Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive, 5 - 38123 POVO, Trento - Italy
http://disi.unitn.it

KEYWORD QUERY TO GRAPH QUERY

Zekarias Kefato
zekarias.kefato@studenti.unitn.it
Matteo Lissandrini
ml@disi.unitn.it
Davide Mottin
mottin@disi.unitn.it
Themis Palpanas
themis@disi.unitn.eu

October 2013

Technical Report # DISI-14-003

Abstract

Keyword query to graph query is a problem which aims at mapping entities and relationships mentioned in a query to entities and relationships in a knowledge-base. It has a very important application in different domains like that of information retrieval, data mining, genomics etc.

Obviously keyword queries are preferred by users because they are based on natural language. Even though this point is a plus from the users point of view, it is a challenge from the machines perspective. This is because of the fact that natural languages are ambiguous and keyword queries are over specified.

In this work a model decomposed into three sub-components is proposed. The first sub-component is used to remove ambiguity from the keywords and map them to clearly known entities in a knowledge base. The remaining two sub-components are used to extract interesting relationships between the clearly known entities. At the end top-k output sub-graphs are returned as the final output of the model. These results show mappings of the keywords to entities and the relationship between the mapped entities.

Experimental results proved that the model has a good performance in terms of both quality and efficiency. Particularly the quality is similar to what is reported in state-of-the-art works. The experiments are performed on both synthetic and real queries.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

It has become very common to use search engines to look for a wide variety of concepts on the web by using keywords. With a very little clue, usually keywords, in mind on a particular subject, users make use of search engines to explore more. As in the traditional search engines, one way of quenching this quest is providing documents which talk about the keywords. Apart from that however, it will be more appealing and interesting to a user, if he could get semantically enriched results, instead of a mere occurrence based results. In particular, this can be done for those keywords which target real world entities.

Thanks to the availability of massive knowledge-bases on the web, this need can be satisfied by employing them. Several combined efforts has already led to the emergence of various homogeneous and heterogeneous knowledge bases [1, 2, 20, 21, 26] which contains facts about real world entities .

However due to the massiveness of these knowledge bases, there is desideratum for efficient and effective techniques at the back end of query processing frameworks.

In this particular work, we wish to efficiently and effectively extract meaningful and interesting relationships that exist between entities mentioned (entity mentions) in a query. Given an entity based query, we aim at mapping those entity mentions, which are inherently ambiguous, to a clearly known entities in a

knowledge base, and extract the most interesting relationships that hold between them.

Assuming that we have a knowledge base modeled as a graph, a model which is decomposed into three interdependent sub-components is proposed in this work. The first sub-component is the disambiguation sub component, where its goal is to produce disambiguated entities for each entity mention by leveraging a knowledge base. Each entity mention will have a list of candidate entities extracted from the knowledge base according to some string similarity measure. Because these lists are very long, the aim of this sub-component is to generate top-τ candidate entities.

And the second sub-component, the candidate sub-graph generator, extracts a local sub-graph that strongly interconnect candidate entities represented by nodes (entity nodes) of the graph. This sub-component generates a candidate sub-graph between all ties, that is between the candidate entities of different entity mentions. Even though a candidate sub-graph strongly interconnects the entity nodes, most of the connections are meaningless and noisy.

The third and last sub-component will take the responsibility of setting aside meaningless and noisy relationships and picks up the interesting ones according to some given interestingness criteria.

The *disambiguation* task in this particular work is very similar to that of *entity linking* [7, 8, 19, 27]. In a nutshell, their difference is, in the later case mostly there will be a document from which entity mentions are extracted. However, in the former case it is from an over specified query that entity mentions are extracted.

Closely related studies [3, 12, 13, 14, 17, 18, 22] with different assumptions and targeted applications has already been done. Some however, either implicitly or explicitly aimed at extracting the most interesting or informative relationships between entities. Nonetheless [3, 18] focused on a pair of entities, [12, 13, 22] considers only entity keywords, and [17] impose a constraint on the position of keywords in the query. In this work we will consider not only pair of entities, but any number of entities and we also make use of relationship mentions whenever they exist. Moreover we impose no constraint on the position of keywords so that we might be able to extract relatively complex relationships. Therefore we can

informally state the problem as mapping of both entity mentions and relationship mentions of a given query, into known entities and relationships in the knowledge base.

1.2 Motivating Example

Suppose a user is searching for two related entities for which he has some knowledge about. For example, for two scientists “Friedrich Hegel" and “Heinrich Hertz". A traditional search engine may provide him a list of documents that may talk about these entities. However, the user might also be interested to know existing facts or some useful relationships between them. It will be a very daunting, if not impossible, task to manually extract these facts and relationships from the documents provided by the search engine. Thus, possible results that might be of interest for the user are:

1. Both are scientists, and philosophers
2. Friedrich Hegel is a Physician and Heinrich Hertz’s son “Carl Hellmuth Hertz" is Physician

Cases like the first one are relatively easy to be extracted, however cases like the second one are a little bit involved, and need a careful consideration. Let us further suppose that the user has included a keyword indicating relationships between entities, consider the following example “Comedy starring Lili Tomlins". Where “starring" being a keyword indicating some sort of relationship between the two entities (concepts) “Comedy" and “Lili Tomlins". One might bring an argument about the possible interpretations of the query; however we set aside this argument by simply taking one valid interpretation of the query. Then, one possible interpretation is, the user is asking for movies with “Comedy" genre, where “Lili Tomlins" is “starring". Suppose that this interpretation is the intent of the query, then the user is most probably looking for results of “starring" embedded relationships between “Comedy" and “Lili Tomlins". We refer “Friedrich Hegel", “Heinrich Hertz", “Comedy" and “Lili Tomlins" as entity mentions, and “starring" as relationship mention.

Therefore a system claiming to support such a feature should be able to provide some facts about and relationships between entities. This will also to some extent bridge the semantic gap in the traditional search engine.

Even though there are existing techniques for this task, we claim that there are still important gaps that needs to be addressed. Because of the existence of our approach this problem can be solve by considering that

- There can be any number of entity mentions in the query
- There is an inherent ambiguity on any keyword (mental model) that a user provides and there should be a way to disambiguate it.
- No constraint should be imposed on the order of the keywords position in the query, the system should automatically take care of this while extracting relationships.
- A ranked list of results should be provided
- The solution has to be online, or at least fairly close to online.

We know of no work addressing all these at once; and these are the main contributions of this work. Therefore, we believe that the realization of this work is very important.

Chapter 2

Problem

2.1 Problem Definition

We assume that we have a set of keywords, where these keywords might correspond to real world entities e.g Person, location, company, thing, etc. They might correspond to some kind of relationship between the entities such as, parental, containment, marriage, position, etc.

It might be the case that a keyword can be specified by using an atomic word or as a combination of atomic words. Generally a keyword is composed of n words, where we refer to it as n-gram keyword, and it is formally stated as:

Definition 2.1.1 (n-gram keyword) *An n-gram keyword is a sequence* $k = \langle w_1,\ w_2,\ ...\ w_n \rangle$ *of atomic words of cardinality n, where* $w_i \in V$ *(any human language Vocabulary).*

Thus, we give the formal definition for a query as:

Definition 2.1.2 (Query) *A query is a represented as a set of n-gram keywords* $\mathcal{N}$*, i.e.* $\mathcal{N} = \{k_1, k_2,\ ...\ k_m \mid m \geq 2\}$*. Where each n-gram keyword is categorized as either n-gram entity mention keyword,* e_m *or n-gram relationship mention keyword* r_m*.*

An entity mention is a mention of any real world entity in a query and relationship mention is mention of a relationship between the entity mentions. It is not necessary for n to be the same across n-gram keywords (i.e, each n-gram

keyword might have a different value of n). For brevity sake, an n-gram keyword is simply termed as keyword from this point on.

In addition to a query we assume that we have access to a knowledge base, which contains facts about entities and the relationships between them. A knowledge base is formally defined as follows.

Definition 2.1.3 (Knowledge-base) *Let $\mathcal{E}$ be a finite set of entities, and let $\mathcal{R}$ be a finite set of relationship between entities in $\mathcal{E}$. Each relationship $r \in \mathcal{R}$ is defined on an alphabet of relationships $\Sigma_{\mathcal{R}}$; Thus a multigraph $\mathcal{G}_{\mathcal{KB}} = \langle \mathcal{E}, \mathcal{R}, \Sigma_{\mathcal{R}} \rangle$ is called Knowledge base.*

Every entity in the Knowledge base is represented by a node $e \in \mathcal{E}$, and there are a finite set of labeled relationships $\{r_i \mid i = 1, ..., t\} \subseteq \mathcal{R}$ that goes from and come into this entity from and to a (finite) set of other entities $\{e_i | i = 1, ..., m'\} \subseteq \mathcal{E}$. Every relationship $r \in \mathcal{R}$ is an edge and it has an associated label $l \in \Sigma_{\mathcal{R}}$ where the edges are directed.

Definition 2.1.4 (Candidate entity) *An entity $e \in \mathcal{E}$ which corresponds to an entity mention keyword e_m is called candidate entity. The set of candidate entities of an entity mention e_m, is represented by $\mathcal{E}(e_m)$ Such that $\mathcal{E}(e_m) \subseteq \mathcal{E}$*

Throughout the paper, we interchangeably refer to candidate entities as entities and to entity mentions as mentions.

Therefore, given a query as a set of n-gram keywords $\mathcal{N} = \{k_i \mid i = 1, ..., m\}$ and a knowledge base $\mathcal{G}_{\mathcal{KB}} = \langle \mathcal{E}, \mathcal{R}, \Sigma_{\mathcal{R}} \rangle$, our goal is, to find a mapping of each keyword to entities and relationships in the knowledge base. Such mapping yields a connected sub-graph $\mathcal{G} = \langle \mathcal{E}', \mathcal{R}', \Sigma'_{\mathcal{R}} \rangle$ of the knowledge base, $\mathcal{G} \subseteq \mathcal{G}_{\mathcal{KB}}$.
Thus, the core of our problem, which is the mapping, is formally stated as follows.

Problem 2.1.1 (Mapping) *Given a set of m keywords $\mathcal{N} = \{k_i | \, i = 1, ..., m\}$ as input, we wish to find a* MAPPING*, which is a function $\phi : \mathcal{N} \rightarrow \mathcal{G}_{\mathcal{O}}$ that will give us a set of small connected output sub-graphs, $\mathcal{G}_{\mathcal{O}} = \{\mathcal{G} | \mathcal{G} \subseteq \mathcal{G}_{\mathcal{KB}}\}$.*

And hence the last challenge is, given a set of output sub-graphs $\mathcal{G}_{\mathcal{O}}$, we wish to find a ranking function $\mathcal{F} : \mathcal{G}_{\mathcal{O}} \rightarrow \mathcal{G}'_{\mathcal{O}}$, to produce top-K ranked output sub-graphs.

2.2 Auxiliary Definitions

In the following additional definitions are given. They are simply given in order to provide more clarity to the sections to follow.

Definition 2.2.1 (Full query) *is a keyword query which contains both entity mention keywords and relationship keywords as part of it's keyword sequence, and it is formally given as:*
Let $EM = \{e_i | i = 1, ..., t\}$ *be the set of entity mention keywords and* $RM = \{r_i | i = 1, ..., u\}$ *be the set of relationship keywords, then a full query is a set* $\mathcal{Q}_{\mathcal{F}} = \{EM \cup RM\}$

Definition 2.2.2 (Partial query) *is a query which contains only entity mention keywords as part of it's keyword sequence, and it is formally given as a set* $\mathcal{Q}_{\mathcal{P}} = EM$

The above definitions will bring us to the following definition, which is the type of a query.

Definition 2.2.3 (Type of query) *is a property of a given keyword query indicating whether the query is a full query or a partial query.*

Definition 2.2.4 (Type of a keyword) *is an indicator of the nature of keywords in a given keyword query, i.e. whether the keyword is a entity mention keyword or relationship keyword.*

Example 2.2.1 *Consider the following user query*
$Q =$ *"C. S. Lewis's author of the Chronicles of Narnia."*
The query represented as a set of n-gram keywords will be
$\mathcal{N} = \{$*"C. S. Lewis", "author of", "the Chronicles of Narnia"*$\}$.
The set of n-gram keywords with keyword type entity mention are
$EM = \{$*"C. S. Lewis", "the Chronicles of Narnia"*$\}$
The set of n-gram keywords with keywords type relationship are
$ER = \{$*"author of"*$\}$

The query in the above example has the type full query.

Chapter 3

Literature Review

Though machines understand formal languages, it is apparently clear that they are not handy for communication among *homo sapiens*. Natural languages are so easy and preferred for personal communication and if possible even with machines. These trend has already led scientists, long before, to consider communication between a human being and a machine using natural languages. Applying keyword based communication, more particularly in the information retrieval community, is fairly ubiquitous. Obviously, this very work also revolves around keywords, and a couple of other studies which are some how related to this are briefly observed as follows.

