University of Amsterdam

Master Thesis

Explaining relationships between entities

Author:
Nikos Voskarides

Supervisors:
Dr. Edgar Meij,
Dr. Manos Tsagkias

A thesis submitted in fulfilment of the requirements
for the degree of Master's of Science in Artificial Intelligence

November 2014

Acknowledgements

I would like to thank my daily supervisor Edgar Meij for his support and encouragement throughout this thesis. Our numerous discussions and his feedback were always insightful and inspiring.

Many thanks to my co-supervisor Manos Tsagkias for giving useful feedback that helped to improve the quality of this work. His support and advice have been invaluable for me during the last two years.

I would like to express my gratitude to Maarten de Rijke for giving me the opportunity to work on interesting research problems with him and other exciting people.

Thanks to all ILPS members and especially Anne Schuth, Daan Odijk and Wouter Weerkamp for supporting me.

I thank Yahoo for providing the data used in this work and especially the Semantic Search research group in Barcelona.

Also, I thank Henk Zeevat and Maarten de Rijke for agreeing to be members of the examination committee.

Finally, I would like to thank my parents and my brothers for their endless support.

Abstract

Modern search engines are increasingly aiming to understand users' intent in order to answer information needs more effectively by providing richer information than the traditional "ten blue links". This information might include context about the entities present in the query, direct answers to questions that concern entities and more. A recent trend when answering queries that refer to a single entity is providing an additional panel that contains some basic information about the entity, along with links to other entities that are related to the initial entity. A problem that remains largely unexplored is how to provide an explanation of why two entities are related. In this work, we study the problem of explaining pre-defined relations of entity pairs with natural language sentences in the context of search engines. We propose a method that first extracts sentences that refer to each entity pair and then ranks the sentences by how well they describe the relation between the two entities. Our ranking module combines a rich set of features using state-of-the-art learning to rank algorithms. We evaluate our method on a dataset of entities and relations used by a commercial search engine. The experimental results demonstrate the effectiveness of our method, which can be efficiently applied in a search engine scenario.

Contents

Chapter 1

Introduction

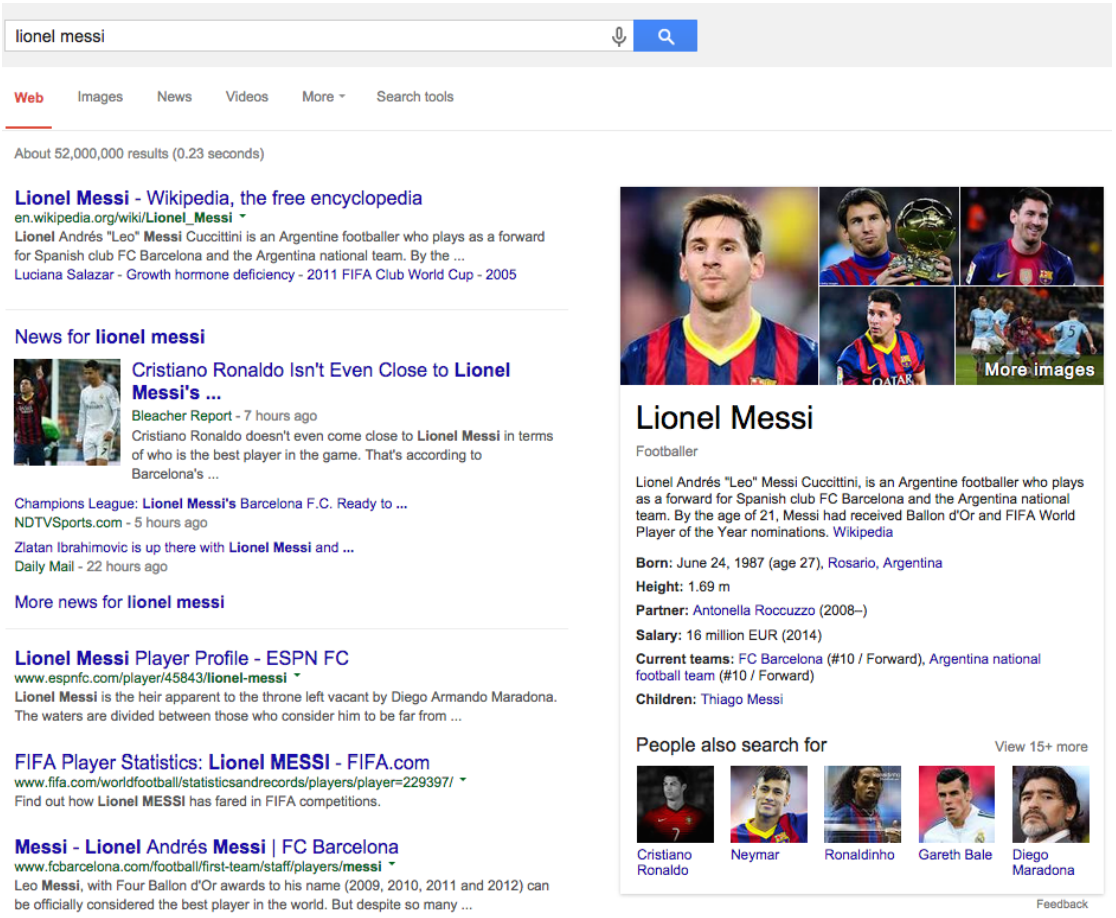
Commercial search engines are moving towards incorporating semantic information in their result pages. Examples include answering specific questions (e.g. "Barcelona weather", "who built the Eiffel tower") and presenting entities related to a query (e.g. the tennis player for the query "Roger Federer"), usually in a condensed version. This changes the way users traditionally interact with search results pages, as the information need might be satisfied by observing the provided information only, without having to observe the traditional web page results.

Recent work has focused on devising methods that provide semantically enriched search results [8, 31, 76]. In order to be able to provide the users with the right information, it is necessary to understand the semantics of the query. An analysis conducted in [50] showed that 71% of queries contain entities, a result which motivates research in recognizing and identifying entities in queries [50, 84]. Semantic information for entities is usually obtained from external structured data sources (i.e. knowledge bases). Deciding which data source is relevant to the query requires understanding the query intent and several methods have been proposed recently that try to tackle this problem [24, 58, 74, 107, 131].

Major commercial search engines, including Google,[1] Microsoft and Yahoo, identify and link entities in queries to a knowledge base and provide the user with context about the entity he/she is searching for. An example of this is Google's knowledge graph.[2] The contextual information about the entities is mined from knowledge bases such as Wikipedia, Freebase or proprietary ones constructed by the search engines. It usually includes a basic description, some important information about the entity (e.g. headquarters location for companies or details of birth for people) and a list of related entities. An example of this can be seen in figure 1.1, which shows the result page

[1] http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html
[2] http://www.google.com/insidesearch/features/search/knowledge.html

FIGURE 1.1: Part of Google's search result page for the query "lionel messi". The information about the identified entity is shown in the right panel of the page.



of Google's search engine when searching for the Argentinian football player "Lionel Messi".

The problem of providing users with the most relevant results to the query has been the main problem studied in the context of search engines. However, the rapid growth of the web has increased the need to support exploratory and serendipitous search [6, 46, 67]. Search engines have tried to overcome this challenge by providing query and web page recommendations [13, 20, 61, 81]. The development of semantic web techniques, including the creation of entity graphs which include information about entities and relationships[3] between them, has made it easier to explore entity recommendations [12, 133]. In that way it is likely to enhance user engagement, as users searching for a particular entity may also be interested in finding information about other, related entities. The right panel of the search results page in figure 1.1 shows the top-5 related entities to "Lionel Messi", suggested by Google's search engine.

[3]In this thesis we use the terms "relationship" and "relation" interchangeably.

FIGURE 1.2: Part of Google's search result page for the query "barack obama". When placing the mouse over the related entity "Michelle Obama", an explanation of the relation between her and "Barack Obama" is shown in a yellow box.

An important component that is usually missing when suggesting related entities is the explanation of why these are related to the entity that the user searched for. For example, when suggesting a related movie director for an actor, one would expect to see in which movies the two co-operated, possibly with some more information about their relation. In this work we focus on this problem which we name *Entity Relationship Explanation*. The motivation is that even though some commercial search engines provide a description of the relation for some entity pairs (see figure 1.2), this does not happen for every pair. For example, none of the suggested related entities in figure 1.1 has an explanation of the relation to football player Lionel Messi, whereas it would be expected to explain for example that both Messi and Neymar are players of Barcelona F.C. In addition, to the best of our knowledge, there is no published work being explicit on how search engines generate descriptions of relations between entities.

Recently, there has been an emerging interest in relationship extraction [3, 9, 16, 36, 92], sentence retrieval and question answering [1, 94, 98, 120, 129], and learning to rank (LTR) [77]. This work tries to combine ideas from these areas to address the problem of explaining relationships between entities. A related study has focused on finding and ranking sentences that explain the relationship of an entity and a query [11], while

REX [37] has focused on generating a ranked list of knowledge base relationships for an entity pair. Our work differs from the former study in that we want to explain the relation of two different entities, and from the latter in that we try to select sentences that describe a particular relation, assuming that this relation is given.

In this work, we approach *Entity Relationship Explanation* as a sentence ranking problem. Given a pair of entities and a pre-defined relation between them, we automatically extract sentences from a document corpus and rank them with respect to how well they describe the relation of the entities. Our main goal is to have a sentence that perfectly explains the relation in the top position of the ranking. We employ a rich set of features and use state-of-the-art supervised learning to rank algorithms in order to effectively combine them. Our feature set includes both traditional information retrieval and natural language processing features which we augment with entity-dependent features that can leverage information from the structure of a knowledge base. In addition, we use features that try to capture the presence of the relation of interest in a sentence.

This work was done in collaboration with a major commercial search engine, Yahoo, and focuses on explaining relationships between "people" entities in the context of web search. We test our methods on a dataset of entities and relations used by the Yahoo search engine in production. We give more details about this dataset in section 4.1.1.

1.1 Research Questions

The main research question of this thesis is whether we can effectively explain the relation of interest of an entity pair in a knowledge base. In order to address this research question we aim at answering the following sub-questions. First, we examine which is the most effective method among state-of-the-art retrieval models and learning to rank algorithms to explain a relation of interest between two entities (RQ1). To this end, we perform a comparative study of these methods. We also experiment on how different parameter settings affect retrieval performance (RQ2). Furthermore, we investigate which features among the ones we devised for this task are the most important in a machine learning scenario (RQ3). In addition, we examine how difficult this task is for human annotators by measuring inter-annotator agreement (RQ4). Finally, we examine how our method performs compared to a competing system on a separate entity pairs dataset that contains popular entities (RQ5).

1.2 Contributions

The main contributions of this work are:

- a supervised method for ranking sentences with respect to how well they explain the relation of an entity pair;
- insights into how the performance varies with different learning to rank algorithms and feature sets;
- analysis of failure cases and suggestions for improvements;
- a manually annotated dataset which we plan to make publicly available to the research community.

The remainder of this thesis is organized as follows. In Chapter 2 we discuss related work. Our methods are described in Chapter 3. In Chapter 4 we describe the experimental setup and in Chapter 5 we report and analyse our results. We conclude and discuss future research directions in Chapter 6.

Chapter 2

Related work

In this section we provide an overview of work in research areas that are directly related to the problem tackled in this thesis. More specifically, we describe work in semantic search, sentence retrieval, question answering, relation extraction and learning to rank.

2.1 Semantic search

Semantic search aims at improving the search experience by understanding users' intent and the contextual meaning of queries and documents. Search engines utilize knowledge bases which contain entities, relations, facts and more in order to enrich their results with rich context and eventually improve the level of satisfaction from answering information needs. In order to achieve this, semantic search faces several challenges, including entity linking, entity retrieval and query understanding, which we describe below.

A crucial component of semantic search is entity linking (also called entity disambiguation), which is the task of automatically linking raw text to entities. These entities are usually taken from semi-structured knowledge bases such as Wikipedia [30, 40, 52, 54, 57, 69, 85, 87, 100, 106]. Among others, entity linking can facilitate the design of entity-oriented user interfaces that can help the users access additional relevant information, enable automatic knowledge base population and be used in order to improve text classification and retrieval.

Entity linking can be split in three steps: the detection of mentions (phrases) that are worth linking, the generation of candidate entities that a mention might link to and the selection of the best candidate concepts to be linked according to the context. Some of these steps can either be merged or performed in a different order. The first attempt on entity linking first detects mentions using link probabilities that are constructed

based on how Wikipedia articles are linked inside Wikipedia and then disambiguates the mentions to the appropriate entities [87]. The disambiguation is performed using features of the mentions and the surrounding words. Another attempt on entity linking first detects unambiguous entities that form the disambiguation context and then performs the disambiguation by utilizing the similarity between each possible entity candidate and the entities in the context, along with other features [91]. Similarity between entities is computed using a semantic similarity measure for Wikipedia articles [126].

One challenge we face in this work is that the documents we extract the sentences from are already linked to entities, but each entity is only linked once in the document, thus not every sentence is linked to the entities it mentions. Since we are interested in the entities each individual sentence mentions, we propose a heuristic entity linking algorithm that links each sentence in a document to entities already linked in the document. This approach is described in Section 3.1.1.

Another aspect of semantic search is entity retrieval, which regards information needs that are more effectively answered by enhancing the document results with specific entities [32, 125]. Entity retrieval can be done using unstructured text collections, (semi)structured data sources or a mixture of these [97, 113, 122]. The application of entity retrieval which is closest to our work is related entity finding or recommendation [17, 33]. Given a specific entity, entity recommendation aims at finding entities that are somehow related to that entity. Entity recommendation has recently been applied by commercial search engines to support exploratory and serendipitous search. A publicly available study combines various data sources and uses a rich set of features to provide entity recommendations for web search for a major commercial search engine [12]. Another approach uses heterogeneous information networks and implicit feedback [132]. This problem has also been studied in the context of personalized recommendation [133]. A complementary problem to entity recommendation is the explanation of why two entities are related, a problem which we address in this work.

Query understanding is fundamental for effectively answering user information needs. This involves identifying users' search intent which helps the search engines to provide the users with direct answers to the queries. Several approaches have been proposed for search intent prediction, Some of these approaches use text information from queries or web pages [134], search logs [29] or combinations of the two [105]. Other approaches focus on mining query templates or structured data in order to identify query intents and attributes in them [2, 112]. An important part of query understanding is to identify the entities that appear in the query [50, 84]. This was one of the tasks of the recent

ERD challenge which received considerable attention.[1] Recently, involving entities have been found beneficial for interpreting user intent [13, 58, 103, 107, 131].

In this work, we draw inspiration from the recent advances in semantic search and involve ideas from this area by utilizing knowledge base entities, entity attributes and knowledge base structure in order to facilitate relationship explanation between entities.

2.2 Sentence retrieval

Sentence retrieval regards finding relevant sentences that answer an information need in a sentence corpus. It has been applied in various information retrieval applications including novelty detection [5], summarization [44], question answering [98] and opinion mining [68].

One of the first approaches for sentence retrieval introduced a vector space based model, tf-isf, which is a variation of the classic tf-idf function used for document retrieval [5]. tf-isf accounts for inverse sentence frequency of terms in contrast to tf-idf which accounts for inverse document frequency of terms. Despite its simplicity, tf-isf is considered very competitive compared to document retrieval models tuned specifically for sentence retrieval, such as BM25 and language modeling [79]. Empirical valuations on passage retrieval suggested that methods based on vector-space models perform well when retrieving small pieces of text [64, 65].

