000	0,0000
P@10	0,0000	0,0000	0,0000	0,0000	0,0000
nDCG	0,0000	0,0000	0,0000	0,0000	0,0000

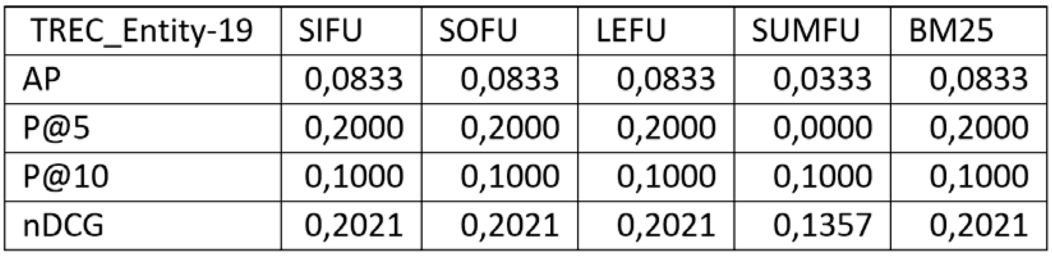
Table A1.33: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 16.

TREC_Entity-17	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,3093	0,3108	0,3133	0,2923	0,3133
P@5	0,2000	0,2000	0,2000	0,8000	0,2000
P@10	0,3000	0,2000	0,2000	0,7000	0,2000
nDCG	0,6418	0,6373	0,6508	0,6402	0,6508

Table A1.34: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 17.

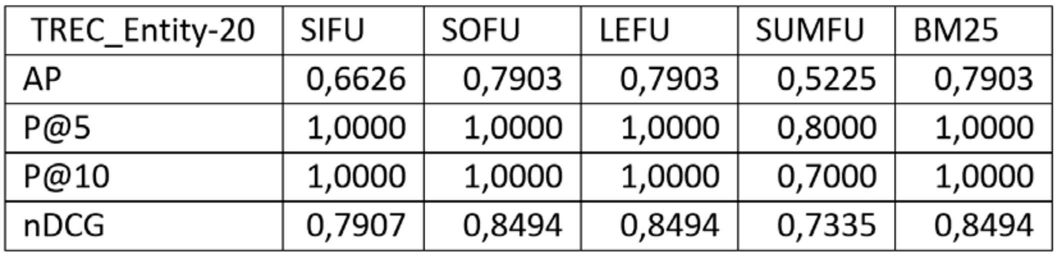
TREC_Entity-18	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,4911	0,4799	0,4760	0,3365	0,4758
P@5	0,8000	0,8000	0,8000	0,4000	0,8000
P@10	0,9000	0,7000	0,7000	0,3000	0,7000
nDCG	0,6647	0,6594	0,6551	0,6107	0,6548

Table A1.35: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 18.



TREC_Entity-19	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,0833	0,0833	0,0833	0,0333	0,0833
P@5	0,2000	0,2000	0,2000	0,0000	0,2000
P@10	0,1000	0,1000	0,1000	0,1000	0,1000
nDCG	0,2021	0,2021	0,2021	0,1357	0,2021

Table A1.36: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 19.



TREC_Entity-20	SIFU	SOFU	LEFU	SUMFU	BM25
AP	0,6626	0,7903	0,7903	0,5225	0,7903
P@5	1,0000	1,0000	1,0000	0,8000	1,0000
P@10	1,0000	1,0000	1,0000	0,7000	1,0000
nDCG	0,7907	0,8494	0,8494	0,7335	0,8494

Table A1.37: Numeric results for AP, P@5, P@10 and nDCG measures for fusion systems for topic 20.