3.1 Entity Linking

Entity linking like the *keyword query to graph query mapping*, has a goal of mapping entity mentions in a document to entities in a knowledge base. However, these mappings are usually for the sake of disambiguating the entity mentions. It doesn't go further into dealing with relationships between disambiguated entities.

Collective entity linking [8] is one which addresses this problem. This is a graph based approach, which uses the graph representation called "Referent Graph" to tackle this problem. The graph captures the local compatibility relation, a relation between entity mentions and candidate referent entities, and the global interdependence between entity linking (EL) decisions. After observing

that considering the global interdependence between entity linking decisions, instead of only considering the local compatibility relation, will give a more effective entity linking result, they exploited this interdependence through the corresponding candidate referent entities in the referent graph. This is what they referred as collective entity linking.

On top of that, the graph structure is used to propagate and iteratively update evidence score starting from an initial evidence score of entity mentions. The initial score is directly obtained from the entity mentions relative importance for the document they are extracted from. And this evidence is considered as a prior importance of the mentions. It is used to anticipate which mention is more important.

A final inference on the entity linking is made after this evidence is iteratively propagated using the stochastic transition matrix; and hence a ranking score for each candidate.

Another approach which is followed by the same authors of the above work is [7] - *A Generative Entity-Mention Model for Linking Entities With Knowledge Base*. In this approach, they used a generative approach, which models the likelihood that a given entity will be a referent entity of an entity mention as a combination of three heterogeneous probabilistic distributions. The distributions model three knowledge, *i.e* the entity popularity, name and context knowledge.

The popularity knowledge is used to know the likely hood of an entity appearing in a document, so that it can help to identify the possible referent entities for an entity mention. The other knowledge, name knowledge will provide the possible names of the entity and the likelihood of a name referring to a specific entity. The last knowledge, which is also applied in their graph-based approach is the context knowledge. This knowledge tells us the likelihood of a name appearing in a given context. Therefore, the final decision to select a referent entity for an entity mention is made based on the one which maximizes the likelihood of the referent entity on the combination of these three heterogeneous models.

Related to the entity linking task another similar work is [19] LINDEN: linking named entities with knowledge base via semantic knowledge. They also used a semantic network graph to link entity mentions in a document with candidate entities obtained from knowledge base. A given document which contains the

entity mentions is scanned to obtain Wikipedia concepts. This concepts and the candidate entities together with Wikipedia articles linked with the concepts and the taxonomy of the Knowledge base (YAGO - [20, 21]), are used to construct the Semantic network.

Four statistics, (Link Probability - LP, Semantic Associativity - SA, Semantic Similarity - SS, and Global coherence - GC) computed by leveraging the semantic network are used as features for each candidate referent entity. In addition, a learned weight vector is used to capture the different degrees of importance that these features have. A final score to rank the candidate entities is computed by taking the product of the weight vector and the feature vector.

As one can easily observe the entity linking problem goes upto dis-disambiguating entity mentions from a given document, however in our work we not only wish to disambiguate entity mentions, we would also like to analyze the relationship between the disambiguated entities in the knowledge base. The other clear difference is that, while our study considers entity mentions and relationship mentions in short queries, the entity linking problem is almost studied only for documents with rich context information. In a more succinct expression, it is a subset of our problem.

3.2 Keyword Interpretation

Keyword query interpretation is a problem that is getting an increasing research interest because of the trend towards semantic web. In this problem in stead of a simple occurrence based search for keywords, the goal is to understand the intent of keywords and return facts about them. Relatively speaking this is a problem which is more close to ours.

Interpreting Keyword Queries over web knowledge bases [16] is one among state-of-the-art works in this direction. The authors target is interpreting short and ambiguous keywords over a knowledge base. The approach is a method which will find the best or top-k formal semantics of the user query by leveraging a knowledge base and coarse linguistic structures of keyword queries. They first annotate the given keyword query using semantic constructs which are learned from part of speech tags of the query terms. And they use semantic summary,

which is the set of semantic constructs used for the term annotation by respecting their order, to abstract the query content. This term annotation is also used to segment the query into keyword phrases. Semantic summaries are then used to learn query templates again using an annotated query log. Like the semantic summaries, query templates are also used to abstract the formal query structure.

Finally query templates and term annotation are combined to give a structured query. Mapping the structured keyword query to the knowledge base to get the formal query interpretation (semantic query understanding) is carried out by extending their previous work [17] - Expressive and flexible access to web-extracted data: a keyword-based structured query language.

In this other work, they used two domain related data sources, a knowledge base and document corpus; they indexed them offline for efficient search. A disambiguation graph is then constructed by extracting knowledge base items (What are referred as candidate entities in this work). The extraction takes into account the fact that selected knowledge base items should form semantically coherent query and also represent the user's keyword query (that is, it should represent the users "intention" well). The disambiguation graph is a fully connected t-partite (for t conjuncts) graph with partitions. Each partition (sub query) is list of knowledge base items for each keyword phrases in the structured query. All partitions are fully connected by respecting nested boundaries (There are no edges from a partition to another partition, if one of the partitions is another nested level).

Therefore a possible formal interpretation of the query is a path that spans all the partitions. To find the best or top-k interpretations a score is computed for each node (syntax similarity) and each edge (Knowledge base support). Therefore, a spanning path will get it's score from this two values and hence rank of a formal interpretation. This work, specially [17] is very similar to our problem, however they are different because their target includes giving answer to queries and on top of that we follow a different approach to tackle the challenge in general. Moreover, unlike [17], in this work we have removed assumptions as it is described in Chapter 1

In their work, [11] Understanding UserâĂŹs Query Intent with Wikipedia, they aimed at identifying the intent of a given query in three specific domains,

using the Wikipedia knowledge base. It's shown that they have built an intent identifier which has the generalization ability to other domains, however. They first build a Wikipedia link graph in which each entry denotes a link relation either within concept articles or between articles and categories. Then they obtain a measure of how probable each concept in the link graph belonging to the defined intent, through a random walk algorithm on the built Wikipedia graph. For those search queries which are not covered by Wikipedia concepts, they applied the technique proposed by [5], This technique (Explicit semantic analysis) utilizes Wikipedia for feature generation for the task of computing semantic relatedness. This is done by first issuing a search query for a search engine to enrich the features of the query with the top-K returned search result snippets. A list of relevant Wikipedia concepts are obtained by ESA for every sentence, where a sentence is created by segmenting the title and description of these top-K results. Finally they used summarized max-min normalized rank score of retrieved concepts of all the sentences, and then the prediction is made according to the sum of the intent probabilities for the top M highest-scoring concepts, and hence ranked query intents. This is also a more similar problem with that of the entity linking problem in a sense that it seeks for mapping primarily, which is still, so to speak, a subset of our problem.

3.3 Relationship explanation

Most popular search engines like Google, Bing, Yahoo! provide related entities for entity based keyword queries. That is given an entity (Eg. Person, Location, Organization) as a query, in addition to returning documents as a result, these search engines provide another section which gives related entities with the one searched for. In *relationship explanation* then the purpose is to give some explanation on the relationship between the entity searched for and the suggested related entities. This is done for each related entity independently with the searched entity. And because of this the focus of the problem will be limited to pair of entities only.

One state-of-the-art work is [4] REX: Explaining relationships between entity pairs. The goal of REX is explaining the relationship between two entities. Given

two entities, one provided by the user searching for an entity, and the second provided by a system which returns a set of entities considered to be related to the entity that the user is looking for. This work only focus on a pair of entities, and explains why the later (related) entity is considered to be related. REX is different from our work, in two ways that it first considers only two entities, and second its purpose is only to explain a relationship between two entities which makes the problem some how narrow because the related entities are close, since they are already considered to be related. Unlike REX, our work targets not only two entities, rather an arbitrary number of entities which might even be totally un-related to each other.

Chapter 4

Methodology

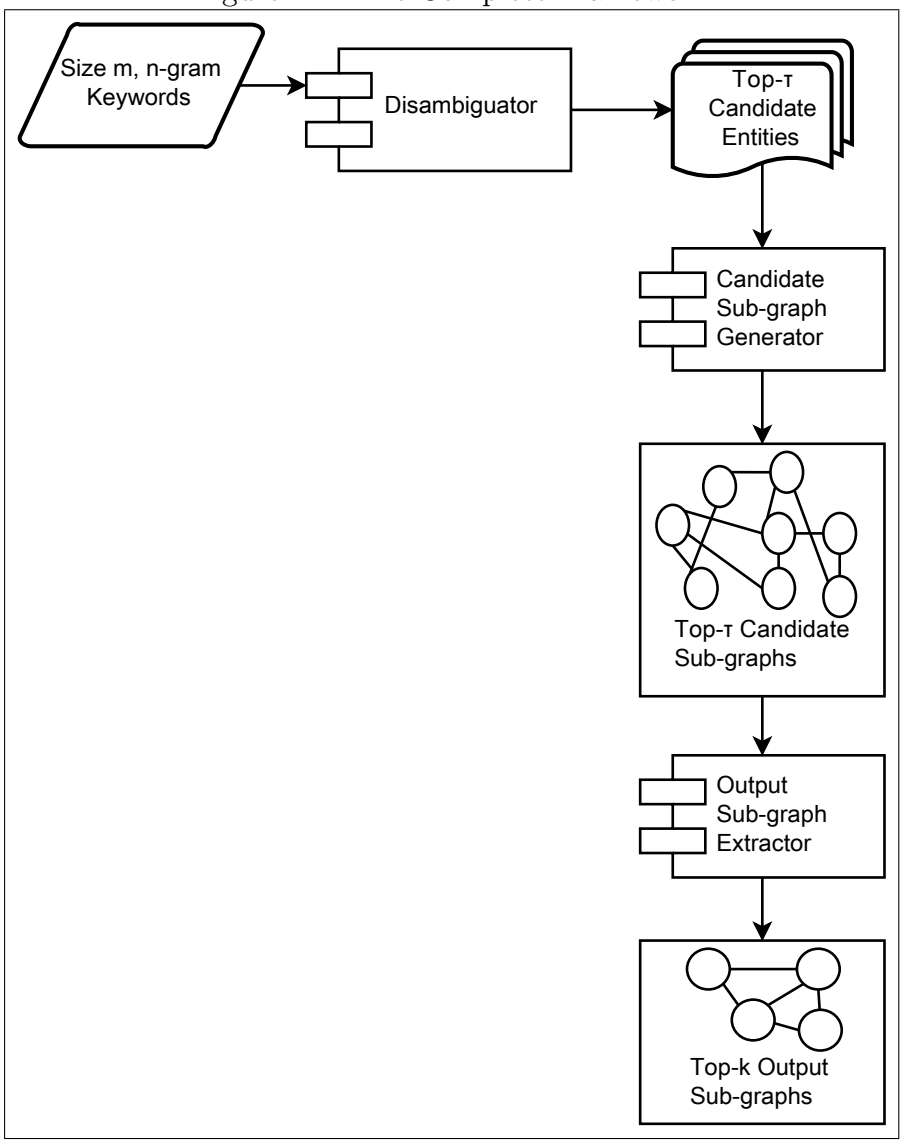
4.1 Overview

Our proposed model to the general problem of mapping keywords to real world entities and relationships of the knowledge base is designed by decomposing into three sub-components. The complete framework is shown in Figure 4.1. The input of this model are a set of keywords, and the discussion of how we obtain the keywords is left out because the problem is orthogonal to ours. Existing studies [6, 24] or off the shelf products can be used for this purpose. We therefore directly move to the three constituents where the model is composed of.

1. **Disambiguation:** We are given a set of keywords, entity and relationship mentions. For each of the entity mentions, this step answers to what clearly known real world entity do they refer to. For example, an entity mention Java, in some given query may refer to the programming language Java, the island Java, or the coffee Java. Using the disambiguation model we wish to associate this keyword to the most likely entity from some knowledge base, according to the intent of the query where this keyword is specified in. However, due to the inherent irremovable ambiguity of natural languages the top-τ candidate entities are picked up instead of attempting to pick exactly one.

   In order to carry out this, a graph based approach adopted from [8] is used. The graph has a good property which allows it to capture local compatibility

Figure 4.1: The Complete Framework



(based on string similarity) of an entity mention with its candidate entity and semantic relatedness between the candidate entities of different entity mentions. By relying more on the semantic relationship, we produce top-τ results.

2. **Candidate sub-graph generation:** In the second step, we need our system to generate a candidate connected sub-graph for each tie of entities. In fact, we don't consider all possible ties for it is not feasible, and the candidate sub-graph generation by itself is an expensive operation. Nonetheless only the ties between the top-τ candidate entities of each entity mention are considered.