The tf-isf function is unaware of the surrounding sentences and the document from which a sentence was extracted from. These signals can provide context regarding the relevance of a sentence with respect to a topic. Several methods have been proposed that try to incorporate local context in sentence retrieval. These methods include involving mixture models that take into account not only the sentence but also the document and the collection [95], incorporating very frequent words from the top retrieved documents in the retrieval function [80], or adjusting tf-isf and the well-known language modeling framework to account for sentence context [34, 39].

Another study approached question-based sentence retrieval using topic-sensitive LexRank [35] that accounts for the relevance of the sentence to the question and the similarity between the candidate sentences [98]. Query expansion has also been studied for the task of sentence retrieval in order to address the vocabulary mismatch between the query and the sentences [26, 79]. This problem was also tackled using translation models for monolingual data modified for sentence retrieval [94]. Translation models proved to be successful for question answering and novelty detection. We build on this idea by

[1] http://web-ngram.research.microsoft.com/ERD2014/

utilizing methods for obtaining phrases similar to the relation in order to account for vocabulary mismatch. These methods are described in Section 3.2.2.3.

Our method involves a sentence ranking module that aims to retrieve the sentences that best describe the relation of interest between two entities. Instead of only relying on sentence retrieval functions, our method combines various features that can help identify relevant sentences using state-of-the-art learning to rank algorithms.

2.3 Question answering

Question answering (QA) is the task of providing direct and concise answers to questions formed in natural language [56]. QA is a very popular task in natural language processing and information retrieval. In fact, it is considered one of the oldest natural language processing tasks [115] and it has gained considerable attention since the launch of the TREC question answering track [123]. A famous automatic question answering system developed at IBM named Watson won the Jeopardy quiz television show in February 2011 [41]. Here we give a brief overview of the main pipeline of QA systems. For a more comprehensive overview of QA, please refer for example to [56, 78, 124].

There are two types of questions that QA systems answer : questions about facts (factoid) and narrative questions (non-factoid). Most prior work in QA has focused on factoid questions, although there has also been interest in non-factoid questions [4, 117, 120]. Our task is more related to factoid QA, as the explanation of a relation of an entity pair can be considered as a fact. For this reason, we base our discussion on this type of QA.

Factoid QA in the IR setting usually consists of three main components : question processing, passage retrieval and answer processing [62, 124]. Each of these components is a separate machine learning task with different feature sets.

Question processing extracts the type of the question and the answer, which are usually represented with named entity types [73]. It also filters terms from the question and chooses keywords for document retrieval. Furthermore, it finds the question terms that should be replaced in the answer and finds relations between entities in the questions Passage retrieval starts with document retrieval using the keywords extracted from the previous step as the query terms. Then, it segments the documents into passages and finally ranks the passages by using the answer type extracted from the previous step. The final component, answer processing, first extracts and then ranks candidate answers using features extracted both from the text and from external data sources, such as knowledge bases.

Note that QA can be regarded as a similar task to ours, assuming that the entity pair and the relation of interest form the "question" and that the "answer" is the sentence describing the relation of interest. Even though we do not follow the QA paradigm in this work, some of the features we used are inspired from QA systems. In addition, we employ learning to rank to combine the devised features, which has recently been successfully applied for QA [1, 120].

2.4 Relation extraction

Relation extraction is the task of extracting semantic relations between entities from text. This is useful for several applications including knowledge base enrichment [83] and question answering [75],

One of the first approaches in relation extraction uses hand-written regular expression patterns to detect hypernym relations [55]. This approach suffers from low-recall and lack of generalization, as it is practically impossible to accurately derive human patterns that work in every domain. Therefore, several studies have investigated the use of supervised machine learning for this task. This involves labelling data in a corpus with named entities and relations between them and combining various lexical, syntactic and semantic features with machine learning classifiers [51, 119, 136]. Even though these approaches can achieve high levels of accuracy given similar training and test sets, they need expensive data annotation. Furthermore, they are usually biased towards the domain of the text corpus.

Another type of relation extraction utilizes semi-supervised learning. A semi-supervised approach used for relation extraction is bootstrap learning. Given a small number of seed relation instances and patterns, bootstrap learning iteratively finds sentences which contain the seed instances and uses the context of the sentences in order to create new patterns [3, 16, 111]. Because of the small number of initial seeds and patterns, this approach does not achieve high precision and suffers from semantic drift [92]. Another semi-supervised approach used for relation extraction is distant supervision [92, 116]. This approach is different from bootstrapping techniques in that it uses a large knowledge base (e.g. Freebase) to obtain a very large number of examples from which it extracts a large number of features. These features are eventually combined using a supervised classifier. This approach benefits from the fact that it uses very large databases instead of labelled text and in that way manages to overcome overfitting and domain-dependence problems from which supervised methods typically suffer [92].

Other methods approach relation extraction using unsupervised learning [9, 114]. These methods use large amounts of parsed text, extract strings between entities and process these strings in order to produce relation strings. One shortcoming of the unsupervised approaches compared to distant supervision is that the relations they produce might not be compatible with relations that already exist in knowledge bases. This makes the automatic enhancement of knowledge bases non-trivial. A recent unsupervised approach introduces lexical and syntactic constraints in order to produce more informative and coherent relations [36]. Other approaches combine unstructured and structured data sources for relation extraction [108].

In this work we propose a unified approach for relation identification in sentences and sentence ranking that uses some features that are also used for relation extraction. However, our method is less expensive because it does not use heavy-weight features that require complicated linguistic analysis, such as shallow semantic parsing or dependency parsing.

2.5 Learning to rank

Learning to rank (LTR) for information retrieval is a machine learning framework[2] that ranks instances by combining different features (or models) using training data. This framework has been very popular in the research community recently [18, 19, 48, 77]. It has also been used by commercial search engines in order to account for the large number of dimensions of web pages.[3]

There are three main approaches for LTR : pointwise, pairwise and listwise [77]. Below we provide an overview of these approaches and an overview of the specific algorithms used in our experiments.

2.5.1 Pointwise methods

Pointwise methods approach the problem of document ranking indirectly by trying to approximate the true label of the documents. These methods utilize regression, classification, or ordinal regression techniques. The intuition behind pointwise methods is that if the predicted labels are close to the actual labels, then the resulting ranking of the documents will be close to the optimal. In regression techniques, the label of each document is regarded as a real number and the goal is to find the ideal scoring

[2]Since learning to rank algorithms fall into the machine learning framework, in this thesis we use the terms "machine learning" and "learning to rank" interchangeably when referring to such algorithms.

[3]http://blog.searchenginewatch.com/050622-082709

function [27]. Classification techniques try to classify the documents according to their labels [96]. Ordinal regression models try to find a scoring function, the output of which can be used to discriminate between different relevance orders of the documents [28].

We focus on Random Forests (RF) [14] and MART [49], proven to be successful for information retrieval tasks [22]. Both algorithms utilize Classification and Regression Trees (CART) [15]. Random Forests (RF) [14] is a bagging algorithm that combines an ensemble of decision trees. In bagging [15], a learning algorithm is applied multiple times to a sub-sampled set of the training data and the averaged results are used for doing the prediction. RF uses CART as the learning algorithm. At each iteration, it samples a different subset of the training data with replacement and constructs a tree with full depth. The decisions for the best split at each node of the tree are taken by only using a subset of the features. In that way, overfitting is minimized as the individual algorithms are learned using different subsets of the training data. The parameters of RF are the number of trees and the number of features used for finding the best splits. The algorithm is easily parallelizable as the trees created are independent of each other.

Gradient Boosted Regression Trees (GBRT) [49] has been reported as one of the best performing algorithms for web search [19, 22, 93]. It is similar to RF in that it uses the average of different learned decision trees. At each iteration, a tree with small depth is added, focusing on the instances that have the higher current regression error. This is in contrast to RF which uses full depth trees at each iteration. Formally, GBRT performs stohastic gradient descent on the instance space $\{\mathbf{x}_i\}_{i=1}^{n}$, where $\mathbf{x}_i$ is the feature representation of document d_i and y_i is the value of the corresponding label l of the document. $T(\mathbf{x}_i)$ denotes the current prediction of the algorithm for instance $\mathbf{x}_i$. The algorithm assumes a continuous, convex and differentiable loss function $L(T(\mathbf{x}_1), ..., T(\mathbf{x}_n))$ that is minimized if $T(\mathbf{x}_i) = y_i$ for each i. Here we utilize the square loss function $L = \frac{1}{2}\sum_{i=1}^{n}(T(\mathbf{x}_i) - y_i)^2$. The current prediction $T(\mathbf{x}_i)$ is updated at each iteration using:

$$T(\mathbf{x}_i) \leftarrow T(\mathbf{x}_i) - \alpha\frac{L}{T(\mathbf{x}_i)},$$

where *alpha* is the learning rate and the negative gradient $-\frac{L}{T(\mathbf{x}_i)}$ is approximated using the output of the current regression tree for $\mathbf{x}_i$. In our case, this gradient is the difference between the observed and the estimated value of $\mathbf{x}_i$, as we use the square loss function. The other parameters of the algorithm are the learning rate α, the number of iterations and the depth of the tree.

Because of their nature, pointwise methods do not directly consider principles behind IR metrics [77]. They ignore the fact that different documents are associated to different

queries, thus queries with a large number of documents are given more priority during the learning procedure. This can be a problem if the number of documents for each query in the dataset is not balanced. In addition, the loss functions used by the pointwise methods do not directly account for documents ranking for each query. For this reason, they might put too much effort in correctly predicting the labels of documents that have to be positioned lower in the ranking. Therefore, more sophisticated approaches have been proposed that try to address these problems have been proposed, namely the pairwise and the pointwise approaches, which we describe in the next sections.

2.5.2 Pairwise methods

Pairwise methods treat the problem of document ranking as a pair ordering problem. The idea is that if all document pairs are correctly ordered then it is straightforward to combine them and create a perfect ranking of the documents. Pairwise methods are closer than pointwise methods to how IR metrics such as MAP or NDCG measure the quality of the ranking . Thus the problem of ranking is treated as a binary classification problem. All possible document pairs are constructed and labelled for whether the first or the second document should be ranked first according to their relevance to the query (e.g. +1 if the first document in the pair should be ranked higher, -1 otherwise). Therefore the learning goal is to minimize the number of mis-classified pairs. We test two pairwise methods for this task, RankBoost [48] and RankNet [18].

RankBoost [48] is a modification of AdaBoost [47] which operates on document pairs and tries to minimize the classification error. It is a boosting algorithm that maintains a distribution over the document pairs. It starts with a distribution that gives equal weights to all the pairs and iteratively trains "weak" rankers that are used to modify the distribution so that incorrectly classified pairs are given higher weights than the correctly classified ones. Thus it forces the weak ranker to focus on hard queries at future iterations. The final ranking of the algorithm is constructed using a linear combination of the "weak" rankers learned during this procedure.

RankNet [18] tackles the binary classification problem using a neural network. The target probability for each document pair is 1 if it is correctly ordered and 0 if not. During training, each document is associated with a score. The differences of the scores of the two documents in the pair are used to construct the modeled probability. The cross entropy between the target and the modeled probability is used as the error function, so that if the modeled probability is farther from the target probability, the error is larger. Gradient descent is employed to train the neural network. It has been reported that a variation of RankNet was used by Microsoft Live Search for web search [77].

2.5.3 Listwise methods

Listwise methods move one step forward from pairwise methods in terms of modeling the ranking problem as they try to minimize a loss function that directly takes into account the ordering of all the documents in a query. Some of these methods, such as AdaRank [128], try to optimize an approximation or bound of IR metrics, since some metrics are position-based and thus non-continuous and non-differentiable [109]. Other methods such as ListNet [21], utilize error functions that measure ranking differences across lists. Here, we examine four listwise algorithms : AdaRank [128], Coordinate Ascent [86], LambdaMART [127] and ListNet [21].

AdaRank [128] is a boosting algorithm based on AdaBoost. It follows the boosting idea that we described previously for RankBoost but operates on query-documents lists, thus the distribution is over the queries. It optimizes a listwise error function which is usually an IR metric such as MAP or NDCG.

Coordinate Ascent (CoordAscent) [86] is also a listwise algorithm that directly optimizes IR metrics. It is a linear feature-based method that uses the coordinate ascent optimization technique. This procedure optimizes a single document feature repeatedly, keeping the rest unchanged each time. In addition, it uses random restarts in order to increase the likelihood of arriving to a global maximum.

LambdaMART [127] is a combination of GBRT [49] - also called MART (multiple additive regression trees) - and LambdaRank [104]. LambdaRank [104] uses an approximation to the gradient of the cost by modeling the gradient for each document in the dataset with lambda functions, called λ-gradients. This is done because costs like NDCG are non-continuous and therefore the gradient of the cost cannot be optimized directly. Assuming that NDCG is being optimized, the λ-gradient for a document that belongs in a document pair is computed using both the cross entropy loss and the gain in NDCG that we will have if we swap the current order of the document pair (note that cross entropy loss is also used by RankNet). Therefore, if a metric such as NDCG is optimized, not all document pairs will have the same importance during training, as this will not only depend on their labels but also on the document order for each query. LambaMART builds on MART, the main difference of them being that LambdaMART computes the derivatives as LambdaRank does. At each iteration, the tree being constructed models the λ-gradients of the entire dataset, thus focusing on the overall performance of all queries in the dataset. For more details on this algorithm, please refer to [127].

A listwise algorithm that does not directly optimize an IR metric is ListNet [21], which uses a neural network as a model. This algorithm is similar to RankNet (described before), the important difference between the two being that ListNet employs a listwise

instead of a pairwise loss function. The intuition behind this algorithm is that the problem of ranking a set of documents can be mapped to the problem of finding the correct permutation of the documents. During training, each document is given a score and the algorithm employs the Luce model to define a probability distribution over the possible distributions of the documents using the document scores [101]. A second, target probability distribution is constructed using the true labels of the documents. The training goal is to minimize the KL divergence between the first and the second probability distributions. Gradient descent is used to train the neural network.

In this work, we combine methods and ideas inspired from the research areas described in this chapter for the task of "Entity Relationship Explanation". Section 3 provides a detailed description of the proposed method.

Chapter 3

Method

We try to build automatic methods for explaining pre-defined relations of entity pairs in the context of search engines, a problem which we named *Entity Relationship Explanation* in Chapter 1. To this end, we utilize a dataset of a major commercial search engine that contains entities and relations between them. We describe this dataset in Section 4.1.1.

The problem we are trying to solve can be split in two parts. The first is to extract sentences from a document corpus that refer to the entity pair and the second is to rank these sentences based on how well they describe a pre-defined relation of the entity pair. More formally, given two entities e_a and e_b and a relation r between them, the task is to extract a set of candidate sentences $S = \{s_i\}$ that refer to e_a and e_b and to provide the best ranking for the sentences in S. Relation r has the general form : $type(e_a)\_terms(r)\_type(e_b)$, where $type(e)$ is the type of the entity e (e.g. "Person","Actor") and $terms(r)$ are the terms of the relation (e.g. "CoCastsWith", "IsSpouseOf"). The main notation we use is summarized in table 3.1.

TABLE 3.1: Main notation used in this work.

Notation	Explanation
e_a	the first entity of the entity pair.
e_b	the second entity of the entity pair.
r	the relation of interest between e_a and e_b.
S	a set of candidate sentences possibly referring to e_a and e_b.

In this chapter we describe how we extract and enrich the candidate sentences and how we tackle the sentence ranking problem.

3.1 Extracting sentences

We aim to find sentences that refer to the two entities e_a and e_b and to eventually rank the sentences according to how well they describe the relation of interest r. In this section we describe how we create the candidate set of sentences S and the way we enrich the representation of the sentences with entities.

A natural source to extract sentences from is Wikipedia, a widely used semi-structured knowledge base that provides good coverage for the majority of the entities. We hypothesize that if both entities of an entity pair have a Wikipedia article, then it is likely that a sentence related to their relation is included in one or more articles. In order to achieve good coverage, we use three different text representations of entities: the title of the Wikipedia article of the entity (e.g. "Barack Obama"), the labels that can be used as anchor in Wikipedia to link to it (e.g. "president obama") and the titles of the redirect pages that point to this entity's Wikipedia article (e.g. "Obama"). A sentence is included in the candidate sentences set if it is found in the article of either e_a or e_b and it contains the title, a label and/or a redirect of the other entity in the entity pair. A sentence that contains the title, a label and/or a redirect of both entities in the entity pair is also included in the candidate sentences set.

3.1.1 Sentences enrichment

As our ranking task is based on entities, it is natural to augment the sentences representation with entities in order to take advantage of information from external knowledge bases when ranking. To this end, we first perform co-reference resolution at the document level in order to replace the n-grams that refer to the entity with the title of the entity and then perform entity linking on each sentence of the document [87, 91].

3.1.1.1 Co-reference resolution

Co-reference resolution is the task of determining whether two mentions are co-referent [71, 72]. In our setting, we want to match the n-grams that refer to the entity of interest in a document in order to be able to link these n-grams to the entity in the knowledge base and also make the sentences self-contained. For example, if we are interested in the entity "Brad Pitt", the Wikipedia article of this entity contains the sentence *"He gave critically acclaimed performances in the crime thriller Seven..."*. We therefore need a way of identifying the referent of "He", in this case "Brad Pitt". The same need appears for other cases as well, such as referring to "Toyota" as "the company" or to "Gladiator" as "the film".

We have experimented with the Stanford co-reference resolution system [71] and the Apache OpenNLP tool[1] and found that these systems were not able to consistently achieve the desired behaviour for people entities in Wikipedia, which are the ones we study in this work. Therefore, we devised a simple heuristic algorithm targeted specifically to our problem. Since we are only interested in people entities, we count the appearances of "he" and "she" in the article in order determine whether the entity is male or female. We then replace the first appearance of "he" or "she" in the sentence with the entity title. In order to avoid having multiple occurrences of n-grams referring to the same entity in the sentence, we skipped the replacement when a label of the entity is already contained in the sentence.

3.1.1.2 Entity linking

In order to augment each sentence with entities, we need to perform entity linking, which is the task of linking free text to knowledge base entities [87, 91]. In a document retrieval scenario using Wikipedia articles, this would not be needed, as the articles already contain links to entities. However, the linking guidelines for Wikipedia articles only allow one link to another article in the article's text.[2] For this reason, not every sentence in an article contains links to the entities it mentions and thus it is not possible to derive features dependent on entities. We describe the entity-dependent features we devised for this task in the next section.

We employ a simple heuristic algorithm to perform entity linking in Wikipedia articles at the sentence level. We restrict the candidate set of entities to the article itself and the other articles that are already linked in the article. By doing this, no disambiguation is performed and our linked entities are very unlikely to be wrong. The algorithm takes as input the sentence annotated with the already linked entities and finds the n-grams that are not already linked. Then, if the n-gram is used as an anchor of a link in Wikipedia and it can be linked to a candidate entity, we link the n-gram to that entity.

Even though we do not evaluate these two components, we have observed that they perform reasonably well in the end-to-end task. However, the co-reference resolution component may produce grammar mistakes in some cases.

[1] https://opennlp.apache.org/
[2] http://en.Wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking

3.2 Ranking sentences

After extracting the candidate sentences S, we need to rank them by how well they describe the relation of interest r of entities e_a and e_b. A naive approach for ranking the sentences would be to use the title of each entity and the relation of interest as a query and tackle the problem using state-of-the-art information retrieval models. Below we briefly overview some of these models.

A classic vector space model is term frequency - inverse document frequency (tf-idf) [7], which scores a document d with respect to a query $q = \{q_1, \ldots, q_k\}$ as follows:

$$score(d, q) = \sum_i tf(q_i, d) \cdot idf(q_i, C),$$

where $tf(q_i, d)$ is the number of times q_i appears in document d, and

$$idf(q_i, C) = log \frac{|C|}{|\{d \in C : q_i \in d\}|}, \tag{3.1}$$

where $|C|$ is the size of the document collection and $|\{d \in C : q_i \in d\}|$ is the number of documents in the collection that contain the term q_i.

Language modeling for information retrieval estimates $p(d|q)$, which is the conditional probability that a document d generates the query q [102]. After applying Bayes' rule, we have:

$$p(d|q) = \frac{p(d)p(q|d)}{p(q)},$$

where $p(d)$ is the document's prior probability, $p(q|d)$ is the likelihood of the query given the document and $p(q)$ is the probability of the query. $p(q)$ is ignored as it is independent of the document and therefore not useful for ranking. By using a uniform document prior we have:

$$p(d|q) = p(q|d) = \prod_i p(q_i|d)$$

When dirichlet smoothing is used [135], the probability of each term q_i given the document d is estimated as follows:

$$p(q_i|d) = \frac{tf(q_i, d) + \mu \cdot p(q_i|C)}{|C| + \mu},$$

where $tf(q_i, d)$ is the number of times q_i appears in document d, $p(q_i|C)$ is the background collection language model, $|C|$ is the size of collection C and μ is a smoothing parameter.

BM25 [118] scores a document d with respect to a query q as follows:

$$score(d, q) = \sum_i idf(q_i, C) \cdot \frac{tf(q_i, d) \cdot (k + 1)}{tf(q_i, d) + k \cdot (1 - b + b \cdot \frac{|d|}{avgDocLength(C)})},$$

where $tf(q_i, d)$ is the number of times q_i appears in document d, $idf(q_i, C)$ is the inverse document frequency of q_i in collection C (see equation 3.1), $|d|$ is the length of document d and $avgDocLength(C)$ is the average document length in collection C. k and b are free parameters.

In our setting we have a sentence retrieval scenario, where the collection C consists of sentences and therefore the retrieval unit is a sentence s instead of a document d.

However, the candidate sentences have attributes that might be important for ranking but cannot be captured by retrieval models, which are solely based on terms. The same problem can be found in web search, where a web page importance can be measured by features not dependent on terms or the query, such as links [99]. Therefore, a way of combining different features is needed. This has been the subject of research in previous years which has led to algorithms for learning to rank which proved to be very successful for web search and other information retrieval applications [1, 19, 120]. We follow the learning to rank framework and represent each sentence s with a rich set of features $F = \{f_i\}$ that aim to capture different dimensions of the sentence and we use learning to rank algorithms in order to combine them.

In this section we provide an overview of the learning to rank framework and describe the features we devised for our task.

3.2.1 LTR framework

In Section 2.5 we provided an overview of LTR and described the different approaches (pointwise, pairwise, listwise) and some important algorithms. Here we give an overview of the LTR framework and how use it for our task.

In the LTR framework [77], the training data consists of a set of queries $\{q_i\}$ each of them being associated with a set of documents $\{d_j\}$. Each document is represented by a set of features F and is judged with a label l. This label declares the degree of relevance of document d_j with respect to the query q_i. A LTR algorithm learns a model that combines the document features and is able to predict the label l of the documents in

the training data. The prediction accuracy is measured using a loss function. When a new query comes in, the documents are ranked by the degree of relevance to the query using the learned model.

While LTR is usually applied on documents, there is no restriction for applying it on different kinds of instances. In fact, LTR has been successfully employed for question answering [1, 120]. That task can be regarded similar to ours, as it can also contain a sentence ranking component. The formulation of our problem makes LTR a natural solution as we represent our sentences by feature vectors that capture different dimensions, as described in the previous section. The only difference between our framework and the classic LTR framework is that our retrieval unit is a sentence instead of a document.

3.2.2 Features

Table 3.2 contains an overview of the features F and below we describe each feature in detail. In this section we group the features per type and provide a detailed description of each feature.

3.2.2.1 Text features

This feature type regards the importance of the sentence s at the term level. A very basic feature is the length of the sentence, $Length(s)$, which is calculated at the term level. A classic way of measuring term importance in a corpus is inverse document frequency, *idf*, calculated as in Equation 3.1. We use Wikipedia as a background corpus and calculate *idf* for every term t in s and include the average *idf* of the terms in s:

$$AverageIDF(s) = \frac{1}{|t|} \sum_{t \in s} idf(t, C) \tag{3.2}$$

where $|t|$ is the number of terms in s. We also include the sum of *idf* of the terms in s:

$$SumIDF(s) = \sum_{t \in s} idf(t, C) \tag{3.3}$$

Density-based selection was originally proposed in the field of question answering in order to rank sentences [70] and was also used in comment-oriented blog summarization for sentence selection [59]. In our setting, we treat stop words and numbers in s as non-keywords, and the rest terms in s as keywords. We calculate the density of s as

TABLE 3.2: Features used to represent each sentence, grouped by feature type.

Text features	
$AverageIDF(s)$	Average IDF of terms of s in Wikipedia, see Equation 3.2.
$SumIDF(s)$	Sum of IDF of terms of s in Wikipedia, see Equation 3.3.
$Length(s)$	Number of terms in s.
$Density(s)$	Lexical density of s, see Equation 3.4.
$POS(s)$	Part of Speech distribution of s.
Entity features	
$NumEntities(s)$	Number of entities in s.
$ContainsLink(e, s)$	Whether s contains a link to the entity e, calculated for both e_a and e_b (binary).
$ContainBothLinks(e, s)$	Whether s contains links to both both e_a and e_b.
$AverageInLinks(s)$	Average in-links counts of the entities in s.
$SumInLinks(s)$	Sum of in-links counts of the entities in s.
$Spread(e_a, e_b, s)$	Distance between e_a and e_b in s.
$BetweenPOS(e_a, e_b, s)$	Part of Speech distribution between e_a and e_b in s.
$LeftPOS(e_a, e_b, s)$	Part of Speech distribution left of the entity (either e_a or e_b) found first in s (left window) .
$RightPOS(e_a, e_b, s)$	Part of Speech distribution right of the entity (either e_a or e_b) found first in s (right window).
$NumEntitiesLeft(e_a, e_b, s)$	Number of entities in s in the left window of the entity found first in s (either e_a or e_b).
$NumEntitiesRight(e_a, e_b, s)$	Number of entities in s in the left window of the entity found first in s (either e_a or e_b).
$NumEntitiesBetween(e_a, e_b, s)$	Number of entities between e_a and e_b in s.
$ContainsCommLinks(e_a, e_b, s)$	Whether s contains one of the top-k links shared between the entity pair (binary).
$NumCommLinks(e_a, e_b, s)$	Number of top-k links shared between the entity pair in s.
Relation features	
$MatchTerm(r, s)$	Whether s contains any term of r (binary).
$MatchSyn(r, s)$	Whether s contains any phrase in $wordnet(r)$ (binary).
$Word2vecScore(r, s)$	Average score of phrases in $word2vec(r)$ that are matched in s.
$MaxWord2vecScore(r, s)$	Score of the phrase with the maximum score in $word2vec(r)$ that is matched in s.
$MatchTermOrSyn(r, s)$	Whether s contains any term of r or any phrase in $word2vec(r)$ (binary).
$MatchTermOrWord2vec(r, s)$	Whether s contains any term of r or any phrase in $word2vec(r)$ (binary).
$MatchTermOrSynOrWord2vec(r, s)$	Whether s contains any term of the relation r, any phrase in $wordnet(r)$ or any any phrase in $word2vec(r)$ (binary)
$ScoreLC(e_a, e_b, r, s)$	Lucene score of s with $\{e_a, e_b, r, wordnet(r), word2vec(r)\}$ as the query, using $Title(e)$, $Redirects(e)$ or $Labels(e)$ to represent the entities e_a and e_b (3 features).
$ScoreBM25(e_a, e_b, r, s)$	BM25 score of s. The query is constructed as above.
Source features (Wikipedia)	
$Position(s, d(s))$	Position of s in document d.
$SentenceSource(e, d(s))$	Whether sentence's s document d is the entity's Wikipedia article, calculated for both e_a and e_b (binary).
$DocCount(e, d(s))$	Number of occurrences of e in sentence's s document d, calculated for both e_a and e_b.

follows:

$$Density(s) = \frac{1}{K \cdot (K+1)} \sum_{i=1}^{n} \frac{score(t_j) \cdot score(t_{j+1})}{distance(t_j, t_{j+1})^2}, \tag{3.4}$$

where K is the number of keyword terms t in s, $score(t)$ is equal to $IDF(t)$ and $distance(t_j, t_{j+1})$ is the number of non-keyword terms between keyword terms t_j and t_{j+1}.

Part-of-speech distribution is frequently used as a feature in relation extraction applications [92]. We calculate the POS distribution of the whole sentence $POS(s)$ and use the percentages of verbs, nouns, adjectives and others in s as features.

3.2.2.2 Entity features

Here we consider features that concern the entities of the sentence and are therefore dependent on a knowledge base (in our case Wikipedia). First, we consider the number of entities in the sentence, $NumEntities(s)$.

The number of links that point to an entity in Wikipedia is an indication of popularity or importance of the entity in the Wikipedia graph. We calculate this for every entity in s and include $AverageInLinks(s)$ and $SumInLinks(s)$.

We include $ContainsLink(e, s)$, which is an indicator of whether s contains a link to either e_a or e_b. $ContainBothLinks(e, s)$ indicates whether both e_a and e_b are contained in s. We also calculate the distance of the entities of the pair in the sentence, $Spread(e_a, e_b, s)$, a feature also included in a closely related application [11].

Apart from the POS distribution of the sentence we discussed previously, we also consider the POS between the entities e_a and e_b and POS on the left/right window of e_a or e_b (depending on which appears first) [92]. We also include the number of entities between ($NumEntitiesBetween(e_a, e_b, s)$), to the left ($NumEntitiesLeft(e_a, e_b, s)$) or to the right ($NumEntitiesRight(e_a, e_b, s)$) of the entity pair. The max length of the window is set to 4. Note that these features are calculated only when both entities are included in the sentence.

Semantic similarity measures such as [126] assume that if two articles in Wikipedia have many common articles (links) that point to them, then it is likely that the two are strongly related. We hypothesize that if a sentence contains common links of e_a and e_b, the sentence might contain some important information about their relation. An example of this is the entity pair “Lionel Messi” - “Neymar”, for which “Barcelona FC”

is a common link. We score the common links between e_a and e_b and using the following heuristic scoring function:

$$score(l, e_a, e_b) = similarity(l, e_a) * similarity(l, e_b),$$

where l is the link examined and the similarity between two articles a_1 and a_2 in Wikipedia is calculated using a version of Normalized Google Distance based on Wikipedia links [126]:

$$similarity(a_1, a_2) = \frac{log(max(|A_1|, |A_2|)) - log(|A_1 \cap A_2|)}{log(|W|) - log(min(|A_1|, |A_2|))},$$

where A_1 and A_2 are the sets of articles in Wikipedia that link to A_1 and A_2 respectively and W is the set of articles in Wikipedia. We then select the top k links (we set $k = 30$) in order to calculate $ContainsCommLinks(e_a, e_b, s)$, which indicates whether one of the top-k entities is contained in the sentence and $NumCommLinks(e_a, e_b, s)$, which indicates the number of these common links.