   In this approach, the knowledge base is explored until a candidate sub-graph that strongly interconnects the given entities is generated. For the exploration purpose a *balanced expansion heuristic* [10] strategy is employed. In this work, expansion is made from the expanded set of an entity with the minimum cardinality.

3. **Output sub-graph extraction:** Once a candidate sub-graph is generated, in step three we aim at extracting the most *interesting* relationship between the entities in the candidate sub-graph. Formal discussion on the measure of interestingness is presented on Section 4.4. Such process will give us a set of informative output sub-graphs.

   In order to produce these output sub-graphs, two basic iterations are used. Essentially these iterations are used in order to assign some interestingness score for all non entity nodes. The first round will indicate nodes local interestingness. That is interestingness w.r.t each entity node independently. The second is used to evaluate the nodes global interestingness, i.e w.r.t all the entity nodes together. In a nutshell, we use the interestingness measure, to pick nodes that should be included in an output sub-graph

Finally a ranking scheme is employed on the resulting set of output sub-graphs based on nodes interestingness value on each sub-graph.

Table 4.1: Basic Symbols Used

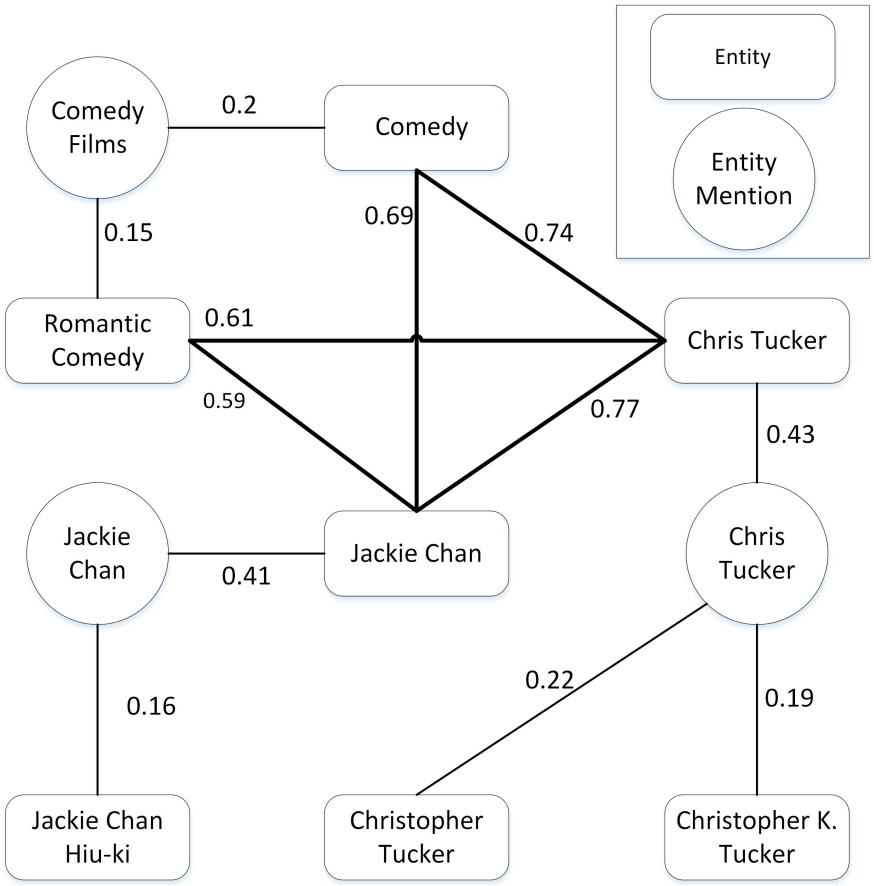
Symbol	Description
Q	User query
$\mathcal{N}$	System query as set of n-gram keywords
$\mathcal{Q}_{\mathcal{F}}$	A user query with *full* query type
$\mathcal{Q}_{\mathcal{P}}$	A user query with *partial* query type
EM	keywords with keyword type entity mention
RM	keywords with keywords type relationship mention
e_m	An n-gram entity mention keyword
$\mathcal{E}(e_m)$	A set of candidate entities of entity mention e_m
e	Candidate entity $e \in \mathcal{E}(e_m)$of an entity mention e_m
r_m	An n-gram relationship mention keyword
$\mathcal{G}_C$	Candidate sub-graph
$Edge(\mathcal{G}_C)$	Edge set of the graph $\mathcal{G}_C$
$\mathcal{V}(\mathcal{G}_C)$	Vertex set of the graph $\mathcal{G}_C$
$\mathcal{G}_R$	Referent graph
$CR(e_m, e)$	Compatible relation scoring function between an entity mention $e_m \in EM$ and an entity $e \in \mathcal{E}(e_m)$
$SR(e, e')$	Semantic relation scoring function between an entity $e \in \mathcal{E}(e_m)$ and an entity $e' \in \mathcal{E}(e'_m)$ where $e_m \neq e'_m$
$\mathcal{G}_O$	A Set of output sub-graphs
$\mathcal{G}_{\mathcal{KB}}$	The multigraph knowledge base

4.2 Disambiguation

Given a set of entity mentions the first challenge is disambiguating them. In order to carry out this task, [8] a graph based approach is adopted. The so called *"Referent Graph"* is used so as to disambiguate each ambiguous entity mention keyword and map them to clearly known entities. The definition of the referent graph which is tailored to this particular problem is given below.

Definition 4.2.1 (Referent Graph) *A **Referent Graph** is a weighted graph $\mathcal{G}_R = \langle V, E, \mathcal{W} \rangle$, with a set of vertices V, a set of Edges E, and a weight function $\mathcal{W}$ of the edges. A vertex $v_i \in V$ is either an entity mention or entity. And an edge $\varepsilon_i \in E$ is either a compatible relation (entity mapping) $CR(e_m, e)$ between an entity mention e_m and an entity e. $e \in \mathcal{E}(e_m)$ or a semantic relation $SR(e, e')$ between two entities e. $e \in \mathcal{E}(e_m)$, and e'. $e' \in \mathcal{E}(e_{m'})$ such that $e_m \neq e_{m'}$. The*

Figure 4.2: Referent graph for the running example



weight function $\mathcal{W}$ will compute weights according to the type of edge (compatible or semantic) it takes.

Consider the following example, "Comedy Film, starring Jackie Chan and Chris Tucker". This will be the running example from this point on. $\mathcal{N} = \{$ Comedy, starring, Jackie Chan, Chris Tucker$\}$. Figure 4.2 shows the *referent graph* for this example.

When ever we want to address an entity on a graph we use the term entity node.

Among the good properties of the Referent graph, the major ones are, it captures (1) local compatibility between a mention and an entity, and (2) global interdependence structure between different entity mappings (bold edges in Fig-

ure 4.2).

In [8], unlike our case which extract entity mentions from a query, they did the extraction from a document which will provide local context. We simply used string similarity measures to replace the local compatibility measure they have used. String similarities are used, because we assume that the user has at least some clue about what he/she is looking for. However, our approach on the disambiguation is strongly dependent on the global interdependence. That is to say, low string similarities might be preferred because of high global interdependence. The Jaro-Winkler similarity metric is used to obtain CR.

$$CR(e_m, e) = 1 - JaroWinkler(e_m, e) \tag{4.1}$$

where $JaroWinkler$ is a function that computes the Jaro-Winkler distance as:

$$JaroWinkler(e_m, e) = \begin{cases} 0, & \text{if } M = 0 \\ \frac{1}{3}(\frac{M}{|e_m|} + \frac{M}{|e|} + \frac{M-T}{M}), & \text{otherwise} \end{cases}$$

where M is the number of matched characters, and T is half of the number of transpositions.

The *global interdependence* is a way to capture semantically related candidate entities, and hence semantic relatedness (SR) measure. This is motivated by the fact that, the mapping of a particular entity mention e_m to a certain candidate entity $e \in \mathcal{E}(e_m)$ will increase or decrease the likelihood of the mapping of another entity mention $e_{m'}$ to a certain candidate entity $e' \in \mathcal{E}(e_{m'})$ where $e_m \neq e_{m'}$. Consider Table 4.2, the illustration given for the running example.

The intuition behind is that, for example, without loss of generality, if we map the entity "Jackie Chan" ($e^1_{m\_1}$) to the entity mention "Jackie Chan" (e_{m_1}), that will increase the likelihood of mapping the candidate entity "Chris Tucker" ($e^1_{m\_2}$) to the entity mention "Chris Tucker" (e_{m_2}). This is due to the strong semantic relatedness between $e^1_{m\_1}$ and $e^1_{m\_2}$ (Look at Figure 4.2). On the other hand the likelihood of mapping the remaining candidate entities ($e^i_{m\_2} \in \mathcal{E}(e_{m\_2}) \setminus e^1_{m\_2}$) will decrease, for there is no semantic relation with "Jackie Chan" ($e^1_{m\_1}$).

Table 4.2: Entity Disambiguation Illustration

Entity Mention		Candidate Entity		Likely hood indicator
Id	Name	Id	Name	
e_{m_1}	Jackie Chan	$e^1_{m\_1}$	Jackie Chan	$\rightarrow$ (*Mapped*)
e_{m_1}	Jackie Chan	$e^2_{m\_1}$	Jackie Chan Hiu-ki	
e_{m_2}	Chris Tucker	$e^1_{m\_2}$	Chris Tucker	$\uparrow$ (Increased due to mapping)
e_{m_2}	Chris Tucker	$e^2_{m\_2}$	Christopher Tucker	$\downarrow$ (decreased due to mapping)
e_{m_2}	Chris Tucker	$e^3_{m\_2}$	Christopher K. Tucker	$\downarrow$ (decreased due to mapping)
e_{m_3}	Comedy Films	$e^1_{m\_3}$	Comedy	$\uparrow$ (Increased due to mapping & prev increase & decrease)
e_{m_3}	Comedy Films	$e^2_{m\_3}$	Romantic Comedy	$\downarrow$ (decreased due to mapping & prev increase & decrease

By the same token, the likely hood of mapping the candidate entity “Comedy" ($e^1_{m\_3}$) to the entity mention “Comedy Films" (e_{m_3}) will increase. And this is because of two facts. One because of the initial mapping for the first entity mention (e_{m_1}) and second the propagated increase of likely hood to the candidate entity ($e^1_{m\_2}$) of the second entity mention (e_{m_2}). In other words, since the entity ($e^1_{m\_3}$) has a strong semantic relation to the already picked entities ($e^1_{m\_1}$ and $e^1_{m\_2}$) than the other entity ($e^2_{m\_3}$), it is more likely that its mapping is valid. Apparently, for the last entity, the likely hood decreased because of its weak semantic relation. The abstraction of the disambiguation model is given in Algorithm 1.

We use the same approach as in [9] to compute semantic relatedness measure. And it is given as follows:

$$SR(e, e') = 1 - \frac{log(max(|\mathcal{L}_e|, |\mathcal{L}_{e'}|)) - log(|\mathcal{L}_e \cap \mathcal{L}_{e'}|)}{log(|\mathcal{G}_{\mathcal{KB}}|) - log(min(|\mathcal{L}_e|, |\mathcal{L}_{e'}|))} \tag{4.2}$$

where e and e' are the two entities of interest; $\mathcal{L}_e$ and $\mathcal{L}_{e'}$ are the sets of all entities that link to e and e' in the knowledge base respectively, and $\mathcal{G}_{\mathcal{KB}}$ is the entire knowledge base. The study of [15] shows that this is an effective and low-cost approach.

The underlying graph, referent graph, used to compute the above two scores is constructed according to the following procedures.

- For each entity mention of the query, retrieve candidate entities from the given knowledge base. Retrieval is done with simple fuzzy string matching against entity names and aliases.
- For each entity mention e_m create a link from them to all the corresponding candidate entities $e_i \in \mathcal{E}(e_m)$ generated in the previous step, and weight the edges according to CR function.
- For each pair of candidate entities $\langle e, e' \rangle$ where $e \in \mathcal{E}(e_m)$ and $e' \in \mathcal{E}(e_{m'})$, such that $e_m \neq e_{m'}$ add a weighted edge according to the SR function, if the measure is not zero, otherwise add no edge for this particular pair.

After the graph is constructed, the nodes in the graph will get importance score as described in the following section.

4.2.1 Nodes importance scoring

Following the above procedure will give us an edge weighted graph. In order to give importance values for the candidate entities a random walk with restart strategy is employed. Unlike [9], which gives initial evidence according to the entity mentions relative importance to the document from which they are extracted, in this work an equal initial score is given to all the entity mentions. This is because, there is no knowledge of the relative importance of each entity mentions of the query. Rather they are assumed to be equally important.