3.2.2.3 Relation features

This feature type regards features that are aware of the relation of interest r between e_a and e_r. As described before, relation r has the general form : $type(e_a)\_terms(r)\_type(e_b)$, where $type(e)$ is the type of the entity e (e.g. "SportsAthlete") and $terms(r)$ are the terms of the relation (e.g. "PlaysSameSportTeamAs").

$MatchTerm(r, s)$ indicates if any of the relation terms is contained in the sentence, excluding the entity types. However, matching only the terms in the relation has low coverage. For example, phrases "husband" or "married to" are more likely to be contained in a sentence describing the relation "Person_isSpouseOf_Person" than the relation term "spouse". To this end, we employ Wordnet [38] in order to get phrases similar to each relation. Similar ideas were investigated for sentence retrieval when constructing monolingual translation tables [94] and for relation extraction [116]. When obtaining phrases similar to relation r from Wordnet we use synonyms of the relation terms only, without taking into account the entity types. For example, we only use synonyms of the term "spouse" when we obtain synonyms of the relation "Person_IsSpouseOf_Person". We refer to the set of Wordnet synonym phrases of r as $wordnet(r)$. $MatchSyn(r, s)$ indicates whether the sentence matches any of the synonyms in $wordnet(r)$ and $MatchTermOrSyn(r, s)$ indicates whether $MatchTerm(r, s)$ or $MatchSyn(r, s)$ is true.

We explore another way of obtaining phrases similar to the relation r by employing an unsupervised algorithm that can be used for measuring semantic similarity among

words or phrases [88, 90]. This algorithm has attracted a lot of attention recently in the research community [10, 63, 89]. The algorithm takes a text corpus as input and learns vector representations of words consisting of real numbers using the continuous bag of words or the skip-ngram architectures [88]. It has been shown that these vectors can be used for measuring semantic similarity between words by employing the cosine distance of two vectors and thus they can be used for analogical reasoning tasks. Another important characteristic of this algorithm is that multiple vectors can be added element-wise and the resulting vector can represent the "combined meaning" of the individual vectors. An example demonstrating this characteristic taken from [90] is that the closest vector of the combination of the vectors of "Vietnam" and "capital" is "Hanoi", as one would expect. A simple algorithm that accounts for word co-occurrence can be used in order to obtain vector representations for phrases [90]. For the rest of this thesis, we refer to the phrase vectors learned with this algorithm as *word2vec* vectors.

In order to compute the most similar phrases to the relation r using the *word2vec* vectors, we select terms both from the relation terms and the entity types of the two entities in the relation, excluding the entity type "person" which proved to be very broad and not informative. We then compute the distance between the vectors of all the candidate phrases in the data and the vector resulting from the element-wise sum of the vectors of the relation terms.

More formally, given the set V_r which consists of the vector representations of all the relation terms and the set V which consists of the vector representations of all the candidate phrases in the data, we calculate the distance between a candidate phrase represented by a vector $\mathbf{v}_i \in V$ and all the vectors in V_r as follows:

$$distance(\mathbf{v}_i, V) = cosine\_sim(\mathbf{v}_i, \sum_{\mathbf{v}_j \in V_r} \mathbf{v}_j), \tag{3.5}$$

where $\sum_{\mathbf{v}_j \in V_r} \mathbf{v}_j$ is the element-wise sum of all the vectors in V_r and the distance between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is measured using cosine similarity, which is calculated as:

$$cosine\_sim(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1| \cdot |\mathbf{v}_2|},$$

where the numerator is the dot product of $\mathbf{v}_1$ and $\mathbf{v}_2$ and the denominator is the product of the euclidean lengths of $\mathbf{v}_1$ and $\mathbf{v}_2$. The candidate phrases in V are then ranked using Equation 3.5 and the top m phrases are selected. We refer to the ranked set of phrases that are selected using this procedure as $word2vec(r)$, where $|word2vec(r)| = m$.

Below we illustrate what phrases this procedure suggests with an example. For relation "MovieDirector_Directs_MovieActor", we compute the distance between the vectors of

the candidate phrases and the vector resulting from the element-wise sum of the vectors of the relation terms "movie", "director", "directs" and "actor". The top 10 phrases suggested for this relation were "film", "cast", "actors", "starring", "films", "movies", "starred", "feature film", "costars" and "lead role".

We include several features which are computed using the most similar phrases to r according to *word2vec*. $Word2vecScore(r, s)$ is the average score (cosine similarity) of phrases in $word2vec(r)$ that are matched in s, $MaxWord2vecScore(r, s)$ is the score of the phrase with the maximum score (cosine similarity) in $word2vec(r)$ that is matched in s. $MatchTermOrWord2vec(r, s)$ indicates whether s contains any term of r or any phrase in $word2vec(r)$. $MatchTermOrSynOrWord2vec(r, s)$ indicates whether s contains any term of the relation r, any phrase in $wordnet(r)$ or any any phrase in $word2vec(r)$.

In addition, we employ state-of-the-art information retrieval ranking functions and include the sentences scores for query q, which is constructed using the entities e_a and e_b, the relation r, $wordnet(r)$ and $word2vec(r)$. We add one feature for each way of representing the entities e_a and e_b : the title of the entity articles $Title(e)$, the titles of the redirect pages of the entity article, $Redirects(e)$, and the n-grams used as anchors in Wikipedia to link to the article of the entity, $Labels(e)$. This produces 3 features per ranking function. As for the ranking functions, we score the sentences using the Lucene scoring function[3] and Okapi BM25 [110].

3.2.2.4 Source features

By source features we refer to features that are dependent on the source document of the sentences. We consider the position of the sentence in the document, $Position(s, d(s))$ and $SentenceSource(e, d(s))$, which indicates whether sentence s originates from entity e_a or entity e_b. We also consider the number of occurrences of e_a and e_b in sentence's s document d, $DocCount(e, d(s))$, a feature inspired by document smoothing for sentence retrieval [94]. The intuition here is that if an entity is found multiple times in a document, then the sentence found in that document might be more important for that entity.

[3] https://lucene.apache.org/core/4_3_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Chapter 4

Experimental setup

In this chapter we provide details on dataset construction, the annotation procedure and the evaluation metrics that we used in order to design our experiments. We also provide experimental details for the learning to rank algorithms we experimented with.

4.1 Dataset

4.1.1 Entity pairs

We focus on "people" entities and relations between them, which we obtain by utilizing an automatically constructed dataset which contains entity pairs and their relations. This dataset is used in production by the Yahoo web search engine [12] and it was constructed by combining information about entities and their relations from various data sources, including Freebase, Wikipedia and IMDB. Note that our methods are independent of the restriction on "people" entities, except from the co-reference resolution step described in Section 3.1.1.

Because of the vast size of that dataset, we pick 90 entity pairs from this dataset to construct our experimental dataset. 21 pairs were manually picked as use cases (e.g. "Brad Pitt - Angelina Jolie (Person_IsPartnerOf_Person)"), while 39 pairs were randomly sampled. For the remaining pairs, we tried not to overemphasize on entities that are either very popular or rare among the search engine users. In order to measure this, we needed a popularity distribution over the entities. To construct this distribution we utilized nine months of query logs of the U.S. Yahoo web search engine and counted the number of times a user clicks the link of the Wikipedia article of an entity in the results page. We then filtered the entity pairs set so that both entities of each pair

appear between the mean and one standard deviation above or below the mean of the popularity distribution. We then sampled 30 pairs from the resulting set.

We extracted our sentences dataset using the approach described in Section 3.1. This procedure extracts 724 sentences in total for the 90 pairs in our dataset. The average number of sentences per pair is 8.04. The maximum number of sentences for a pair is 40 and the minimum is 2.

In order to compute vector representations of phrases and the distance between them, we use a publicly available software package.[1] The model is trained on text extracted from a full Wikipedia dump consisting of approximately 3 billion words using negative sampling and the continuous bag of words architecture [88, 90]. The size of the phrase vectors is set to 500. The trained phrase vectors achieved 75% accuracy on the analogical reasoning task for phrases described in [90]. We use the phrase vector representations of the learned model to construct the phrase set $word2vec(r)$ for each relation r, as explained in Section 3.2.2.3. The size of $word2vec(r)$ is set to $m = 30$.

4.1.2 Sentences preprocessing

We preprocessed Wikipedia with wikipedia-miner[2] in order to extract the sentences and their corresponding features. We performed co-reference resolution and entity linking on the sentences using the heuristics described in Section 3.1.1. The sentences were POS-tagged with the Stanford part-of-speech tagger [121]. We filter out stop words using the Lucene list of English stop words.[3]

4.1.3 Wikipedia

We use an English Wikipedia dump dated March 2, 2014. This dump contains 4,342,357 articles. We used Wikipedia both as a corpus to extract sentences from and as a knowledge base. We indexed Wikipedia both at the sentence level (for extracting sentences) and at the article level (for obtaining term statistics).

4.2 Annotations

Two human annotators were involved in providing relevance judgements for the sentences in our dataset. For every entity pair they annotated, the annotators were presented with

[1] https://code.google.com/p/word2vec/
[2] http://Wikipedia-miner.cms.waikato.ac.nz
[3] http://lucene.apache.org

the Wikipedia articles of the two entities and the relation of interest that we wanted to explain using the extracted candidate sentences. The sentences were judged based on how well they describe the relation of interest on a five level graded relevance (perfect, excellent, good, fair, bad). A perfect or an excellent sentence should describe the relation of interest at a satisfactory level, but a perfect sentence is relatively better for presenting it to the user. A good or a fair sentence indicates a sentence which is about another aspect of their relation, not necessarily related to the relation of interest.

The first annotator provided relevance judgments for the entire dataset. In order to answer research question (RQ4), which examines how difficult this task is for the human annotators, we decided to have a subset of the dataset annotated by the second human annotator, who annotated one third of the dataset. The kappa coefficient of inter-annotator agreement is $k = 0.314$, which is considered as a fair agreement. When weighted kappa [45] is used, the agreement measure is $k = 0.503$, which shows moderate agreement. We noticed that the first annotator was stricter and that one of the main disagreements between the two annotators was whether a sentence was perfect or excellent for describing the relation. We conclude that the task is not easy for the human annotators.

The overall relevance distribution of the sentences in the dataset is : 13.12% perfect, 7.6% excellent, 26% good, 28.31% fair and 24.31% bad. Out of 90 entity pairs, 81 of them have at least one sentence annotated as excellent and 66 of them have at least one sentence annotated as perfect.

4.3 Evaluation metrics

We evaluate the performance of our methods in order to answer research question (RQ1) in two different scenarios. In the first scenario, we want to show a single sentence to the user which would be able to describe the relation of interest of an entity pair. Therefore, we prioritize having the most relevant sentence at the top of the ranking. For this case we report on NDCG@1 [60], ERR@1 [23] and reciprocal rank for perfect or excellent (perfectRR, excellentRR). We also report on excellent@1 which indicates whether we have an excellent or a perfect sentence at the top of the ranking. In addition, we report on perfect@1 which indicates whether we have a perfect sentence at the top of the ranking. Note that not all entity pairs have an excellent or a perfect sentence.

We also consider another scenario, where the user is not only interested in the best sentence that describes the relation of interest between two entities but also in having more information about the entity pair. This information might include more details about

the relation of interest or different relations. Here we report on NDCG@10 [60] and ERR@10 [23]. Note that an ideal ranking can eventually be used as input of a summarization system, designed for aggregating the ranked sentences in a succinct paragraph.

We perform 5-fold cross validation and test for statistical significance using a paired, two-tailed t-test. We depict a significant increase in performance when against a single baseline for $p < 0.01$ with ▲ (▼ for a decrease) and for $p < 0.05$ with △ (▽ for a decrease).

4.4 LTR algorithms

Research Question (RQ1) considers the effect of retrieval models and learning to rank methods on retrieval performance. Here we report the learning to rank algorithms and the default parameters of these algorithms used for answering (RQ1).

As discussed in Section 2.5, LTR approaches can be categorized to pointwise, pairwise and listwise [77]. We consider at least two algorithms from each category for this task: RF and GBRT (pointwise), RankBoost and RankNet (pairwise) and AdaRank, CoordAscent, LambdaMART and ListNet (listwise). For our experiments, we use the RankLib[4] implementation of the above algorithms with the default parameters without tuning or feature normalization, unless otherwise specified.

For RF, we set the number of iterations to 300 and the features sampling rate to 0.3. We set the number of trees for GBRT and LambdaMART to 1000, the number of leaves for each tree to 10, the learning rate to 0.1 and we use all threshold candidates for tree splitting. For RankBoost, we train for 300 rounds and use 10 threshold candidates. The neural network of RankNet consists of 1 hidden layer and 10 nodes per layer, the number of epochs is set to 100 and the learning rate to 0.00005. For AdaRank the number of training rounds is set to 500, the tolerance between two consecutive learning rounds is set to 0.002. A feature can be consecutively selected without change in performance for a maximum of 5 times. We set the number of random restarts for CoordAscent to 5, the number of iterations to search in each dimension to 25 and the performance tolerance between two solutions to 0.001. We don't use regularization for CoordAscent.

We choose RF as our main learning algorithm, as it outperformed the rest of the algorithms we examine in most of our preliminary experiments. Moreover, it is insensitive to parameter settings, resistant to overfitting and parallelizable. We refer to this as the full machine learning algorithm. Note that the results vary slightly for different runs of RF and CoordAscent. For this reason, unless otherwise specified, we report on the average of 5 runs for these two algorithms. We provide a comparison of all the algorithms in

[4] http://sourceforge.net/p/lemur/wiki/RankLib/

Section 5.4, where we also analyse the effect of variance in results for different runs for RF, GBRT, CoordAscent and LambdaMART.

Chapter 5

Results and discussion

In this section we describe and analyse the results of the experiments we conducted in order to answer our research questions introduced in Section 1.1. We present results for the baselines and the full model, provide a feature analysis, analyse failure cases and benchmark different machine learning models for this task. Finally, we provide a comparison of our method to a competing system.

5.1 Baselines

A straightforward way to rank the candidate sentences would be to use state-of-the-art information retrieval models and model our problem at the term level. We employ the Lucene ranking function (LC)[1], which is a heuristic function based on the vector space model, language modeling with dirichlet smoothing (LM) [135] and BM25 [118]. We use the default values for BM25 ($k = 1.2$, $b = 0.75$) and LM ($\mu = 2000$).

The query q can be constructed using representations of the two entities e_a and e_b and of the relation r. Each entity in the entity pair can be represented in the query by the title of the entity article, $Title(e)$, the titles of the redirect pages of the entity article, $Redirects(e)$ or the n-grams used as anchors in Wikipedia to link to the article of the entity, $Labels(e)$. The relation r can be represented by the terms in the relation, by the synonyms in $wordnet(r)$ or by the phrases in $word2vec(r)$.

We explore several ways of constructing the query q, all of them including a representation of the entities e_a and e_r and differing in the way of representing the relation r. Baseline B1 does not include any representation of r in the query. B2 includes

[1] https://lucene.apache.org/core/4_3_1/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

TABLE 5.1: Results for the best performing combination of each baseline. Boldface marks best performance in the respective metric.