Apparently, the transition matrix for the random walk can be obtained from the weighted referent graph. And as discussed earlier the staring vector, indicating the initial importance, is initialized by distributing an equal starting probability to all the entity mentions and zero for the candidate entities.

Algorithm 1 Disambiguation Model

Require: EM ▷ All the entity mention keywords
Require: τ ▷ User parameter τ
Require: r ▷ Restart probability
Ensure: $\tau\mathcal{E}$ ▷ Top-τ entities

procedure DISAMBIGUATEENTITIES(EM, τ, r)
  for $e_m \in EM$ **do**
    $\mathcal{E}(e_m) \leftarrow generateCandidateEntities(e_m)$;
  end for
  for $e \in \mathcal{E}(e_m)$ **do**
    $\mathcal{W}(e_m, e) \leftarrow CR(e_m, e)$;
  end for
  for $e \in \mathcal{E}(e_m)$ **do**
    for $e' \in \mathcal{E}(e'_m) \wedge e_m \neq e'_m$ **do**
      $\mathcal{W}(e, e') \leftarrow SR(e, e')$;
    end for
  end for
  ▷ a random walk with restart on $\mathcal{W}$ with restart probability r
  $impVector \leftarrow RandomWalk(\mathcal{W}, r)$;
  $\tau\mathcal{E} \leftarrow enumAllTop\tau(EM, \mathcal{E}, \mathcal{W}, impVector, \tau)$;
  return $\tau\mathcal{E}$
end procedure

Let ii be the **i**nitial **i**mportance vector and n be the number of entity mentions, then

$$ii = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1/n \\ 1/n \\ 1/n \end{pmatrix} \tag{4.3}$$

The zero's are the initial importance scores for all of the candidate entities and $1/n$ is the initial importance of the n entity mentions of the query.

let $\mathcal{W}$ be the transition matrix of the weighted referent graph. Formally, the final importance score for each of the candidate entities is obtained by propagating

the initial importance using the transition matrix as follows:

$$ii^{t+1} = \mathcal{W} \times ii^{t} \tag{4.4}$$

The super scripts are the indicators of the time that the importance score is obtained.

However the above model has a limitation in the case of nodes that don't propagate importance score. That is, propagated scores upto that node will be trapped and there will be no propagation from that node. So to avoid this, a restart mechanism is allowed by redistributing part of the importance score, restart probability (r), to the initial importance vector. Then Equation 4.4 is reformulated as

$$ii^{t+1} = r \times ii^{0} + (1 - r) \times \mathcal{W} \times ii^{t} \tag{4.5}$$

That is the random walk will be done by respecting the weight (transition) matrix with probability proportional to r or it will be restarted with probability proportional to $1 - r$. Still there is a problem with this modeling too, that is for a large set of nodes it is computationally expensive. And it is obvious that the number of nodes is very large. Hence, the above equation can be solved in a closed form as:

$$ii = (1 - r)(I - r \times \mathcal{W})^{-1} \times ii^{0} \tag{4.6}$$

where I is the identity matrix. And at this stage, ii is the vector which indicates the final importance score of all the candidate entities on the referent graph. Therefore each candidate entity will be given a rank according to their scoring that is based on ii. Table 4.3 shows the ranking of candidate entities for each entity mention based on this score.

However, top ranking candidate entities aren't just taken. This is because of the fact that the top candidate entities might not be semantically related. The detail of this case is presented in the following sub Section 4.2.2.

Finally we wish to find entities which maximize this importance score and semantically related to other entities. The following sub Section 4.2.2 will also

Table 4.3: Importance score of candidate entities

Entity Mention	Candidate Entity	Importance score
Comedy Films	Comedy film	0.17134
	Romantic comedy	0.14366
Jackie Chan	Jackie Chan	0.24403
	Jackie Chan Hiu-ki	0.00152
Chris Tucker	Chris Tucker	0.25096
	Christopher Tucker	0.00094
	Christopher K. Tucker	0.00081

deal with this notion.

4.2.2 Top-τ Path selection

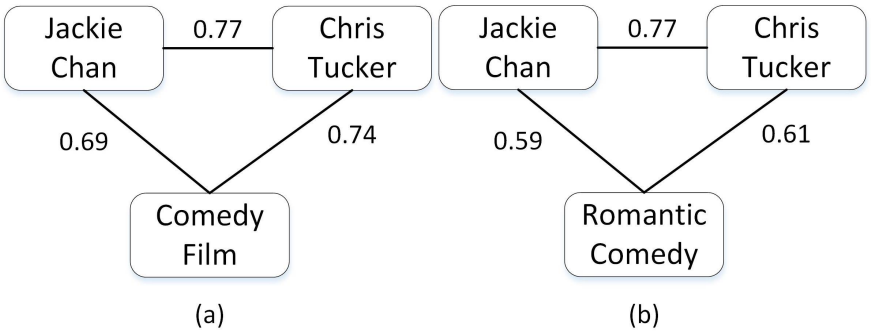
Finding the top-τ entities which maximizes importance score and semantically related to other entities is the same as finding high scoring semantic paths (or cycles). Which is a path along the semantic edges that maximizes the aggregate weights of its edges and nodes.

A straight forward way is simply to take the top-τ candidate entities of each mention. However that will not guarantee us to have semantically coherent entities in the final output. This is due to the fact that we are not always guaranteed for the top-τ entities of each entity mention to be semantically related. For example, as depicted in Figure 4.2, two of the candidates of “Chris Tucker" are not semantically related to any of the candidate entities of the other mentions. Therefore they should be avoided from being part of the top-τ list, or at least we should minimize probabilities of including them. Thus we followed an approach which enumerates all (*ENUMALL*) the semantic edges, referred as semantic path.

The *ENUMALL* approach might suffer from efficiency because of the large candidate entity space. However, a pruning strategy is employed in order to stay in time bound. Those candidate entities which fall bellow a certain CR value are pruned.

ENUMALL works by picking exactly one entity node from each entity mention's set of candidate entities that are connected to the candidate entities in at least on of the other sets. For the referent graph in Figure 4.2, all the possible

Figure 4.3: All semantic paths enumeration for the referent graph in figure 2



semantic path enumerations according to *ENUMALL* are shown in Figure 4.3 (a) and (b). By doing this we favor semantic relations more than compatible relations.

Since our purpose is just ranking, we use sum as an aggregate function. Thus top-τ scoring nodes will be returned from those paths that maximize the aggregate score on the edges and the nodes. One can clearly see that, in this approach effectiveness is highly probable, in a sense that we can obtain the most likely ties between entities which are semantically coherent. However it comes at the cost of efficiency. Nevertheless in order to avoid the efficiency bottleneck, we have already pruned those candidate entities which fall below a certain threshold of compatibility relation. In practice this approach works efficiently for range of threshold values as described in the experimental results section. The pseudo code for this approach is given in Algorithm 2.

The next goal is to find connected sub-graphs, which captures the most interesting relationships between the disambiguated entities (entity nodes), from the knowledge base. The remaining two sub-components will handle this as described in the following sections.

4.3 Candidate Sub-graph Generation

In this step, we aim at retrieving a sub-graph of the knowledge base that strongly interconnects the entity nodes. This task needs exploring the knowledge base until a desired sub-graph is extracted. A *balanced expansion heuristic* technique

Algorithm 2 Top-τ path selection, all possibilities

Require: EM ▷ All entity mentions
Require: $\mathcal{E}$ ▷ Function that extract entities of mentions
Require: $\mathcal{W}$ ▷ Weight function
Require: $impVector$ ▷ Importance vector of entities
Require: τ
Ensure: $\tau\mathcal{E}$ ▷ Top-τ entities

procedure ENUMALLTOPτ(EM, $\mathcal{E}$, $\mathcal{W}$, $impVector$, τ)
  $allPaths \leftarrow \emptyset$;
  for $e_m \in EM$ **do**
    for $e \in \mathcal{E}(e_m)$ **do**
      $weight \leftarrow 0$;
      $path \leftarrow \emptyset$;
      $weight+ = impVector(e)$;
      $path.add(e)$;
      for $e' \in \mathcal{E}(e'_m)$ **do**
        $weight+ = \mathcal{W}(SR(e, e'))$;
        $weight+ = impVector(e')$;
        $path.add(e')$;
        $path.weight(e, e', weight)$;
      end for
    end for
    if $path \notin allPaths$ **then**
      $allPaths.add(path)$;
    end if
  end for
  $top\tau \leftarrow sort(allPaths, \tau)$;
  return $top\tau$;
end procedure

is employed to carry out this task. This approach is preferred because of its good performance in practice, as it is corroborated in [10]. In this method, at each step, expansion is made from an entity node e with the lowest cardinality of expanded set, that is

$$e \leftarrow \underset{e' \in \mathcal{E}(e_m)}{\operatorname{argmin}} |\mathbb{E}(e')| \tag{4.7}$$

where $\mathbb{E}(e')$ is an expanded set of the entity node e'.

To guide the expansion, a similar strategy as in [12] is adopted and combined

with an additional strategy that is employed to suite the specific knowledge base used in this work. The adopted strategy is based on co-occurrence statistics that encodes two distinct scores, which are called *Notability for* and *Popularity* score.

Let $f = (e, r, e')$ be a fact, such that $\mathcal{G}_{\mathcal{KB}} \models f$ (read as, f is modeled by or true in $\mathcal{G}_{\mathcal{KB}}$). For any fact f mentioned throughout this paper $\mathcal{G}_{\mathcal{KB}} \models f$.

Let the structure of any fact f be $f =$ (source, relation, destination). source, relation, and destination are simply position indicators. Then,

- *Instance fact*: it is a fact where all of its position are occupied by actual values from the knowledge base. For example $f =$ (Elvis Presley, is-A, Singer), the general form is $f = (e_v, r_v, e'_v)$
- *Fixed fact*: it is a fact where only portion of its position is occupied by values. For example $f =$ (e, is-A, Singer). The general format is the same as the previous one, it is different it only has the sub-script v for those positions where there is a value. For the above example, it will be $f(e, r_v, e'_v)$.

Consider the following three instance facts $f_1 =$ (Elvis Presley, profession, Singer), $f_2 =$ (Elvis Presley, profession, Author), and $f_3 =$ (Jackie Chan, profession, Singer). The desired scores are obtained by changing these facts to fixed facts as follows. For illustration sake let as fix f_1.

First $f_1 =$ (Elvis Presley, is-A, e'). This fixing will let us know the Notability score for Elvis Presley. Intuitively we are asking what is the thing that Elvis Presley is well notable for. By replacing different values for e' we will get different scores, and the one with the highest score is what Elvis Presley is well notable for. Hence we can say that, f_1 will have high notability score than f_2, for Elvis Presley is well notable for being a singer rather than an author.

Second $f_1 =$ (e, is-A, Singer). For this case, intuitively we are asking "who is a popular singer?", such fixing will give us a popularity score. Just as in the first case, we replace different values for e and obtain who is the most popular singer. Then, it clearly follows that, f_1 will get a higher popularity score than f_3, for Elvis Presley is more popular than Jackie Chan for being a singer.

These two scores are obtained using probability estimations. Following [12], the estimations are done by using co-occurrence statistics.

Then we compute the notability and popularity score according to Equations 4.8 and 4.9, respectively as follows.

$$Pr(e'|r_v, e_v) = \frac{Pr(e_v, r_v, e')}{Pr(r_v, e_v)} \approx \frac{f_{oc}(e_v, e')}{f_{oc}(e_v)} \tag{4.8}$$

$$Pr(e|r_v, e'_v) = \frac{Pr(e, r_v, e'_v)}{Pr(r_v, e'_v)} \approx \frac{f_{oc}(e, e'_v)}{f_{oc}(e'_v)} \tag{4.9}$$

where the approximation function f_{oc} computes the co-occurrence or occurrence frequency of its arguments. $f_{oc}(e_v, e'_v)$ computes the co-occurrence frequency of e_v and e'_v; $f_{oc}(e')$ computes the occurrence frequency of e'_v. For implementation case, these statistics are computed from documents which are extracted from the web. [1]

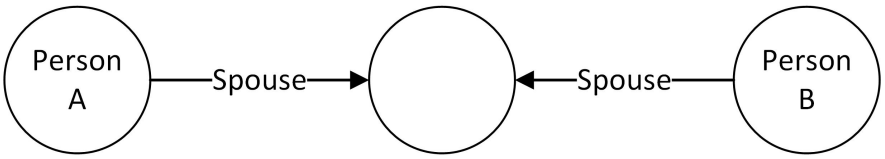
The second technique is used to account for specific cases in Freebase. For example, in Freebase a spouse relationship between any two persons is modeled as in Figure 4.4. Spouse is not the only relationship modeled this way, there are multiple instances of these kinds of modelings. According to observations on the knowledge base, the particular property of the empty intermediary nodes in this kind of cases is, they usually has no labels, and they are connected to a very few nodes. For example the middle node in Figure 4.4 is connected to the node Marriage, which is its type in addition to Person A and B. Often times this kind of nodes are a direct relationships between two entity nodes, which is an indicator for a strong relationship. If only the first approach is considered, then the model will be forced to give zero weights for such kinds of meaningful relationships. Therefore, the so called incident score $\mathcal{IS}$ is added to give a reasonably strong weight for such cases, and it is given by:

$$\mathcal{IS}(e, nil) = 1/Degree(nil) + \mathcal{IL}(l) \tag{4.10}$$

where *nil* stands for the unlabeled node and l for the edge label between e and *nil*. *Degree* is a function that computes the out degree of nil; and $\mathcal{IL}(l)$ is a function that computes the information load on this particular edge with label l. These values tells us how much information is contained on this edge from the

[1] Google Custom Search Engine is used to achieve this

Figure 4.4: Freebase specific modeling



whole knowledge base point of view, and $\mathcal{IL}$ is computed as:

$$\mathcal{IL}(l) = Count(l)/Count(\mathcal{R}) \tag{4.11}$$

where $Count(l)$ is the frequency of edges labeled l, and $Count(\mathcal{R})$ is the total number of facts in the knowledge base.