Method	NDCG@1	NDCG@10	Perfect@1	Excellent@1	ERR@1	ERR@10	PerfectRR	ExcellentRR
B1	0.4970	0.7535	0.2889	0.4444	0.3965	0.5654	0.4345	0.6007
B2	0.5051	0.7544	0.2778	0.4444	0.3938	0.5647	0.4290	0.6007
B3	0.5232	0.7623	0.2778	0.4667	0.4028	0.5692	0.4259	0.6089
B4	**0.5659**	0.7810	**0.3222**	**0.5110**	**0.4389**	**0.5975**	**0.4618**	**0.6523**
B5	0.5587	**0.7842**	0.3111	0.5000	0.4306	0.5943	0.4565	0.6430

the relation terms of r, while B3 includes the relation terms of r and the synonyms in $wordnet(r)$. B4 includes the terms of r and the phrases in $word2vec(r)$, and B5 includes both the the synonyms in $wordnet(r)$ and the phrases in $word2vec(r)$. We combine each of the possible representations of the entities, $Title(e)$, $Redirects(e)$ and $Labels(e)$ with the different ranking functions, thus having 9 possible scores for each way of constructing the query.

Table 5.1 shows the best performing combination of the above for B1, B2, B3, B4 and B5. We omitted the rest combinations from this table because the results were not informative. The best results for each baseline were obtained using the labels of the entities in the query. This is reasonable, as the labels encode the most likely phrases for referring to the entities, thus having better coverage compared to the titles and the redirects. Interestingly, the best combination of all the baselines used the Lucene ranking function, which outperformed BM25 and LM. However, the difference was significant only compared to LM. Note that as our retrieval unit is a sentence, the Lucene ranking function becomes similar to the tf-isf function. As discussed in Section 2.2, tf-isf is considered as a competitive model for sentence retrieval.

As we can observe from table 5.1, B4 and B5 are the best performing baselines. This suggests that adding the phrases in $word2vec(r)$ to the query is beneficial. Also, adding the Wordnet synonyms of the relation to the query can help in some cases. Although a t-test does not account for multiple comparisons and thus reporting significant differences using it might not give safe conclusions when comparing to multiple baselines, we report some complementary significance results. B4 significantly outperforms B1, B2 and B3 ($p < 0.05$) whereas B5 significantly outperforms B1 and B2 for all metrics ($p < 0.05$). However, there is no significant difference between B4 and B5 for any metric.

In order to achieve stronger baselines, we employ machine learning and use the scores of different combinations of the baselines as features. However, this model was not able to outperform any of the best performing combination of each baseline, probably because the features were correlated.

TABLE 5.2: Results for the best performing baselines and the full machine learning model. Boldface marks best performance in the respective metric. Significance is tested against Full-ML.

Method	NDCG@1	NDCG@10	Perfect@1	Excellent@1	ERR@1	ERR@10	PerfectRR	ExcellentRR
Full-ML	**0.7448**	**0.8719**	**0.5222**	**0.6667**	**0.5931**	**0.7024**	**0.6039**	**0.759**
B4	0.5659▼	0.7810▼	0.3222▼	0.5110▼	0.4389▼	0.5975▼	0.4618▼	0.6523▼
B5	0.5587▼	0.7842▼	0.3111▼	0.5000▼	0.4306▼	0.5943▼	0.4565▼	0.6430▼

5.2 Full machine learning model

In this section we provide the results obtained using machine learning (RF) by utilizing all the features described in Section 3.2.2.

5.2.1 Comparison to the baselines

This experiment regards research question (RQ1), which investigates the most effective method for explaining relations between entities. We use B4 and B5 as baselines as they achieved the best scores both for NDCG@1 and NDCG@10 in the previous experiment. We use RF as our learning algorithm, but also provide a benchmark of different machine learning algorithms in Section 5.4.

Table 5.2 shows the results of this experiment. It can easily be derived that the full machine learning model is the best performing method, as both B4 and B5 have significantly worse performance than the full model by a large margin.

Figure 5.1 shows the difference on NDCG@1 per entity pair between B5 and Full-ML. We do not include the corresponding plot for differences between B4 and Full-ML because we observed almost identical trends. It is clear from this plot that Full-ML is a more suited method for this task in most of the cases. Full-ML performance is worse than B5 for only 11% of the pairs. We further looked into the cases where Full-ML hurts performance and found that most of them were due to mistakes in entity linking or preprocessing, which reduced the features reliability. Other failure cases are due to the fact that the model put too much emphasis on specific features, such as features indicating the existence of relation terms in the sentence. However, these failure cases suggest that this is not desirable in all the cases. A more comprehensive analysis of failure cases is given in Section 5.2.2.

Reporting on metrics based on perfect or excellent labelled sentences for pairs that do not have at least one candidate sentence labelled as perfect or excellent is not fair. Metrics that explicitly account for perfect or excellent sentences can never have a score of 1 on

FIGURE 5.1: NDCG@1 difference per entity pair between B5 and Full-ML. A positive difference marks improvement for RF over B5.

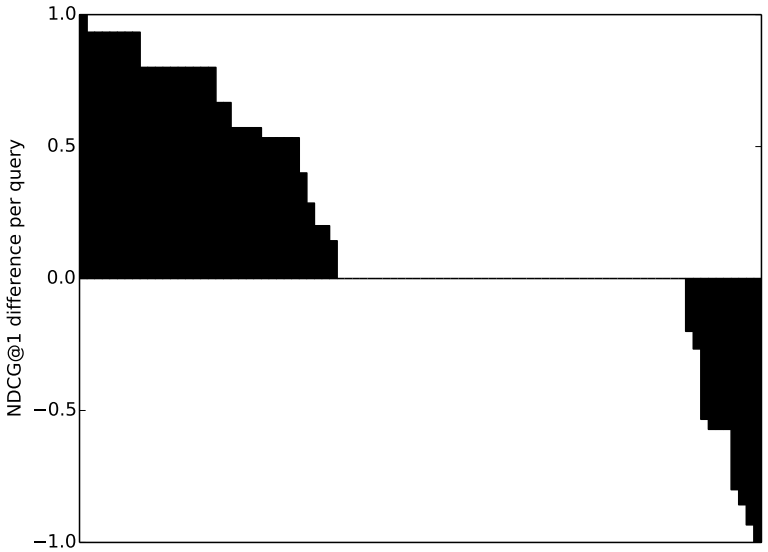


TABLE 5.3: Results for the full machine learning model, where we filtered out entity pairs that do not have any sentence annotated as perfect (HasPerfect) or any sentence annotated as excellent (HasExcellent).

Dataset	NDCG@1	NDCG@10	Perfect@1	Excellent@1	ERR@1	ERR@10	PerfectRR	ExcellentRR
HasPerfect	0.7682	0.8772	0.6923	0.7692	0.7087	0.8129	0.8054	0.8577
HasExcellent	0.7617	0.8768	-	0.7500	0.6516	0.7596	-	0.8525

the full dataset. For that reason, we exclude entity pairs that do not have any sentence annotated as perfect or excellent from the averaged results of the same experiment for the full machine learning model and show the results in table 5.3. We observe that for entity pairs that have at least one sentence annotated as perfect, we get one sentence annotated as perfect or excellent for 75% of the cases at the top of the ranking. Also, a sentence annotated as perfect is ranked at the top of the ranking for almost 70% of the cases. Note that, as we described in Section 4.2, even though a sentence annotated as excellent is not as good as a sentence annotated as perfect, it can explain the relationship of an entity pair at a satisfactory level. In addition, the mean reciprocal rank scores for both perfect and excellent are promising.

5.2.2 Insights & error analysis

In this section we gather results insights and give examples of failure cases of the full machine learning model. Table 5.4 contains some examples of entity pairs for which the model failed to return the best possible sentence on the top of the ranking.

TABLE 5.4: Examples of failure cases. Each group in the table contains an entity pair, the relation of interest and two sentences. The first sentence is the best candidate sentence for the entity pair and the second is the sentence ranked highest by our model.

(#1) Dina Lohan - Michael Lohan (Person_IsSpouseOf_Person)
Michael Lohan (born April 25, 1960) is the ex-husband of Dina Lohan, and the father of actresses Lindsay Lohan and Ali Lohan.
Dina Lohan also is mother of Michael Lohan Junior and Dakota Lohan.
(#2) Heather Langenkamp - Robert Englund (MovieActor_CoCastsWith_MovieActor)
John Saxon and Robert Englund also returned with Langenkamp in "Dream Warriors" and "New Nightmare".
The film introduced the iconic villain Freddy Krueger, played by Robert Englund.
(#3) Letizia Ramolino - Napoleon (Person_IsParentOf_Person)
Nobile Maria Letizia Buonaparte née Ramolino (Marie-Lætitia Ramolino, Madame Mère de l'Empereur) (24 August 1750 – 2 February 1836) was the mother of Napoleon I of France.
Letizia Ramolino bore 13 children, eight of whom survived infancy, and most of whom were created monarchs by Napoleon.
(#4) Karl von Habsburg - Otto von Habsburg (Person_IsChildOf_Person)
Born in Starnberg, Bavaria, Germany, Karl von Habsburg is the son of Otto von Habsburg and Princess Regina of Saxe-Meiningen, and the grandson of the last Austrian emperor, Charles I.
In 1 January 2007, his father, Otto von Habsburg, relinquished his position as the head of the House of Habsburg, a status which then devolved on Karl.
(#5) Ossie Davis - Ruby Dee (Person_IsSpouseOf_Person)
Ruby Dee (born October 27, 1922) is an American actress, poet, playwright, screenwriter, journalist, activist, and widow of actor Ossie Davis.
Three years later Ruby Dee married actor Ossie Davis.

A general observation regardless of the ranking errors is that many sentences contain information that might be regarded unnecessary for explaining the relation of interest between the two entities. For example, the best candidate sentence of the pair "Letizia Ramolino" - "Napoleon" (#3) could be reduced to *"Nobile Maria Letizia Buonaparte née Ramolino was the mother of Napoleon I of France."*, without loss in relation explanation quality. This challenge might be addressed by adding a sentence compression preprocessing step [25, 66], possibly modified for our task. This might also make the features more stable across sentences. In order to further analyse this phenomenon, we asked one annotator to judge each sentence based on whether it would retain its relation explanation quality or even be benefited if one or more parts of the sentence were hypothetically removed. The annotator annotated the full sentence dataset and judged 62% of the sentences as valid candidates for compression. We then asked a second annotator

to do the same for one third of the sentences in the dataset. Their judgements were almost identical, confirming that this phenomenon needs further investigation. However, deciding whether modifying the sentences for this task in the context of search engines is needed is not trivial. For example, consider the best candidate sentence of the pair "Dina Lohan" - "Michael Lohan" (#1). One might argue that if we reduce that sentence to *"Michael Lohan (born April 25, 1960) is the ex-husband of Dina Lohan."*, even though there would be no loss in relation explanation quality, there would be loss in context. The decision of whether to perform sentence compression or not might also depend on the device of the search engine user, as a more succinct version is more reasonable to be preferred for a mobile device. A user study could shed light on this problem and suggest solutions on how to address it. We leave this for future work.

It is interesting that the best sentence of the pair "Dina Lohan" - "Michael Lohan" (#1) was not ranked at the top position by our model, as it contains both the entities and the term "husband". This term is one of the top ranked phrases in $word2vec(Person\_IsSpouseOf\_Person)$. One possible cause of this might be that the entity linking module linked the sentence with the highest model score to "Michael Lohan" instead of "Michael Lohan Junior" (his son), as the latter entity was not included in the Wikipedia version we used. Another possible reason might be that the model had a preference on shorter sentences or sentences in which the two entities were positioned close to each other.

Consider the best candidate sentence of the pair "Heather Langenkamp" - "Robert Englund" (#2), who both appeared in the films "Dream Warriors" and "New Nightmare". Although this sentence refers to both the entities, it might be the case that it is not ranked first by our model because it does not include any terms directly related to the relation of interest, in contrast to the top ranked sentence, which contains the terms "film" and "played". The last two terms were included in $word2vec(MovieActor\_CoCastsWith\_MovieActor)$. A related example is the pair "Karl von Habsburg" - "Otto von Habsburg" (#4). The top ranked sentence of this pair contains the term "father" which is similar to the relation "IsChildOf", but that particular sentence only discusses a minor aspect of the relation of the two entities. These two examples show that matching the relation terms or terms similar to them may cause problems in some cases.

Another example is the model's top ranked sentence for the pair "Letizia Ramolino" - "Napoleon" (#3). Even though that sentence describes the relation indirectly, it could be presented together with the best candidate sentence to the user as it gives more information about the entity pair. A summarization system specifically designed for this task may eventually combine the top ranked sentences of our model in order to produce a succinct paragraph explaining the relation.

Finally, consider the pair "Ossie Davis" - "Ruby Dee" (#5). The sentence ranked highest by the model explains the relation at a satisfactory level, however the best candidate sentence is self contained and gives a more accurate explanation, as it mentions that "Ruby Dee" is the widow of "Ossie Davis". Our model was not able to capture this difference as none of the terms of or similar to the relation contained the term "widow". This suggests further investigation of how relations should be represented. Another interesting observation about this pair is that the sentence ranked highest by our model has a temporal nature which in this case makes the sentence not self-contained. In fact, the previous sentence of this sentence's document refers to the divorce of Ruby Dee with Frankie Dee Brown. Exploring solutions that identify these cases is worth investigating in the future.

We note that some of these errors might be produced because some features work well in general but dominate the decisions made by the model. This results in errors for cases where the intuitions behind these features do not hold. We further analyse the effect of each feature on retrieval performance in Section 5.3.

5.2.3 Parameter settings

Here we examine the effect of parameter settings on our main machine learning algorithm in order to answer research question (RQ2). RF has two main parameters to tune, the number of iterations (one tree is constructed at each iteration) and the features sampling rate. Features sampling rate is the percentage of features used by the algorithm to find the best split at each node in the tree. If not varied, the number of iterations is set to 300 and the feature sampling rate is set to 0.3.

Figure 5.2 shows how the performance varies for different number of iterations. The plot shows that after 300 iterations the performance becomes stable for both NDCG@1 and NDCG@10, suggesting that we can stop the training procedure at that point without decrease in performance.

The effect of varying the features sampling rate is shown in figure 5.3. It can be observed that we can safely set this parameter between 0.1 and 0.3, as there are no significant differences between 0.3 and 0.1 or 0.2. These two experiments confirm that RF is relatively insensitive to parameter settings.

We also examine how different ways of normalizing the features affect the performance of our method. Table 5.5 shows how the performance varies if : (i) no normalization is used (No-norm), (ii) each feature is normalized by the sum of its values (Sum), (iii) each

FIGURE 5.2: Performance for different number of iterations for RF. Boldface marks best performance in the respective metric.