Even though this case is used to account for a specific knowledge-base, it is still valid for any kind of knowledge base. It can be considered as additional structural information. Some sort of discounting factor can be added to give more emphasis to the first approach, if needed.

Now we have everything that is needed to explore the knowledge base. Therefore exploration can commence by randomly starting from one of the entity nodes.

Let $\mathbb{P}(e)$ be the set of pending nodes after expanding the entity node e. Then starting from the second iteration explorations are made from the pending set of an entity e which maximizes the aggregate score in Equations 4.10 and 4.12.

$$e_p \leftarrow \underset{e_{p'} \in \mathbb{P}(e)}{\operatorname{argmax}} \sum_{e' \in \mathbb{E}(e)} Pr(e'|e_{p'}) + Pr(e_{p'}|e') \tag{4.12}$$

where $Pr(e'|e_{p'})$ is computed as

$$Pr(e'|e_{p'}) = \sum_{\substack{r \\ (e', r, e_{p'}) \in \mathcal{G}_{\mathcal{E}}}} Pr(e'|r, e_{p'}) \tag{4.13}$$

and we know how to estimate the probability on the right hand side from Equations 4.8 and 4.9.

Algorithm 3 Candidate Sub-Graph Generation using BEH

Require: $\mathcal{E}$ ▷ Set of entities
Ensure: $\mathcal{G}_C$ ▷ A candidate sub-graph

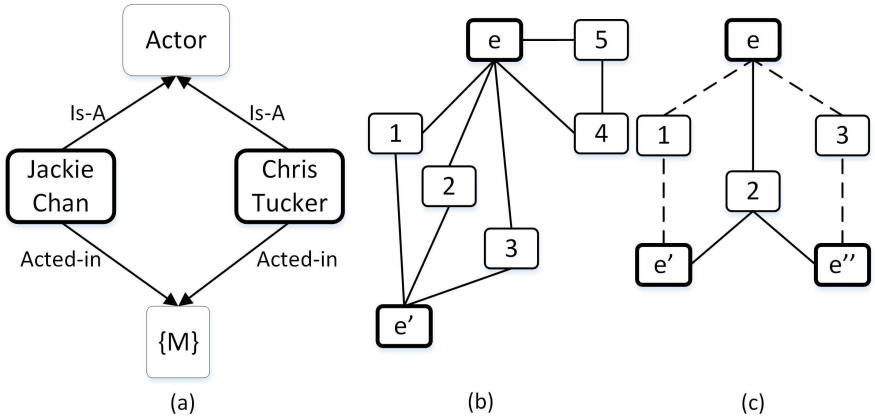
procedure BALANCEDEXPANSION($\mathcal{E}$)
  $\mathcal{G}_C \leftarrow Null$;
  for $e \in \mathcal{E}$ **do**
    $\mathbb{E}(e) \leftarrow \emptyset$; ▷ Expanded set of e
    $\mathbb{P}(e) \leftarrow \{e\}$; ▷ Pending Set of e
  end for
  $buildGraph(\mathcal{G}_C, Null, \mathbb{P})$;
  while $\mathcal{G}_C$ *is not big enough* $\vee \exists e'' \in \mathcal{E}.\ \mathbb{E}(e'') = \emptyset$ **do**
    $e \leftarrow \underset{e_i \in \mathcal{E}}{\text{argmin}}\ |\mathbb{E}(e_i)|$;
    $e_p \leftarrow \underset{e_{p'} \in \mathbb{P}(e)}{\text{argmax}} \sum_{e' \in \mathbb{E}(e)} Pr(e'|e_{p'}) + Pr(e_{p'}|e') + \mathcal{IS}(e', e_{p'})$;
    $N \leftarrow neighbors(e_p)$;
    $\mathbb{E}(e) \leftarrow \mathbb{E}(e) \cup \{e_p\}$
    $\mathbb{P}(e) \leftarrow \mathbb{P}(e) \backslash \{e_p\}$;
    $\mathbb{P}(e) \leftarrow \mathbb{P}(e) \cup \{e_n | e_n \in N \wedge e_n \notin \bigcup_{e \in \mathcal{E}} \mathbb{E}(e) \cup \mathbb{P}(e)\}$;
    $buildGraph(\mathcal{G}_C, e_p, N)$;
  end while
  return $\mathcal{G}_C$;
end procedure

A high level flow of this process is given in Algorithm 3. And its final result is a weighted, connected candidate sub-graph. Where the weight is the sum of the weights obtained in Equations 4.10 and 4.12.

The candidate sub-graph is further pruned to get rid of non-entity nodes with degree one. As it is proven in [4] those are not interesting nodes.

By far we are not making use of relationship keywords in the query. If a query has both entity mention keywords and relationship mention keywords, we want to exploit that in order to boost confidence on what we get from the statistics and incident score based edge weights. Thus, the candidate sub-graph is modified to include the contribution that comes from relationship keywords. And this will be addressed during the final step, in the coming Section 4.4.

Figure 4.5: (a) Edge reward, (b) interestingness w.r.t a single entity node and (c) w.r.t all the entity nodes example



4.4 Output sub-graph extraction

The extraction process commences by adding reward factors to those edges that are related to the relationship mention keywords, if there are any. The interpretation is that, those edges which bear the same meaning as the relationship mention should be favored over other edges. For example, as portrayed in Figure 4.5 (a), for our running example, it will be meaningful to favor the “Acted-in", over the “Is-A" edge. This is because, “Acted-in", is more meaningful than “Is-A" w.r.t the relationship mention $r_m = starring$. Thus a reward factor r_f is added to those meaningful edges as follows.

Let l_ϵ be the label for an edge $\epsilon \subseteq Edge(\mathcal{G}_C)$, taken from an alphabet Σ, where $Edge(\mathcal{G}_C)$ is the edge set of $\mathcal{G}_C$. And $W(\epsilon)$ be the weight on the edge ϵ. Then $W(\epsilon)$ is updated as:

$$W(\epsilon) = W(\epsilon) \times r_f(l_\epsilon, r_m) \tag{4.14}$$

$r_f(l_\epsilon, r_m)$ is a function which gives a score that indicates the words sense similarity. It is computed according to the *vector space model* and their vector is constructed by taking the synset and hypernym set [1] of l_ϵ and r_m.

Even though the edges in the candidate sub-graph interconnects all the entity nodes, we need not all of them. Obviously it is impossible to make any kind of

[1] Wordnet is used for this purpose

meaningful interpretations with all these nodes and edges.

The output sub-graph extraction component of our system, which is presented in this very section, is used to resolve this issue. In order to pick those relevant nodes which will give us the most interesting relationships, we need a way to quantify nodes interestingness. Two kinds of scores obtained during two iterations called:

- *Singular Iteration*
- *Integrated Iteration*

are used to achieve this. Let us present the intuition behind these scores before going into the formal details.

During the first iteration, for each of the nodes in the candidate sub-graph, we compute the likelihood that they will be on any path starting from each of the entity nodes independently. The score obtained in this iteration will only tell us about the interestingness of a node w.r.t each entity node separately, but not w.r.t the whole entity nodes together. It is straight forward to see a node with high value of such a score is not always guaranteed to interconnect all the entity nodes. Figure 4.5 (b) shows one counter example to support this claim.

Assume that nodes 4 and 5 are strongly related to e, that is the edges $\langle$ e, 4 $\rangle$ and $\langle$ e, 5 $\rangle$, perhaps because of high informativeness score assigned to them during the previous step (candidate sub-graph generation). During the first iteration, this will give, for a random surfer doing a random walk starting from e, a high probability of following the path to either node 4 or 5.

Nonetheless, these nodes are not interesting with respect to the whole entity nodes i.e { e, e'}. Thus, it is more plausible, instead of nodes 4 and 5, to assign the highest interestingness score to nodes 1 and/or 2 and/or 3. This is a case dealt during the second iteration. However, prior to its discussion let us first introduce the notions of an *Interconnecting Tree*, *Intersecting Tree* and a *Common Node*.

Definition 4.4.1 (Interconnecting Tree) *For any tree $t \subseteq \mathcal{G}_C$, let $\mathcal{V}(t)$ be its vertex set, and for each entity node e. $e \in \mathcal{V}(t)$; t is an **interconnecting tree**, if for any pair of entity nodes $\langle e, e' \rangle$ there is a path $p' \subseteq t$ and there is no entity node $e'' \in \mathcal{V}(t) \setminus \{e, e'\}$ such that $e'' \in \mathcal{V}(p')$.*

Definition 4.4.2 (Intersecting Tree) *An interconnecting tree t is also called an intersecting tree, if there is at least one intersection point $\forall p' \subseteq t$.*

$$\bigcap_{p' \subseteq t} \mathcal{V}(p') \setminus \mathcal{E} \neq \emptyset$$

Definition 4.4.3 (Common Node) *For any intersecting tree t, a give node $n \in \mathcal{V}(t) \setminus \mathcal{E}$, is called common node, if*

$$n \in \bigcap_{p' \subseteq t} \mathcal{V}(p')$$

The set of such nodes are represented by N_c

Figure 4.5 (c) shows examples of an [non] interconnecting and [non] intersecting trees. The sub-tree along the solid lines is both an *interconnecting* and *intersecting tree* but the one along the dotted lines is none. On Figure 4.5 (c) node 2 is the only *common node*.

The second score is then based on the above notions; the intuition is that given any node $n \in \mathcal{V}(\mathcal{G}_\mathcal{C}) \setminus \mathcal{E}$, this score tells us the likelihood that $n \in N_c$. This alludes to the interestingness of the node from the whole entity nodes point of view.

We compute these scores using the random surfer model. For the first score it is computed by doing a random walk with restart (RWR) which starts from each entity node independently. And for the second score it is computed by doing a RWR starting from each entity node simultaneously. As studied in different works like [12, 22, 23] in RWR the restart probability discourages long connectivity paths in a natural way. In addition, RWRs are well suited to capture the structural connectivity between nodes. These are the motivations behind the singular and integrated iterations. The aggregate score will give the final interestingness value for each non entity node.

4.4.1 Singular Iteration

During the first iteration, we aim at giving interestingness score for all nodes with respect to each entity node independently. $\forall n_i \in \mathcal{V}(\mathcal{G}_C) \setminus \mathcal{E} \wedge \forall e_j \in \mathcal{E}$, we

compute $\mathcal{I}_e(n_i, e_j)$, where $\mathcal{I}_e(n_i, e_j) = [0, 1]$ is an estimation of the interestingness score of node n_i, w.r.t entity node e_j.

In order to compute $\mathcal{I}_e$ we have used a similar strategy as [22]. Where in our case we have used a weighted graph based on informativeness, incident and edge reward scores. Let W be the normalized weight matrix for the weighted candidate sub-graph $\mathcal{G}_C$. For any edge $\langle u, v\rangle | \{u, v\} \subseteq \mathcal{V}(\mathcal{G}_C)$, it is normalized according to the sum of the weights of the outgoing edges of node u.

Let $\mathcal{I}_j$ $[1 \times M]$ be vector of an interestingness score of all the nodes with respect to the the entity node e_j, and $M = |\mathcal{V}(\mathcal{G}_C)|$. Where $\mathcal{I}_j$ is populated by propagating a confidence score (interestingness) of reaching each node through a random walk traversal by starting from an entity node e_j. Since we have a normalized weight matrix W, it is used as a transition matrix.