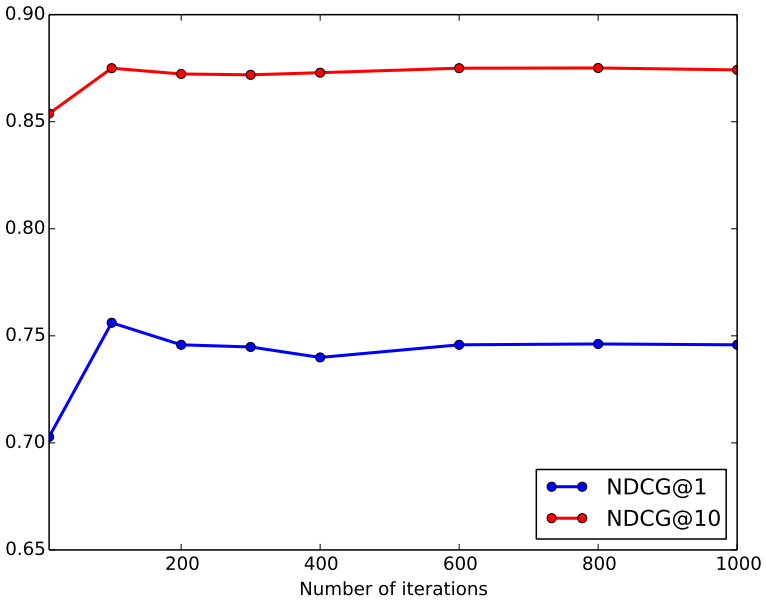
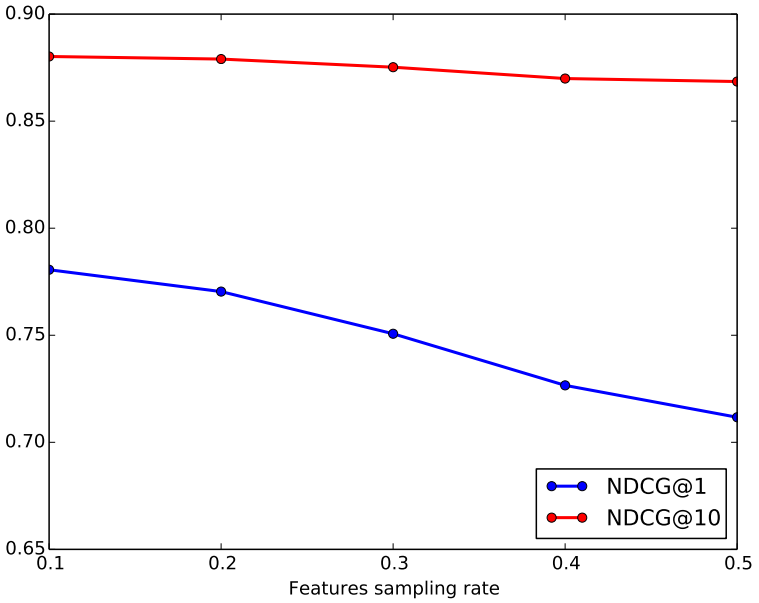


FIGURE 5.3: Performance for different values of feature sampling rate.



feature is normalized using its mean and standard deviation (Z-score), or (iv) each feature is normalized by its min/max values (Linear). The results confirm that our choice of selecting No-norm as our default normalization method is safe, as it is one of the best performing methods. No-norm significantly outperforms Sum and marginally outperforms Linear normalization in all metrics. However, there is no significant difference

TABLE 5.5: Results for different feature normalization methods. Boldface marks best performance in the respective metric. Significance is tested against No-norm.

Normalization	NDCG@1	NDCG@10	Perfect@1	Excellent@1	ERR@1	ERR@10	PerfectRR	ExcellentRR
No-norm	0.7448	0.8719	**0.5222**	0.6667	**0.5931**	0.7024	**0.6039**	0.7590
Z-score	**0.7514**	**0.8813**	0.5000	**0.6778**	0.5854	**0.7032**	0.5939	**0.7720**
Linear	0.7069	0.8625	0.4444	0.6111	0.5389	0.6808	0.5688	0.7367
Sum	0.6107▽	0.8285▽	0.3556▽	0.5222▽	0.4618▽	0.6327▽	0.5119▽	0.6841▽

between No-norm and Z-score in any metric.

5.3 Feature analysis

In this section we try to answer research question (RQ3), which examines which features among the ones we devised are the most important for our task. Here we group features per type and per unit. By feature units we refer to single features, e.g. $Length(s)$, or other features that are actually sets of features, such as the Part-of-Speech distribution of the sentence, which consists of 4 features (percentages of verbs, nouns, adjectives and others).

We analyse the impact of different feature combinations on retrieval performance. The metric we optimize for is NDCG@1 but the effect in performance is stable across metrics. Nevertheless, we also report on NDCG@10 for completeness. Furthermore, we provide a high-level overview of the cost of calculating the features.

5.3.1 Per feature type performance

In this experiment, we group the features per type and examine the effect of each in retrieval performance.

Table 5.6 shows the results for different feature types, combined using RF. It is clear from this table that when feature types are tested in isolation, source features can perform remarkably well. The source features type is the only feature type that does not perform significantly worse than the full feature set. This is because source features encode information about the documents from which the sentences were extracted from, in our case Wikipedia articles. Therefore, sentences extracted from the Wikipedia article of either the entities in the entity pair appear more likely to be relevant. The fact that entity features form the second best group confirms that introducing features based on entities identified in the sentences is beneficial for this task. Note that no individual feature

TABLE 5.6: Performance per feature type. The features were combined using RF. Significance is tested against "all features".

Features	NDCG@1	NDCG@10
All features	0.7448	0.8719
Source	0.7087	0.8397
Entity	0.6418▽	0.8151▽
Text	0.5762▼	0.7890▼
Relation	0.5521▼	0.7969▼

TABLE 5.7: Performance when removing one feature type at a time from the full feature set. The features are combined using RF. Significance is tested against "all features".

Features	NDCG@1	NDCG@10
All features	0.7448	0.8719
All without entity	0.7465	0.8719
All without relation	0.7353	0.8635
All without text	0.7204	0.8662
All without source	0.7060	0.8515

type performs better than the full feature set, a result that confirms the importance of combining different feature types.

Table 5.7 shows how performance varies when removing one feature type at a time from the full feature set. The results are not significantly different against the model that uses the full feature set for any of these settings, thus no safe conclusions can be derived for the impact of every feature type. However, it appears that source features are the ones with the highest impact. Also, even though the text features did not perform well in isolation, it appears that they are beneficial when combined with the other features. The results suggest that RF can effectively select the important features and arrive to reasonable performance. An interesting finding of this experiment is that the model can achieve competitive performance even without the source or the relation dependent features.

5.3.2 Per feature unit performance

In this experiment we examine the effect of different feature units on retrieval performance.

We first report on individual feature units performance in table 5.8. Note that for some entity pairs the feature values were the same for all sentences, therefore RF was not able to produce even a random prediction. Therefore, in this experiment. we combine the features using CoordAscent with ties broken arbitrarily. In this table we observe similar trends to the ones observed in table 5.6 where we combined features inside each feature type. One important difference is that the best individual relation feature unit, $ScoreLCLabels(e_a, e_b, r, s)$, achieves better performance than when combining all relation features using machine learning. Another interesting result is that among the relation features, the ones that take into account phrases in $word2vec(r)$ achieve the best performance. This suggests that using *word2vec* vectors for selecting the phrases to represent the relations is beneficial for this task. The fact that all individual source features, $DocCount(e, d(s))$, $SentenceSource(e, d(s))$ and $Position(s, d(s))$ are among the best performing features confirms the importance of source features, highlighted in the previous section. The results also highlight the importance of entity features. $ContainsBothLink(e, s)$, which indicates the presence of both the entities of the entity pair in the sentence and $NumCommLinks(e_a, e_b, s)$, which indicates whether a sentence contains common links of the entity pair, are the best performing entity feature units. In addition, although $POS(s)$ is one of the worst performing features, $BetweenPOS(e_a, e_b, s)$ performs reasonably well, a result confirming the importance of involving entities in sentence representation. The feature accounting for the length of the sentence achieves the best performance among the text features. Note that no individual feature unit is able to significantly outperform any of the baselines, a result confirming the need of using machine learning to achieve reasonable performance.

While we can get an indication of features importance by examining them in isolation, we also examine which features are considered important by the learning algorithm.To this end, we utilize a greedy feature selection procedure. This procedure starts with a baseline and greedily adds the feature with the biggest gain in performance until no other feature can be added that can improve the performance. Note that as this procedure is greedy, it may skip features that are very correlated and therefore provide similar gain in performance with other features. For this reason, if a feature is skipped during the procedure, this does not necessarily mean that the feature is not important. In order to obtain as accurate results as possible regarding feature importance, we take the average over several runs when adding each feature.

Our first greedy feature selection experiment starts with a baseline that combines the B5 features. B5 features use titles, labels or redirects for representing the entities and Lucene and BM25 as the ranking functions. In this experiment only the $SentenceSource(e, d(s))$, $Position(s, d(s))$ and $LeftPOS(e_a, e_b, s)$ were added before the procedure stopped. This procedure achieved a surprisingly high score

TABLE 5.8: Performance per feature unit. Units that consist of a set of features are combined using CoordAscent. Boldface marks best performing feature in the respective metric among the features of the same feature type.

Feature	NDCG@1	NDCG@10
Text features		
$AverageIDF(s)$	0.3922	0.7040
$SumIDF(s)$	0.4414	0.7324
$Length(s)$	**0.4632**	**0.7397**
$Density(s)$	0.4360	0.7056
$POS(s)$	0.3749	0.6807
Entity features		
$NumEntities(s)$	0.3780	0.6828
$ContainsLink(e, s)$	**0.5278**	**0.7645**
$ContainBothLinks(e, s)$	0.4416	0.7099
$AverageInLinks(s)$	0.4286	0.7236
$SumInLinks(s)$	0.4569	0.7327
$Spread(e_a, e_b, s)$	0.4824	0.7171
$BetweenPOS(e_a, e_b, s)$	0.4358	0.7245
$LeftPOS(e_a, e_b, s)$	0.4545	0.7384
$RightPOS(e_a, e_b, s)$	0.3111	0.6524
$NumEntitiesLeft(e_a, e_b, s)$	0.3414	0.6868
$NumEntitiesRight(e_a, e_b, s)$	0.3411	0.6828
$NumEntitiesBetween(e_a, e_b, s)$	0.3328	0.6907
$ContainsCommLinks(e_a, e_b, s)$	0.4407	0.7397
$NumCommLinks(e_a, e_b, s)$	0.4947	0.7543
Relation features		
$MatchTerm(r, s)$	0.3666	0.6963
$MatchSyn(r, s)$	0.3404	0.6928
$Word2vecScore(r, s)$	0.4824	0.7497
$MaxWord2vecScore(r, s)$	0.5247	0.7627
$MatchTermOrSyn(r, s)$	0.3515	0.6936
$MatchTermOrWord2vec(r, s)$	0.4823	0.7618
$MatchTermOrSynOrWord2vec(r, s)$	0.4743	0.7554
$ScoreLCLabels(e_a, e_b, r, s)$	**0.5587**	**0.7842**
$ScoreBM25Labels(e_a, e_b, r, s)$	0.5447	0.7787
Source features (Wikipedia)		
$Position(s, d(s))$	0.4881	0.7425
$SentenceSource(e, d(s))$	0.5335	0.7751
$DocCount(e, d(s))$	**0.5656**	**0.7889**

($NDCG@1 = 0.7484$), confirming the importance of source features in combination with relation-informed retrieval models for this task.

We might want our methods to be independent of the source of the sentences, thereby making them more generic and hopefully applicable to other domains. In order to examine the effect of the features that not aware of the source of the sentences, we repeat the previous experiment with the same baseline but we remove the source features from the feature candidates. Table 5.9 depicts the results of this experiment. We

TABLE 5.9: Greedy feature selection per feature unit, excluding source features. The features were combined using RF and averaged over 3 runs.

Features	NDCG@1	NDCG@10
B5 (LC+BM25)	0.5402	0.7837
$+Length(s)$	0.5600	0.7917
$+ContainsLink(e, s)$	0.5983	0.8141
$+MaxWord2vecScore(r, s)$	0.6616	0.8361
$+LeftPOS(e_a, e_b, s)$	0.6897	0.8457
$+BetweenPOS(e_a, e_b, s)$	0.6925	0.8506
$+Spread(e_a, e_b, s)$	0.7418	0.8684
$+NumCommLinks(e_a, e_b, s)$	0.7481	0.8664
$+MatchTermOrSynOrWord2vec(r, s)$	0.7564	0.8660

observe that the procedure selects at least one feature unit from each of the entity, text and relation feature types. The features selected during the greedy procedure confirm that modeling this task with entities is beneficial, as five entity-aware features were selected. These are $Spread(e_a, e_b, s)$, which measures the spread of the two entities in the sentence and gives the second larger gain in performance, $LeftPOS(e_a, e_b, s)$, $BetweenPOS(e_a, e_b, s)$, $ContainsLink(e, s)$ and $NumCommLinks(e_a, e_b, s)$. The fact that the greedy procedure selected two features that are aware of the relation and its similar terms, $MaxWord2vecScore(r, s)$ and $MatchTermOrSynOrWord2vec(r, s)$, confirms the importance of matching phrases similar to the relation, obtained using *word2vec* vectors and Wordnet. $Length(s)$ was the only purely text feature selected in the procedure.

We repeat the greedy feature selection procedure without using the source features in order to make sure that the greedy selection is not heavily dependent on the learning algorithm or the baseline. We combine the features using CoordAscent and use B5-LC-Labels as the baseline. The results are shown in table 5.10. Interestingly, the first four features selected, $MatchTermOrSynOrWord2vec(r, s)$, $ContainsLink(e, s)$, $Spread(e_a, e_b, s)$ and $NumCommLinks(e_a, e_b, s)$ were also selected in our previous greedy selection experiment, confirming their importance for this task. In addition, although $POS(s)$, $AverageIDF(s)$ and $NumEntitiesRight(e_a, e_b, s)$ were among the worst individual feature units when tested in isolation (see table 5.8), they can improve performance when combined using machine learning.

Note that we are aware of the existence of other feature importance analysis tools, such as mutual information, information gain [82] or correlation feature selection [53], but we leave the exploration of these for future work.

TABLE 5.10: Greedy feature selection per feature unit, excluding source features. The features were combined using CoordAscent and averaged over 5 runs.

Features	NDCG@1	NDCG@10
B5-LC-Labels	0.5587	0.7842
$+MatchTermOrSynOrWord2vec(r, s)$	0.6069	0.8078
$+ContainsLink(e, s)$	0.6642	0.8271
$+Spread(e_a, e_b, s)$	0.6723	0.8279
$+NumCommLinks(e_a, e_b, s)$	0.7040	0.8503
$+AverageIDF(s)$	0.7085	0.8469
$+POS(s)$	0.7225	0.8495
$+NumEntitiesRight(e_a, e_b, s)$	0.7332	0.8504

5.3.3 Features calculation cost

Regarding the cost of computing the features for each sentence, we can split the features in two categories. The first category contains features that are only dependent on the sentence itself, which include all the features in text and source features types and also $NumEntities(s)$, $AverageInLinks(s)$ and $SumInLinks(s)$. The second category contains features that are dependent on the entity pair and/or the relation of interest, which include all the rest entity features and all the relation features. The features of the first category can be pre-computed at indexing time and only require collection term statistics which are trivial to compute and a trained POS tagger, for which very efficient implementations exist.

In a dynamic scenario, where the entity pair and the relation of interest are not known beforehand, the features of the second category should be computed at run time. The entity features require entity matching, POS distribution calculation for parts of the sentence and matching of common entities. For performing entity matching only the positions of the entities are needed. These are stored in the index during the entity linking step, which happens at indexing time. For calculating the POS distribution for parts of the sentence we only need the pre-computed POS tags of the whole sentence and the positions of the entities in the sentence. The process of obtaining the common entities of an entity pair needs the calculation of similarities between Wikipedia pages. This must happen only once per entity pair, making the cost of the similarity calculations negligible. The relation features include matching of relation terms and also matching of phrases similar to the relation. These phrases need to be obtained from Wordnet synonyms and from operations on *word2vec* phrase vectors, as described in Section 3.2.2.3. For this we require a Wordnet index which can easily be plugged in our system and operations on

TABLE 5.11: Results for different machine learning algorithms. Boldface marks best performance in the respective metric. Significance is tested against RF.

Algorithm	NDCG@1	NDCG@10	Perfect@1	Excellent@1	ERR@1	ERR@10	PerfectRR	ExcellentRR
RF	**0.7448**	**0.8719**	**0.5222**	**0.6667**	**0.5931**	**0.7024**	**0.6039**	**0.759**
GBRT	0.6806	0.8426	0.4444	0.6111	0.5319	0.6652	0.5604	0.7250
CoordAscent	0.6684	0.8549	0.4444	0.5889	0.5306	0.6704	0.5671	0.7164
RankBoost	0.6481▽	0.8425▽	0.4111▽	0.5444▽	0.5021▽	0.6491▽	0.5377▽	0.6830▽
LambdaMART	0.6361▼	0.8262▼	0.4333▽	0.5667▼	0.5049▼	0.6471▼	0.5543▼	0.6950▼
AdaRank	0.4077▼	0.7104▼	0.1889▼	0.3222▼	0.2965▼	0.5104▼	0.3910▼	0.5361▼
ListNet	0.3498▼	0.6899▼	0.1667▼	0.2556▼	0.2597▼	0.4862▼	0.3730▼	0.4971▼
RankNet	0.3460▼	0.6846▼	0.1667▼	0.2444▼	0.2569▼	0.4790▼	0.3665▼	0.4852▼

word2vec vectors, which need to be trained on a large text collection. The training of the *word2vec* phrases vectors can be done offline. In the case where we are only interested in pre-defined relations, the phrases similar to each relation can be pre-computed. In addition, in order to calculate relation features which are based on IR retrieval models, we can use existing efficient implementations of these models.

Note that in a real-world scenario a search engine uses a knowledge base of pre-defined entities and relations. In this scenario, all the previous computations can happen offline and thus the search engine can provide relationship explanations without extra cost.

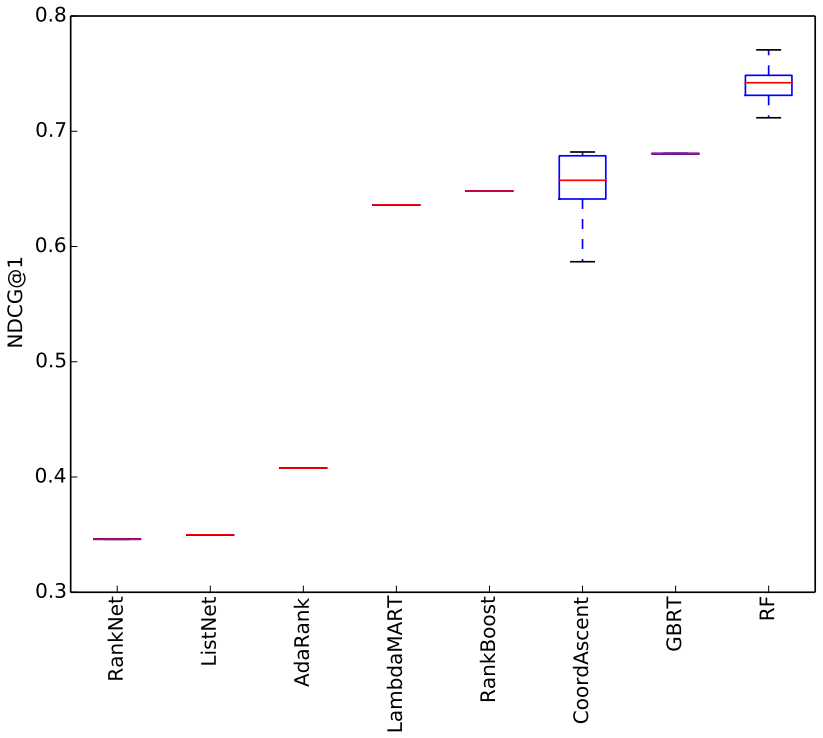
5.4 Machine learning algorithms

In this section we address the first part of research question (RQ4) which regards the effect of different machine learning (learning to rank) algorithms on the performance.

Table 5.11 shows the results for the machine learning algorithms we examine. Note that AdaRank, LambdaMART and CoordAscent are optimized for NDCG@1, whereas the rest algorithms have internal optimization criteria. Also note that we use the default RankLib parameters for all the algorithms, thus no tuning is performed. We observe that RF outperforms all the rest algorithms for all metrics. The improvements are significant for RankNet, ListNet, AdaRank, RankBoost and LambdaMART for every metric but not significant for GBRT and CoordAscent for any metric. This result suggests that even though RF does not significantly outperform every other algorithm, it can be safely selected as the learning algorithm for this task, as it can effectively select and combine the most important features.

The results of CoordAscent and RF are not the same across different runs because of the nature of the algorithms. Figure 5.4 shows how NDCG@1 varies over 10 runs for

FIGURE 5.4: Box plot depicting NDCG@1 scores for the machine learning algorithms we consider. RF and CoordAscent were ran for 10 times, while the rest were ran once. Each box shows the median score (red line) and the upper and lower quartiles (up and bottom blue lines). The maximum and lower scores are shown outside each box with black horizontal lines, connected with the box with vertical lines (whiskers) which indicate score variability.



all the algorithms. Performance for RF is relatively stable, whereas we observe much higher variance for CoordAscent.

It is interesting to note that the two best performing algorithms, RF and GBRT, are both pointwise learning to rank algorithms. Pointwise algorithms outperformed pairwise and listwise algorithms for a similar task, question answering [1].

5.5 Comparison to a competitive system

In this section we try to answer research question (RQ5), which examines how our method works compared to a competing system on a separate dataset that contains popular entities. We split this experiment in two parts. In the first part we evaluate how our method performs on this dataset and in the second part we perform a comparison between our method and a competitive system. Below we describe the dataset construction and present the results of this experiment.

5.5.1 Dataset

5.5.1.1 Entity pairs

In order to construct an entity pairs dataset that contains popular entities, we first sample entities from the top 2000 popular entities among search engine users using the entities' popularity distribution described in Section 4.1.1. We then sample entity pairs that contain at least one of these popular entities, excluding entity pairs that are also included in the dataset described in Section 4.1.1 or did not have a candidate sentence. This results in 156 entity pairs with a total of 1871 sentences and an average of 11.99 sentences per entity pair. The maximum number of sentences for an entity pair is 203 and the minimum is 1.

We evaluate performance in the case where only the top-ranked relationship explanation sentence is shown to the user. In order to obtain a single sentence for each entity pair, we rank the candidate sentences using a model trained on the dataset described in Section 4.1.1. This model was trained using the full feature set and the random forest algorithm. We use the best performing parameter settings of the previous experiments. For each entity pair, we select the sentence with the highest score.

In order to compare the relationship explanations provided by our method to a state-of-the-art competing system, we utilize Google's relationship explanations. We manually obtain these explanations using Google's search engine. Note that not every entity pair in our dataset appears in the recommended entities of Google's search engine and vice versa, as the entity recommendation algorithms are different. Out of the 156 entity pairs in our dataset, 86 of them appear in Google's search engine recommendations, while 81 of these contain an explanation for the entity pair.

5.5.1.2 Annotation

For the first part of this experiment, we asked one human annotator to annotate the relationship explanation sentences provided by our method for each pair with respect to a number of different dimensions. More specifically, each sentence was judged by whether the relation of interest was explained at a satisfactory level (directly or indirectly), whether the sentence was self-contained and whether the sentence is a candidate for compression. Sentences that explain a relation indirectly might be of the form "Actor A co-stars with B and C in movie X". In this case, the relation between entities B and C can be derived indirectly by inspecting the sentence. Sentences that are not self-contained might miss the names of some entities, e.g. "the film", or might be unclear

in terms of the time period they are referring to, e.g. "three years later". Examples of sentences that might be candidates for compression were presented in Section 5.2.2.

The second part of this experiment regards the comparison between our method and Google's relationship explanations. To this end, we asked one annotator to compare the explanations for the entity pairs that are provided by both systems (81 pairs). The explanations were judged in terms of explanation quality, i.e. which of the two is better for describing the particular relation of interest. They were also annotated in terms of detail level, i.e. which of the two gives more details about the entity pair.

5.5.2 Results and analysis

The results of the first part of the experiment which regards the performance of our method on the popular entity pairs dataset can be summarized as follows. In 80% of the cases the top retrieved sentence contained the relation of interest of the entity pair. Out of these cases, 67% explain the relation of interest directly, 72% are self-contained and 69% could be considered for compression. From these results we can derive that even though our automatic method fails in some cases, it performs sufficiently well.

We analyse our results by examining some specific examples. Table 5.12 shows some interesting examples of top-ranked sentences. We have already discussed some failure cases that appear here together with suggestions for improvements in Section 5.2.2. An example sentence that explains the relation directly and is self-contained is sentence (#1). Sentence (#2) is also self-contained but it explains the relation indirectly. Note that for some of these cases, there is no candidate sentence among the ones extracted that explains the relation in a more direct way. Sentences (#3) and (#4) are not self-contained. Sentence (#3) explains the relation directly but misses the name of an entity ("the film"). This can be addressed by using a more sophisticated co-reference resolution system. Sentence (#4) also explains the relation directly but it is not clear to which time period it is referring ("the following year"). Finally, sentence (#5) does not explain the relation of interest. In fact, we observed that this particular type of relation (Athlete_PlaysSameSportTeamAs_Athlete) was impossible to be explained by sentences from Wikipedia in most of the cases. Further exploration is needed in order to address these issues, but this is left for future work.

Since our model is trained using the random forest algorithm, it outputs a confidence score for each sentence. We examine whether we can use a threshold on this score in order to improve the precision of our method when applied in a real-world system. Figure 5.5 shows how the performance varies for different values of the threshold. We observe that applying a threshold can be beneficial for improving precision while keeping

FIGURE 5.5: Precision and recall for different values of the threshold. In this plot, a sentence is relevant if it explains the relation either directly or indirectly.

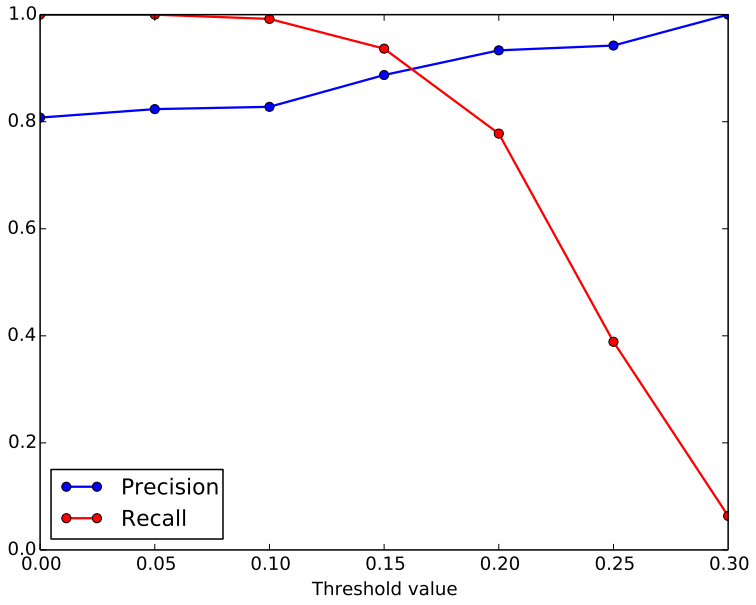


TABLE 5.12: Examples of top-ranked sentences. Each group in the table contains an entity pair, the relation of interest and the top ranked sentence.

(#1) Ben Affleck - Bruce Willis (MovieActor_CoCastsWith_MovieActor)
Affleck starred in "Armageddon" (1998) opposite Bruce Willis.
(#2) Hugh Jackman - Kate Winslet (MovieActor_CoCastsWith_MovieActor)
Katie Finneran's most recent film was "Movie 43" in which she played Angie and also appeared alongside Hugh Jackman and Kate Winslet.
(#3) Bryan Singer - Tom Cruise (MovieDirector_Directs_MovieActor)
The film stars Tom Cruise and is directed by Bryan Singer.
(#4) Cameron Diaz - Tom Cruise (MovieActor_CoCastsWith_MovieActor)
The following year Cruise starred in the romantic thriller "Vanilla Sky" (2001) with Cameron Diaz and Penélope Cruz.
(#5) Cristiano Ronaldo - Karim Benzema (Athlete_PlaysSameSportTeamAs_Athlete)
Karim Benzema was also shortlisted by the French magazine France Football for the 2008 Ballon d'Or award, won by Cristiano Ronaldo.

recall at a reasonable level. A "sweet spot" for the threshold on this dataset is 0.15, for which our method achieves a precision of 88% and a recall of 93%.

Table 5.13 shows the results of the second part of this experiment, in which we compare our method to Google's relationship explanations. In terms of explanation quality, our method performs at least as good as Google's system in 61% of the cases. Note that

TABLE 5.13: Side by side evaluation between our method (Y) and Google's relationship explanations (G). G ∼ Y indicates the number of cases for which the results were indistinguishable in terms of quality, G >Y indicates the number of cases for which Google's result was better than ours and >G indicates the number of cases for which Google's result was worse than ours.

Quality	G ∼ Y	50
	G >Y	29
	Y >G	2
Level of detail	G ∼ Y	33
	G >Y	27
	Y >G	21

we do not have any indication of how Google's system produces the explanations, but our guess is that they are produced by combining evidence from a knowledge base using sentence templates. These explanations might also involve human editing. Because of this, that system achieves almost perfect accuracy. On the other hand, our method only relies on a text corpus for extracting the sentences. This makes it more generic, as it is able to provide explanations for virtually every entity pair for which a good candidate sentence exists. We have seen in figure 5.5 that a threshold can be used in order to further improve the precision of our method.

As we can see in table 5.13, there is no clear winner in terms of explanation detail level. As our method extracts and ranks sentences from Wikipedia, it is able to produce explanations enriched with details about the relation of the entity pair. For example, our method gives the sentence "*Christopher Columbus married Filipa Moniz Perestrelo, daughter of the Porto Santo governor and Portuguese nobleman of Lombard origin Bartolomeu Perestrello.*" for the entity pair "Christopher Columbus" - "Filipa Moniz Perestrelo" (Person_IsSpouseOf_Person), whereas Google's system outputs the sentence "*Filipa Moniz Perestrelo was Christopher Columbus's spouse.*" for the same pair.

On the other hand, Google's system is able to give more details in cases where two entities share more than one relation. An example of this is entity pair "Ben Affleck" - "Jennifer Garner" (Person_IsSpouseOf_Person), for which our method outputs the sentence "*Ben Affleck has been married to Jennifer Garner since June 2005, and they have two daughters and a son.*", whereas Google's system outputs the sentences "*Ben Affleck and Jennifer Garner have been married since 2005. Both appear in Pearl Harbor and Daredevil.*". The second sentence of Google's system gives details about another

relation of the entity pair, (MovieActor_CoCastsWith_MovieActor). We plan to address this challenge by combining multiple sentences in future work.

Chapter 6

Conclusion and future work

This work presents a method for explaining pre-defined relations between knowledge base entities with sentences. Our method extracts candidate sentences that refer to each entity pair, identifies entities in them and ranks the candidate sentences by combining a rich set of features using state-of-the-art supervised machine learning algorithms. In a commercial search engine scenario where a knowledge base containing entities and pre-defined relations is used, both sentence extraction and sentence ranking can be done offline, thus providing the user with relation explanations with negligible cost.