Initially, nothing is known about the interestingness of each node apart from the fact that we know the random surfer is waiting to start the walk from the entity node e_j. We can represent this state as an initial vector with all of its components set to zero except the one that corresponds to the starting entity node.

$$\mathcal{I}_j^0 = \begin{pmatrix} 0 \\ . \\ . \\ 0 \\ 1 \\ 0 \\ . \\ . \\ 0 \end{pmatrix} \tag{4.15}$$

Therefore, the random walk can be done according to the following recursive transition.

$$\mathcal{I}_j^{t+1} = W \times \mathcal{I}_j^t \tag{4.16}$$

However, this method will not account for discouraging long and loose connectivity paths. According to the studies of [22, 23] Random walk with restarts will properly deal with this kind of situations. Therefore, in order that we can cap-

ture this, we extend the previous model to let the random walker able to restart (RWR) the walk. We allow to restart the random walk by distributing a restart probability, r ($r = 0.85$ is used), on the initial vector as:

$$\mathcal{I}_j^{t+1} = r \times \mathcal{I}_j^0 + (1 - r) \times W \times \mathcal{I}_j^t \tag{4.17}$$

That is the random walk will be done by respecting the weight (transition) matrix with probability proportional to r or it will be restarted with probability proportional to $1 - r$. It is clear that for a large set of nodes this computation is expensive, therefore the above equation can be solved in closed form:

$$\mathcal{I}_j = (1 - r)(I - rW)^{-1}\mathcal{I}_j^0 \tag{4.18}$$

where I is identity matrix, and $\mathcal{I}_j$ is a steady state probability vector indicating the interestingness of each node with respect to the entity node e_j.

4.4.2 Integrated Iteration

As we have stated earlier, in addition to $\mathcal{I}_e$ we need a way to capture interestingness w.r.t the whole entity nodes. That is the aim of this iteration. For each node we will determine it's interestingness w.r.t the whole entity nodes. $\forall n_i \in \mathcal{V}(\mathcal{G}_C) \setminus \mathcal{E}$, we compute $\mathcal{I}_\mathcal{E}(n_i, \mathcal{E})$, where $\mathcal{I}_\mathcal{E}(n_i, \mathcal{E}) = [0, 1]$ is an estimation of the interestingness score of node n_i w.r.t the whole entity nodes in $\mathcal{E}$.

$\mathcal{I}_\mathcal{E}$ is simply computed by combining the $\mathcal{I}_e$ as:

$$\mathcal{I}_\mathcal{E}(n, \mathcal{E}) = \prod_{e \in \mathcal{E}} \mathcal{I}_e(n, e) \tag{4.19}$$

Therefore the output sub-graph extraction model can be formally stated as:

Definition 4.4.4 (Top-K output sub-graph extraction) *For a candidate sub-graph $\mathcal{G}_C$, top-K output sub-graph extraction amounts to finding a set of connected sub-graphs such that the aggregate score function f is maximized:*

$$\max f(\mathcal{E}, \mathcal{G}_C) = \sum_{\substack{n \in \mathcal{V}(p) \setminus \mathcal{E} \\ p \subseteq \mathcal{G}_\mathcal{C}}} \mathcal{I}_\mathcal{E}(n, \mathcal{E}) \tag{4.20}$$

$$\text{Subject to:} |\cup_{p \subseteq \mathcal{G}_\mathcal{C}} \mathcal{V}(p) \setminus \mathcal{E}| < b$$

where b is a node budget (The maximum number of nodes other than entity nodes).

4.4.3 Extraction

For the extraction algorithm, we have used Dijkstra's shortest path algorithm in the following way.

- Sorted all node $n \in \mathcal{V}(\mathcal{G}_\mathcal{C}) \setminus \mathcal{E}$ according to the interestingness ($I_\mathcal{E}$) score. Let this sorted list be referred as Ω.
- Pick a node $n \in \Omega$ with the highest score
- Find a shortest path from each of the entity nodes $e \in \mathcal{E}$ to n within a given radius.
- Synchronize all the paths with the candidate sub-graph, and build a small output sub-graph $\mathcal{G}_\mathcal{O}$.
- If the number of nodes excluding the entity nodes is less than the node budget remove n from Ω and repeat the whole process starting from step 2.

Algorithm 4 gives a high level description of the extraction process. To carry out the extraction using any shortest path algorithm, we first need to have a directed weighted graph. Our candidate sub-graph is already directed sub-graph, and also weighted. However this is not the kind of weight we are interested in right now. We therefore use another weighting scheme on the edges. A scheme that accounts for the scores we have on nodes at this stage. Thus, for each edge $e \in Edge(\mathcal{G}_\mathcal{C})$ a weight is given by taking the inverse of the interestingness score of the destination node on e. This is a process which is handled in line 8 during the execution of the "dijkstraShortestPath" method. This method finds the shortest path between e and n, within the specified radius, *rad* on the given candidate sub-graph $\mathcal{G}_C$.

An output sub-graph $\mathcal{G}_\mathcal{O}$ weighs the sum of interestingness score of its nodes $n \in \mathcal{V}(\mathcal{G}_\mathcal{O}) \setminus \mathcal{E}$.

Algorithm 4 Output sub-graph extraction

Require: $\mathcal{G}_{\mathcal{C}}$ candidate sub-graph
Require: $\mathcal{E}$ entity nodes
Require: rad Radius
Ensure: $\mathcal{G}_{\mathcal{O}}$ output sub-graph

```
1: G_O ← NULL;
2: procedure EXTRACT(G_C, E, rad)
3:     Ω ← sort(V(G_C));
4:     while | V(G_O) |< b do
5:                                  ▷ pick(Ω) selects a highest scoring node from Ω
6:         n ← pick(Ω);
7:         for e ∈ E do
8:             path ← dijkstraShortestPath(e, n, G_C, rad);
9:             build(G_O, path);
10:        end for
11:        remove(Ω, n);
12:    end while
13:    return G_O;
14: end procedure
```

It is obvious from the justification of Integrated Iteration why we need to pick from nodes sorted by $I_{\mathcal{E}}$. By doing these we will be able to extract interesting relationships between any number of entity nodes.

4.5 Ranking

We have ranked output sub-graphs of a given candidate sub-graph by simply taking the sum of the interestingness of non entity nodes on each output sub-graph as explained in the previous section. Every time extraction of an output sub-graph is finished, that is when the sub-graph reached the budget constraint, another sub-graph construction commences. This will leave us with a set of output sub-graphs that needs to be ranked according to their weight. This is merely a sorting operation based on the weights of the output sub-graphs, which are obtained by summing the nodes interestingness ($\mathcal{I}_{\mathcal{E}}$) value.

Nevertheless, the above ranking is only for output sub-graphs generated from a single candidate sub-graph; that is, its local ranking. Therefore we need a way

to globally rank output sub-graphs obtained from all the candidate sub-graphs.

Though it is simple, the global ranking strategy used in this work proved to be good. This is corroborated by the quality of results which are presented in the experiments and results chapter, Chapter 5.

The global ranking works by respecting the ranking during the disambiguation phase. Remember that in Section 4.2.2 a ranked list of candidate entities are obtained from the top-τ semantic paths. We assume that this ranking should be taken into account.

Therefore the global ranking is done as follows:

1. Pick the candidate sub-graph generated for the entity tie in the first rank of the top-τ ranking.

2. Take the first $\lceil \frac{k}{\tau} \rceil$ output sub-graphs of the local ranking of the candidate sub-graph under consideration

3. For the remaining $k - (\lceil \frac{k}{\tau} \rceil)$ positions, take the first $\lfloor \frac{k - (\lceil \frac{k}{\tau} \rceil)}{\tau - 1} \rfloor$ output sub-graphs of the candidate sub-graphs generated for entity ties ranking from $2 - \tau$.

Chapter 5

Experiments and Results

This section provides the experiments carried out and the evaluation results of the approach followed in this work. The main goal is to show the performance of the model, in terms of quality and efficiency, according to different experimental settings.

5.1 Experimental Setting

In order to carry out the experiments and evaluate the performance of the model, a set of ground truth relationships (sub-graphs) for all the queries are used. These are manually extracted sub-graphs of the knowledge base. Thus each query will have a pre-manually-extracted sub-graph, where the the produced results of the model should adhere to.

There are parameters which we need to take care of during this experiment. They are:

- Candidate referent entity pruning threshold (α).
- τ, for top-τ candidate entities.
- k, the number of final out sub-graphs

In the following we present the data set used in the experiments and the evaluation metrics.

5.1.1 Data set

- *Query:* A set of selected queries, totally 80, which are taken from [12] [1] (50 queries) and AOL search engine query log (30 queries) are used. Both sets are fairly heterogeneous.

- *Knowledge-base:* Freebase [2] is the selected knowledge-base. It is preferred because of it's concept (entities) and fact (relationship) coverage. By the time that this experiment was being carried out, Freebase has 39,898,859 entities and 1,825,718 ,199 facts. This makes it the largest knowledge base in terms of concept space. Which is appropriate for this work.

5.1.2 Evaluation Measures

For systems that produce ranked results, Mean reciprocal rank (MRR) is one of the evaluation techniques which is commonly used in information retrieval. However, MRR is well suited when there should be one relevant answer in some position out of the top-K results. But for this work case, there might be more than one relevant answers. A suitable evaluation technique for such cases is then either Mean Average Precision (MAP) or Normalized Discounted cumulative gain (NDCG).

Even though we have these competing metrics for the evaluation of the quality of the results, due to the following observed facts, during the experiment, which are,

extracted output sub-graphs may

- miss relationships which are found on the ground truth.

- have additional relationships which are not found on the ground truth.

it is hardly possible to give binary judgment on the results (100% correct or 100% wrong). They should be quantified in some degree of uncertainty. This makes NDCG, which is based on graded judgments instead of binary judgments [25], an ideal choice. And hence the quality of results is evaluated according to NDCG.

From top-k ranked results, a position is graded according to Table 5.1.

[1] Queries used in this work

Table 5.1: Results quality grading scheme

$\frac{MissingEdges}{TotalEdges}$	Number of Added Edges	Grade	Gain
0	≤ 3	Perfect	10
	(3, 6]		9
	>6		8
$(0, \frac{1}{3}]$	≤ 3	Excellent	7
	(3, 6]		6
	>6		5
$(\frac{1}{3}, \frac{2}{3}]$	≤ 3	Good	4
	(3, 6]		3
	>6		2
$(\frac{2}{3}, 1)$	Any	Fair	1
1	Any	Bad	0

Once the gain is obtained, for a given query q, following [25], the $NDCG$ is calculated as

$$NDCG_q = \frac{DCG_q}{IDCG_q}$$

Where

$$DCG_q = \sum_{j=1}^{k} \frac{2^{gain(q,j)-1}}{log_2(1+j)}$$

gain(q,j) is the gain obtained for result j of query q, and it is normalized by the ideal discounted gain ($IDCG_q$) to give the $NDCG_q$. Intuitively the $IDCG_q$ is obtained by taking the ideal ordering, in decreasing order, of results gain for the particular query.

Let $gain'$ be a version of the gain function which preserves the ordering of the grading by the *gain* function. Then

$$IDCG_q = \sum_{j=1}^{k} \frac{2^{gain'(q,j)-1}}{log_2(1+j)}$$

Suppose the grading for a particular query be:

$$gain = 9, 4, 0, 7, 3, 0, 6, 1, 0, 0$$

then

$$gain' = 9, 7, 6, 4, 3, 1, 0, 0, 0, 0$$

For a set of queries $\mathcal{Q}$, the final $NDCG$ will be given by the following equation:

$$NDCG = \frac{1}{|\mathcal{Q}|} \sum_{q=1}^{|\mathcal{Q}|} NDCG_q$$

5.2 Time Performance evaluation

The performance of our model in terms of time is evaluated for each sub component of the model and hence for the complete model. Since the time performance is significantly affected by the number of entity nodes and α, the evaluation comprises combination of different values for this two parameters. In addition this evaluation is performed both on a disk resident and memory based graph.

5.3 Results

This section will present the results of the experiments carried out for this work. It shows how sensitive the model is to parameters, and also shows effectiveness and efficiency results.

As shown in Figure 5.1, the quality of the result is not significantly affected by the parameters. This proves that this model is stable.

The efficiency of the model for both disk resident and memory based graphs is also presented in Figure 5.2. For both experiments the analysis is made for every sub-component of the model. This is to show that which operation is the most expensive one in terms of time.

In addition, Figure 5.3 gives efficiency analysis for different values of τ and α. Still there is no great deal of increase in time because of the change in τ. The only thing affecting the time is alpha. This is natural, because as the values of α increased, the model is forced to take only very similar (string similarity) candidate entities.