We conducted several experiments in order to examine the effect of different features on retrieval performance. Features dependent on the source document of the sentences were found to be very important for this task. Also, we found that state-of-the-art phrase vector representations are very helpful for identifying phrases that refer to the relation of interest, especially when combined with state-of-the-art retrieval models. We found that simple features that account for the presence or the position of the entity pair in a sentence can be helpful. Furthermore, features which exploit similarity measures between the entities based on Wikipedia structure in order to identify the presence of other entities related to the entity pair can also be beneficial. Finally, we found that features only dependent on the sentence text can also provide improvements in performance.

Note that our features do not require expensive linguistic analysis tools for sentence pre-processing such as dependency parsing or semantic parsing (used for example for QA [92]). Our sentence pre-preprocessing step only requires a POS tagger. The features we devised mainly focus on how to represent the existence of entities and relations in sentences, most of them being easy to compute. This makes our method relatively efficient even in a dynamic scenario, where the relations are not pre-defined.

In addition, we benchmarked several learning to rank algorithms and found that random forest is the most appropriate algorithm for this task among the ones that we examined. The experiments concerning the effect of parameter settings on the performance of this algorithm showed that the algorithm is relatively insensitive to parameter settings. We further analysed the results by providing specific entity pair examples for which our method failed to provide the best possible sentence ranking, together with suggestions for improvements.

Furthermore, we have seen that our method can perform reasonably well on an entity pairs dataset that contains popular entities, producing a satisfactory explanation for 80% of the cases. On this dataset, our method is able to perform at least as good as a state-of-the-art competing system for 61% of the cases. In addition, our method gives at least as many details as the competing system for 66% of the cases.

There are multiple future research directions that stem from this work. First, we would like to evaluate our method on a larger dataset consisting of entity pairs and relations of any type. This would require to change the heuristic co-reference resolution step which is designed especially for people entities to a more sophisticated approach. We would also like to evaluate our method on other domains. A possible source of sentences is the content of the web pages that are returned from search engines when searching for an entity pair. For this we would have to exclude the Wikipedia-dependent features from our features set, which we showed that are important but not indispensable for this task. Another research direction is the investigation of the effect of having a different trained model for each relation type, which would require a larger training dataset. A similar idea was investigated in the context of QA [130].

Our model is able to achieve competitive performance, even without the relation dependent features. An idea for future work is to extract the most logical relation(s) between two entities based on the top-ranked sentences of a relation-independent version of our method. This might involve a top-level classifier that decides whether a relation exists between two entities. Another future direction might be to aggregate different learning to rank algorithms in a single rank using supervised aggregation methods, an idea successfully applied for the task of QA [1]. We have shown that our method might benefit from the inclusion of a sentence compression pre-processing step [25, 66]. An investigation of sentence compression techniques with a user study and of ways to modify these techniques in order to count for the characteristics of this task is in our plans.

Another possible research direction is the inclusion of a summarization module that automatically combines sentences that cover different aspects of the relation on the top of the ranking or even combines different relations of the entity pair. For example, two persons might be married and also co-appear in a movie. Some preliminary experiments

showed that a strong multi-sentence compression algorithm based on redundancy [42] performs poorly for this task, suggesting the need for further development. We would also like to explore whether sentence fusion techniques are appropriate for this task [43]. Furthermore, special attention should be given when combining temporal dependent sentences. A similar idea has been investigated by a commercial search engine but the methods used are not publicly available.[1]

At the time this work was written, the proposed method was considered to be further developed for production use by a major commercial search engine, Yahoo.

[1] http://blogs.bing.com/search/2014/02/21/timeline-understanding-important-events-in-peoples-lives/

Bibliography

[1] A. Agarwal, H. Raghavan, K. Subbian, P. Melville, R. D. Lawrence, D. C. Gondek, and J. Fan. Learning to rank for robust question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 833–842. ACM, 2012.

[2] G. Agarwal, G. Kabra, and K. C.-C. Chang. Towards rich query interpretation: walking back and forth for mining query templates. In *Proceedings of the 19th international conference on World wide web*, pages 1–10. ACM, 2010.

[3] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM, 2000.

[4] E. Agichtein, S. Lawrence, and L. Gravano. Learning to find answers to questions on the web. *ACM Transactions on Internet Technology (TOIT)*, 4(2):129–162, 2004.

[5] J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 314–321. ACM, 2003.

[6] P. André, J. Teevan, S. T. Dumais, et al. Discovery is never by chance: designing for (un) serendipity. In *Proceedings of the seventh ACM conference on Creativity and cognition*, pages 305–314. ACM, 2009.

[7] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[8] N. Balasubramanian and S. Cucerzan. Topic pages: An alternative to the ten blue links. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 353–360. IEEE, 2010.

[9] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[10] M. Bansal, K. Gimpel, and K. Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014.

[11] R. Blanco and H. Zaragoza. Finding support sentences for entities. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346. ACM, 2010.

[12] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec. Entity recommendations in web search. In *The Semantic Web–ISWC 2013*, pages 33–48. Springer, 2013.

[13] I. Bordino, G. De Francisci Morales, I. Weber, and F. Bonchi. From machu_picchu to rafting the urubamba river: anticipating information needs via the entity-query graph. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 275–284. ACM, 2013.

[14] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[15] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[16] S. Brin. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183. Springer, 1999.

[17] M. Bron, K. Balog, and M. De Rijke. Ranking related entities: components and analyses. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1079–1088. ACM, 2010.

[18] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[19] C. J. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. In *Yahoo! Learning to Rank Challenge*, pages 25–35, 2011.

[20] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[21] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[22] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *Yahoo! Learning to Rank Challenge*, pages 1–24, 2011.

[23] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630. ACM, 2009.

[24] J. C. K. Cheung and X. Li. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 383–392. ACM, 2012.

[25] J. Clarke and M. Lapata. Global inference for sentence compression: An integer linear programming approach. *J. Artif. Intell. Res.(JAIR)*, 31:399–429, 2008.

[26] K. Collins-Thompson, P. Ogilvie, Y. Zhang, and J. Callan. Information filtering, novelty detection, and named-page finding. In *TREC*, 2002.

[27] D. Cossock and T. Zhang. Subset ranking using regression. In *Learning theory*, pages 605–619. Springer, 2006.

[28] K. Crammer, Y. Singer, et al. Pranking with ranking. In *NIPS*, volume 14, pages 641–647, 2001.

[29] N. Craswell and M. Szummer. Random walks on the click graph. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246. ACM, 2007.

[30] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 7, pages 708–716. Citeseer, 2007.

[31] N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12. ACM, 2009.

[32] A. P. De Vries, A.-M. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. Overview of the inex 2007 entity ranking track. In *Focused Access to XML Documents*, pages 245–251. Springer, 2008.

[33] G. Demartini, T. Iofciu, and A. P. De Vries. Overview of the inex 2009 entity ranking track. In *Focused Retrieval and Evaluation*, pages 254–264. Springer, 2010.

[34] A. Doko, M. Štula, and D. Stipaničev. A recursive tf–isf based sentence retrieval method with local context. *International Journal of Machine Learning and Computing*, 3(2):195–200, 2013.

[35] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.(JAIR)*, 22(1):457–479, 2004.

[36] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[37] L. Fang, A. D. Sarma, C. Yu, and P. Bohannon. Rex: explaining relationships between entity pairs. *Proceedings of the VLDB Endowment*, 5(3):241–252, 2011.

[38] C. Fellbaum. *WordNet*. Wiley Online Library, 1998.

[39] R. T. Fernández, D. E. Losada, and L. A. Azzopardi. Extending the language modeling framework for sentence retrieval to include local context. *Information Retrieval*, 14(4):355–389, 2011.

[40] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[41] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[42] K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics, 2010.

[43] K. Filippova and M. Strube. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185. Association for Computational Linguistics, 2008.

[44] S. Fisher and B. Roark. Feature expansion for query-focused supervised sentence ranking. In *Document Understanding (DUC 2007) Workshop Papers and Agenda*, 2007.

[45] J. L. Fleiss, B. Levin, and M. C. Paik. *Statistical methods for rates and proportions*. John Wiley & Sons, 2013.

[46] A. Foster and N. Ford. Serendipity and information seeking: an empirical study. *Journal of Documentation*, 59(3):321–340, 2003.

[47] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[48] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[49] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[50] 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.

[51] Z. GuoDong, S. Jian, Z. Jie, and Z. Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics, 2005.

[52] B. Hachey, W. Radford, and J. R. Curran. Graph-based named entity linking with wikipedia. In *Web Information System Engineering–WISE 2011*, pages 213–226. Springer, 2011.

[53] M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[54] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM, 2011.

[55] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[56] L. Hirschman and R. Gaizauskas. Natural language question answering: the view from here. *Natural Language Engineering*, 7(04):275–300, 2001.

[57] N. Houlsby and M. Ciaramita. A scalable gibbs sampler for probabilistic entity linking. In *Advances in Information Retrieval*, pages 335–346. Springer, 2014.

[58] J. Hu, G. Wang, F. Lochovsky, J.-t. Sun, and Z. Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 471–480. ACM, 2009.

[59] M. Hu, A. Sun, and E.-P. Lim. Comments-oriented blog summarization by sentence extraction. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 901–904. ACM, 2007.

[60] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[61] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396. ACM, 2006.

[62] D. Jurafsky and J. H. Martin. *Speech & language processing*. Pearson Education India, 2000.

[63] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 31–39, 2014.

[64] M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *ACM SIGIR Forum*, volume 31, pages 178–185. ACM, 1997.

[65] M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, 2001.

[66] K. Knight and D. Marcu. Statistics-based summarization-step one: Sentence compression. In *AAAI/IAAI*, pages 703–710, 2000.

[67] R. Kop. The unexpected connection: Serendipity and human mediation in networked learning. *Educational Technology & Society*, 15(2):2–11, 2012.

[68] L.-W. Ku, L.-Y. Lee, T.-H. Wu, and H.-H. Chen. Major topic detection and its application to opinion summarization. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 627–628. ACM, 2005.

[69] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM, 2009.

[70] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, et al. Siteq: Engineering high performance qa system using lexico-semantic pattern matching and shallow nlp. In *TREC*, 2001.

[71] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language*

Learning: Shared Task, pages 28–34. Association for Computational Linguistics, 2011.

[72] W. Lehnert, C. Cardie, D. Fisher, E. Riloff, and R. Williams. University of massachusetts: Description of the circus system as used for muc-3. In *Proceedings of the 3rd conference on Message understanding*, pages 223–233. Association for Computational Linguistics, 1991.

[73] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[74] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346. ACM, 2008.

[75] D. Lin and P. Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04):343–360, 2001.

[76] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *Proceedings of the 21st international conference on World Wide Web*, pages 589–598. ACM, 2012.

[77] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[78] V. Lopez, V. Uren, M. Sabou, and E. Motta. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2(2):125–155, 2011.

[79] D. Losada. A study of statistical query expansion strategies for sentence retrieval. In *Proceedings of the SIGIR 2008 Workshop on Focused Retrieval*, pages 37–44, 2008.

[80] D. E. Losada and R. T. Fernández. Highly frequent terms and sentence retrieval. In *String Processing and Information Retrieval*, pages 217–228. Springer, 2007.

[81] H. Ma, C. Liu, I. King, and M. R. Lyu. Probabilistic factor models for web site recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 265–274. ACM, 2011.

[82] D. J. MacKay. *Information theory, inference, and learning algorithms*, volume 7. Citeseer, 2003.

[83] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79, 2001.

[84] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. De Rijke. *Learning semantic query suggestions*. Springer, 2009.

[85] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 563–572. ACM, 2012.

[86] D. Metzler and W. B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007.

[87] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM, 2007.

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

[89] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[90] 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.

[91] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[92] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[93] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. In *Yahoo! Learning to Rank Challenge*, pages 77–89, 2011.

[94] V. Murdock and W. B. Croft. A translation model for sentence retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 684–691. Association for Computational Linguistics, 2005.

[95] V. G. Murdock. *Aspects of sentence retrieval*. PhD thesis, University of Massachusetts Amherst, 2006.

[96] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM, 2004.

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

[98] J. Otterbacher, G. Erkan, and D. R. Radev. Using random walks for question-focused sentence retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 915–922. Association for Computational Linguistics, 2005.

[99] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[100] A. Pilz and G. Paaß. From names to entities using thematic context distance. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 857–866. ACM, 2011.

[101] R. L. Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

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

[103] J. Pound, A. K. Hudek, I. F. Ilyas, and G. Weddell. Interpreting keyword queries over web knowledge bases. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 305–314. ACM, 2012.

[104] C. Quoc and V. Le. Learning to rank with nonsmooth cost functions. *NIPS'07*, 19:193, 2007.

[105] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th international conference on World wide web*, pages 1171–1172. ACM, 2010.

[106] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.

[107] X. Ren, Y. Wang, X. Yu, J. Yan, Z. Chen, and J. Han. Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 23–32. ACM, 2014.

[108] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation extraction with matrix factorization and universal schemas. 2013.

[109] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimization. *Information Retrieval*, 10(3):321–339, 2007.

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

[111] B. Rozenfeld and R. Feldman. Self-supervised relation extraction from the web. *Knowledge and Information Systems*, 17(1):17–33, 2008.

[112] N. Sarkas, S. Paparizos, and P. Tsaparas. Structured annotations of web queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 771–782. ACM, 2010.

[113] U. Sawant and S. Chakrabarti. Learning joint query interpretation and response ranking. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1099–1110. International World Wide Web Conferences Steering Committee, 2013.

[114] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311. Association for Computational Linguistics, 2006.

[115] R. F. Simmons, S. Klein, and K. McConlogue. Indexing and dependency logic for answering english questions. *American Documentation*, 15(3):196–204, 1964.

[116] R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*, 2004.

[117] R. Soricut and E. Brill. Automatic question answering using the web: Beyond the factoid. *Information Retrieval*, 9(2):191–206, 2006.

[118] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing & Management*, 36(6):779–808, 2000.

[119] M. Surdeanu and M. Ciaramita. Robust information extraction with perceptrons. In *Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07)*, 2007.

[120] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

[121] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[122] D. Vallet and H. Zaragoza. Inferring the most important types of a query: a semantic approach. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 857–858. ACM, 2008.

[123] E. M. Voorhees and D. M. Tice. The trec-8 question answering track evaluation. In *TREC*, 1999.

[124] M. Wang. A survey of answer extraction techniques in factoid question answering. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2006.

[125] Q. Wang, J. Kamps, G. R. Camps, M. Marx, A. Schuth, M. Theobald, S. Gurajada, and A. Mishra. Overview of the inex 2012 linked data track. In *Copyright cG2012 remains with the author/owner (s). The unreviewed pre-proceedings are collections of work submitted before the December workshops. They are not peer reviewed, are not quality controlled, and contain known errors in content and editing. The proceedings, published after the Workshop, is the authoritative reference for the work done at INEX.*, page 11, 2012.

[126] I. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30, 2008.

[127] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[128] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[129] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 475–482. ACM, 2008.

[130] X. Yao, B. Van Durme, and P. Clark. Automatic coupling of answer extraction and information retrieval. In *ACL (2)*, pages 159–165, 2013.

[131] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web*, pages 1001–1010. ACM, 2010.

[132] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 347–350. ACM, 2013.

[133] X. Yu, H. Ma, B.-J. P. Hsu, and J. Han. On building entity recommender systems using user click log and freebase knowledge. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 263–272. ACM, 2014.

[134] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2004.

[135] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.

[136] G. Zhou, M. Zhang, D. H. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. *EMNLP-CoNLL 2007*, page 728, 2007.