From the efficiency experimental results, it is obvious to observe that the expensive computation is incurred during the disambiguation process. This is

due to the fact that we need to compute semantic relatedness for every tie of candidate entities of different mentions. This means, as the number of candidate entities grow so is the computational cost. Therefore, the parameter α, a syntactic similarity (on CR) threshold, plays an essential role in deciding the number of candidate entities. Those candidate entities which fall below a certain threshold are pruned away.

One problem with pruning is the case for name variants of entities. There are different ways of addressing a certain entity in the world. This will be a serious problem when our consideration is only syntactic similarity. For example, if a given query contains an entity mention “USA", one of the matching, if not the correct, candidate entities “United States of America" might fall bellow the specified threshold. Apparently because of a very low syntactic similarity. Therefore, in order to cope with these kinds of situations, we have considered set of aliases for each of the candidate entities of each entity mention. A study [27] has shown that aliases are very important for entity disambiguation when there is no context; and it is a very important input in this task. Let $\mathcal{AL}(e)$ be the set of aliases for an entity $e \in \mathcal{E}(e_m)$; the syntactic similarity between the entity mention e_m and entity e is given by taking the max syntactic similarity between the entity mention and each of the aliases of the candidate entity or the name of the candidate entity itself. It is formally given as:

$$CR(e_m, e) \leftarrow \max synt(e_m, items(e))$$

where $items(e) = \mathcal{AL}(e) \cup e$ and $synt$ is an equivalent computation as Equation 4.1.

In practice the size of a candidate sub-graph is relatively small. Particularly after pruning non-entity degree one nodes. Therefore, the computation during the two sub components “CSG", and “OSE" of the whole system is relatively efficient, below 5 seconds for all kinds of parameter settings.

Figure 5.1: Quality evaluation for $\tau = 1, 2, 3,\ \alpha =$ 0.5,0.6,0.7,0.8,0.9 and fixed number of entities = 3.

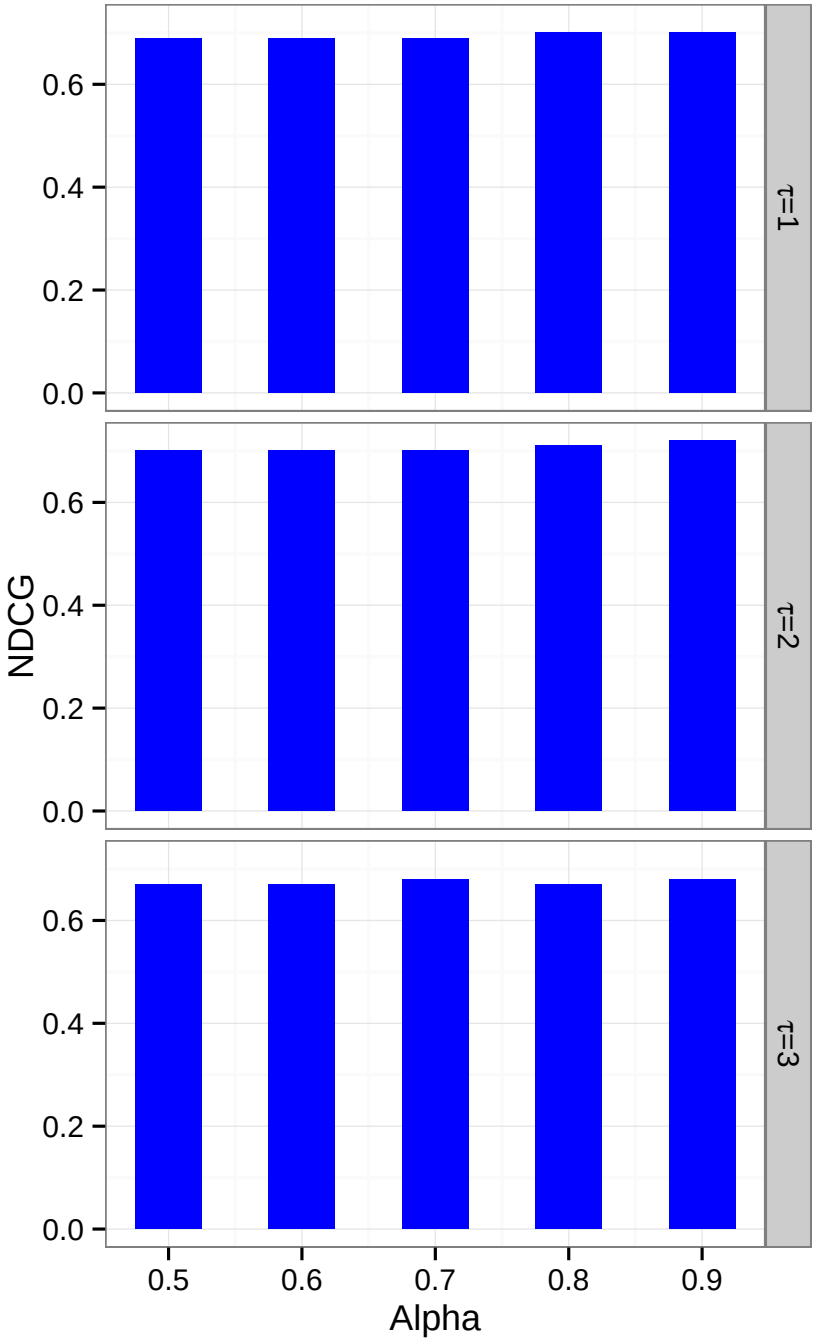


Figure 5.2: Time performance evaluation for fixed $\alpha = 0.7$ and $\tau = 2$.

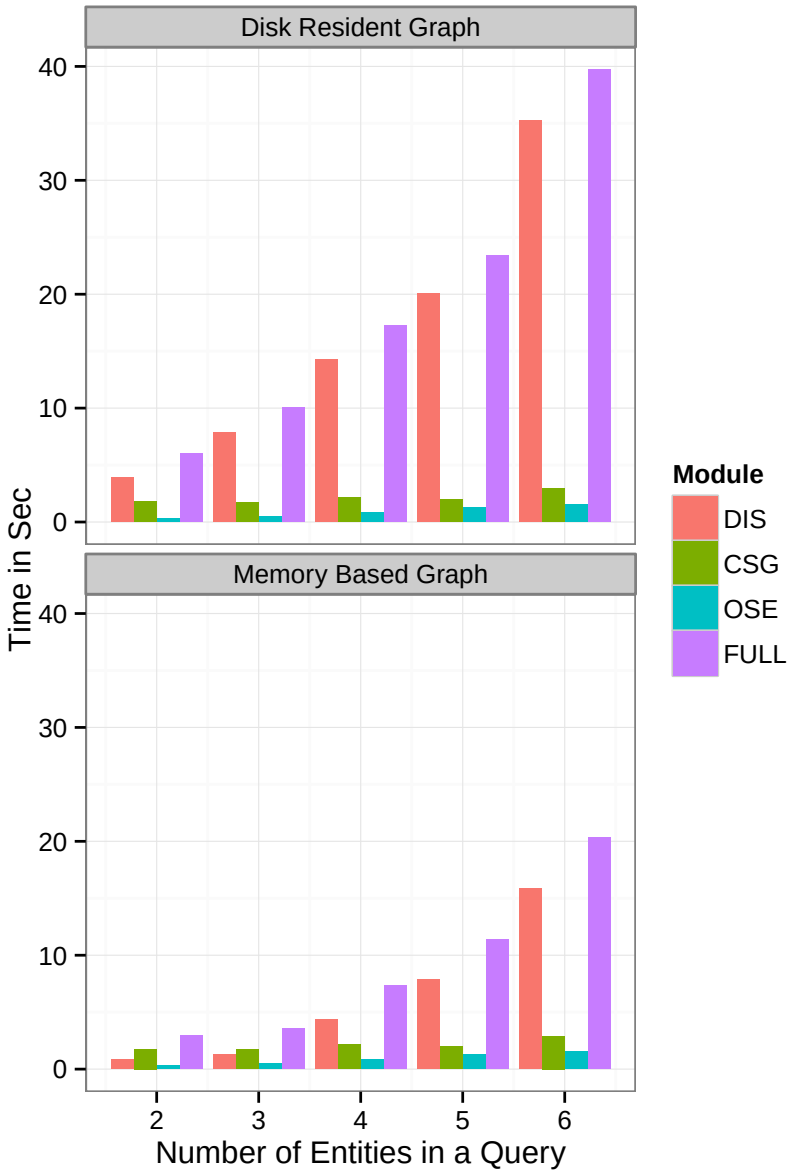
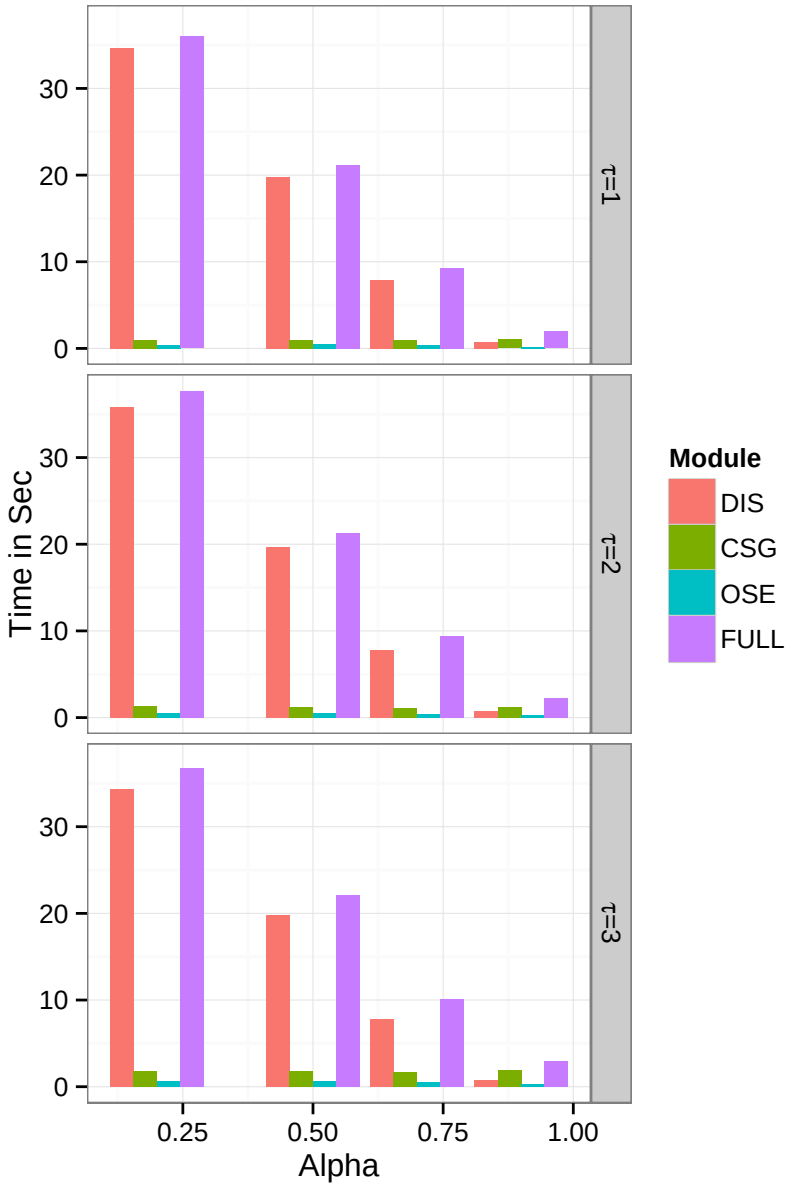


Figure 5.3: Time performance evaluation for τ=1,2,3 , α = 0.2,0.5,0.7,0.9 and fixed number of entities = 3.



Chapter 6

Conclusion and Future Work

In these work a model is proposed to map keyword queries to graph queries. It is obvious that keywords are convenient for users to formulate their queries and forward them to machines. This is because keywords

- are based on natural languages, where users are familiar with
- can easily lead to more advanced searches on different topics by merely starting from a very narrow clue

Even though keywords are preferred by users, they are usually ambiguous and over specified. On top of that, it is different when trying to address keywords simply for a document retrieval and interpreting or understanding them. The fact that they are over specified makes it harder to tell the intention of keyword queries, or what the user has in mind. Thus given these ambiguous keywords, in this work, an extended approach from an existing work is implemented to disambiguate them. This is a graph based approach that uses the so called "*referent graph*". In order to anticipate the intention of the queries and map the keywords to clearly known entities the syntactic and semantic information encoded in this graph are exploited well.

Further more, once keywords are disambiguated techniques which analyze the relationship between the disambiguated entities are also demonstrated. These phases are also based on graph, which is a snippet of the knowledge base. By extracting a local sub-graph from the knowledge-base, which strongly interconnects

the disambiguated entities, we analyzed the relationship between them and give top-k output sub-graphs as a final result.

The performance, in terms of quality and time, of the model is also presented for different setting of the model's parameters. As it is described in the experiments and results section, the model has a good performance. Though the time performance is not extremely satisfactory, still it is a good performance. The drawback in time performance is most probably due to the case that a graph-based approach is used to tackle the problem.

However, the performance in terms of quality is satisfactory. Most state-of-the-art works in this area reports very close records.

Doing further study on how we can improve the time performance and even achieving greater quality of results are goals for a future work. In addition doing a comparative work with state-of-the-art studies is also plan for future work.

APPENDIX A

Sample output sub-graphs of the model

In the following example queries, represented as n-gram keywords, entity mentions are separated by '−', and relationship mentions are put after a ';'.

Example 1: $Q = Max\ Born - Enrico\ Fermi - Paul\ Dirac$

Example 2: $Q = Clue\ Dungeons\ \&\ Dragons - Cluedo - Dungeons\ \&\ Dragons - Gary\ Gygax$

Example 3: $Q = Clint\ Eastwood - Helen\ Hunt - Edie\ Falco$

Example 4: $Q = Google - YouTube; Acquisition$

Figure 1: Example 1: ground truth query graph

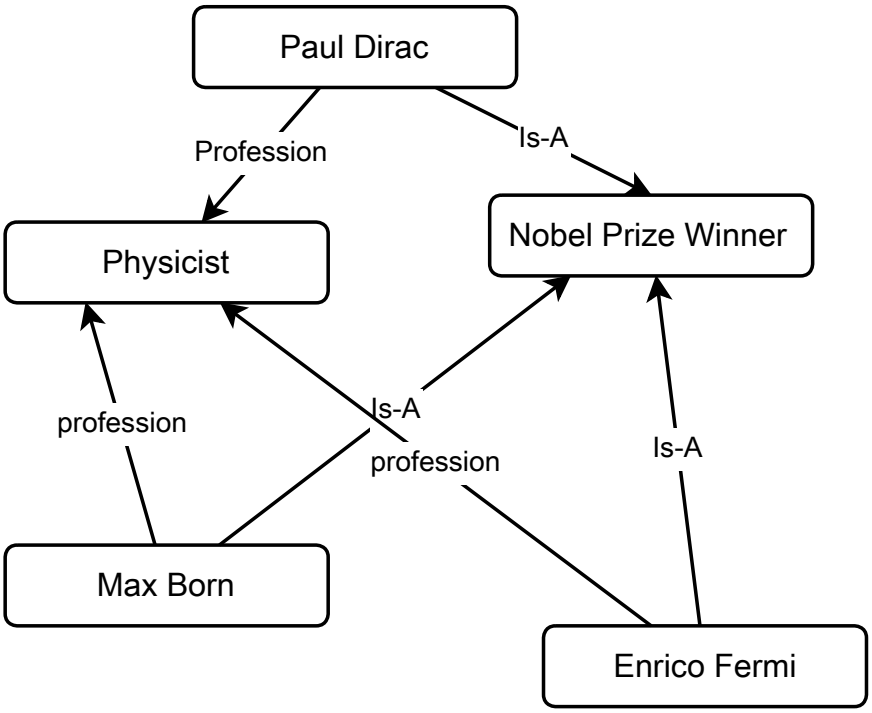


Figure 2: Example 1: output sub-graph, rank 1^{st}

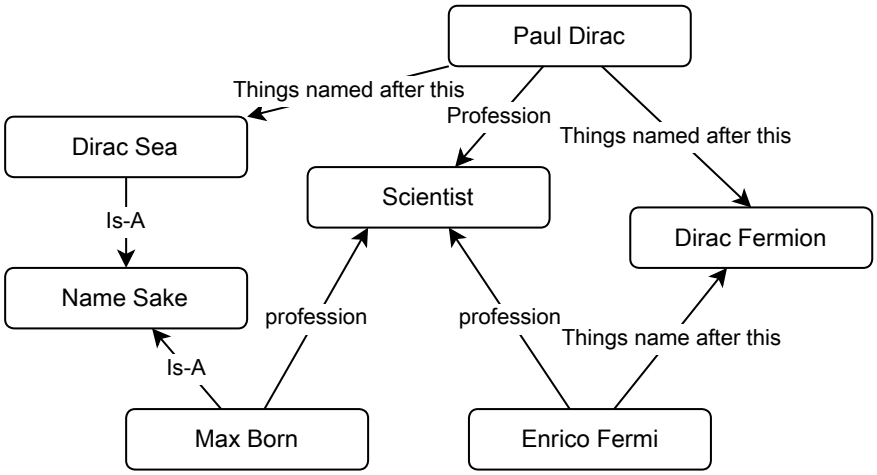


Figure 3: Example 1: output sub-graph, rank 2^{nd}

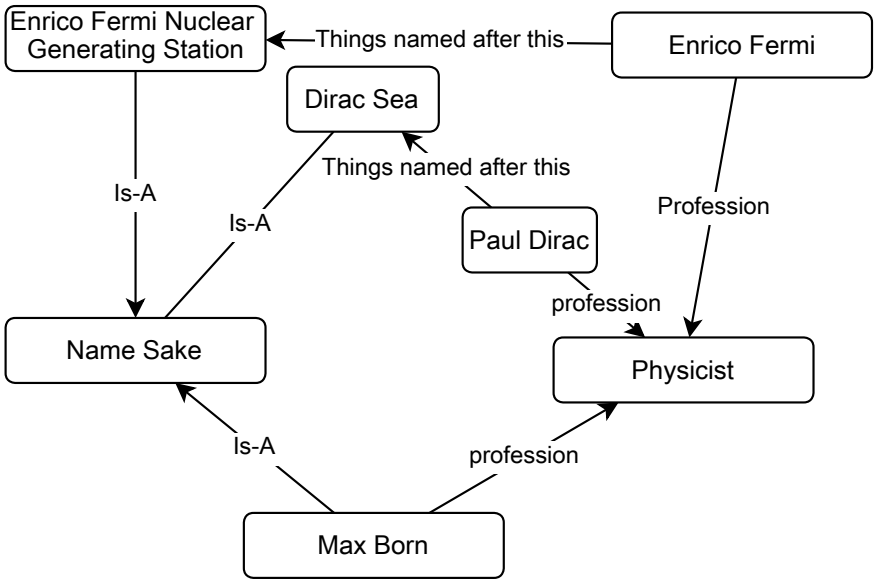


Figure 4: Example 1: output sub-graph, rank 3^{rd}

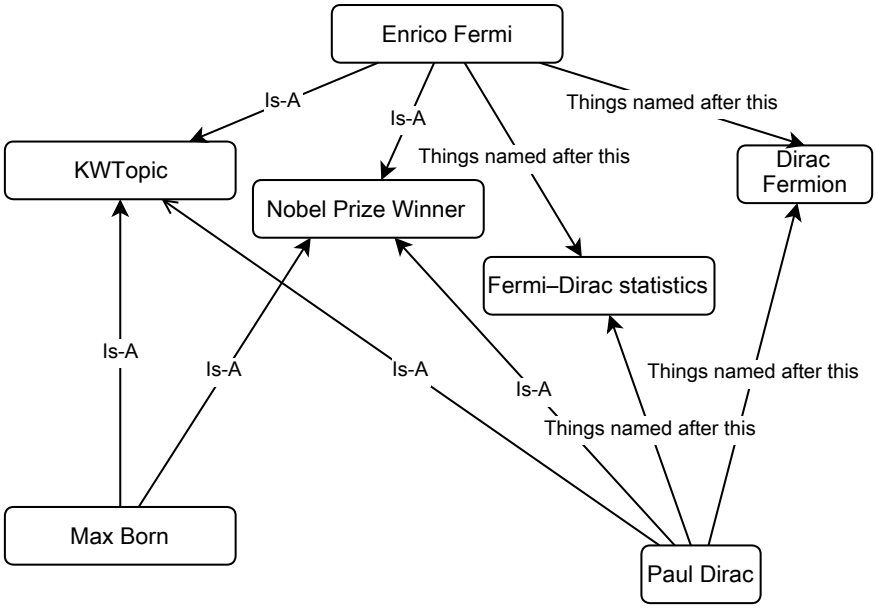


Figure 5: Example 2: ground truth query graph

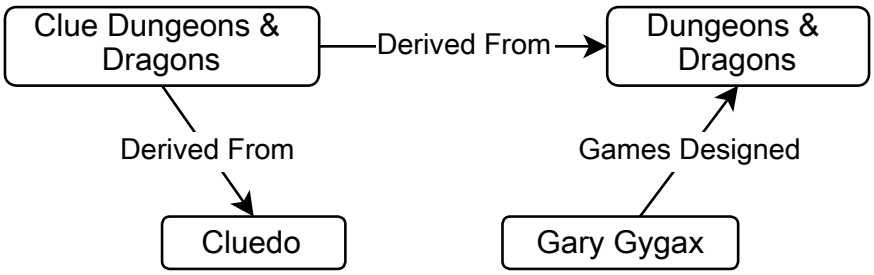


Figure 6: Example 2: output sub-graph, rank 1^{st}

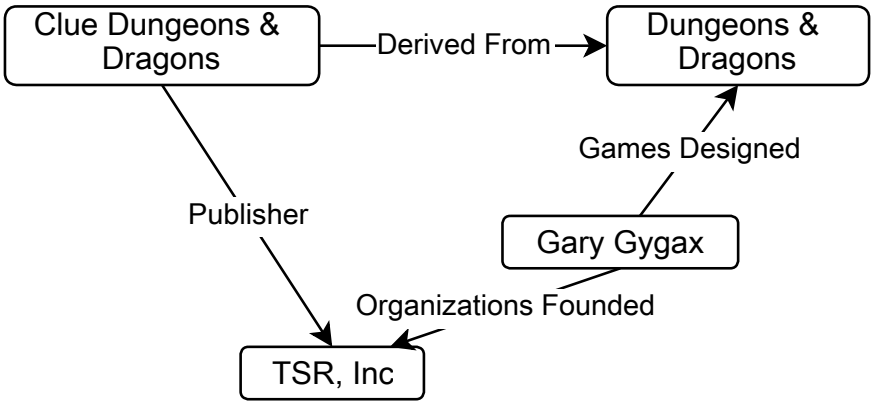


Figure 7: Example 2: output sub-graph, rank 2^{nd}

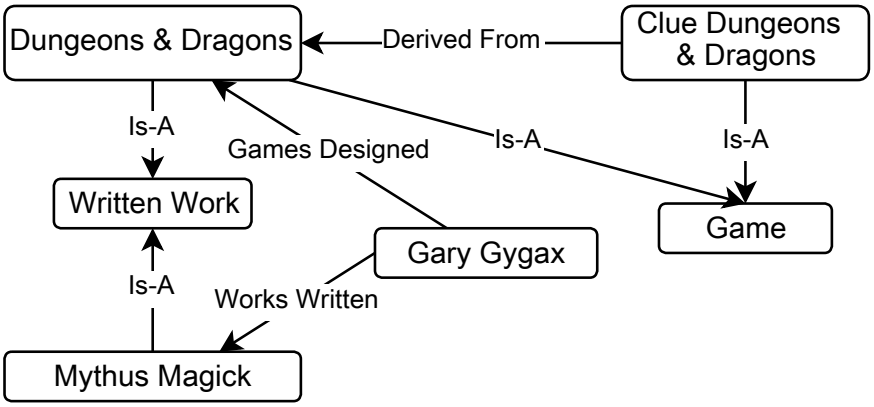


Figure 8: Example 2: output sub-graph, rank 3^{rd}

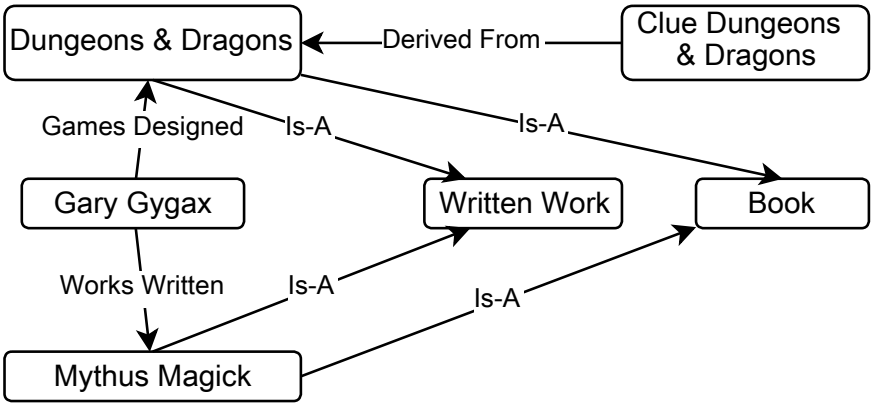


Figure 9: Example 3: ground truth query graph

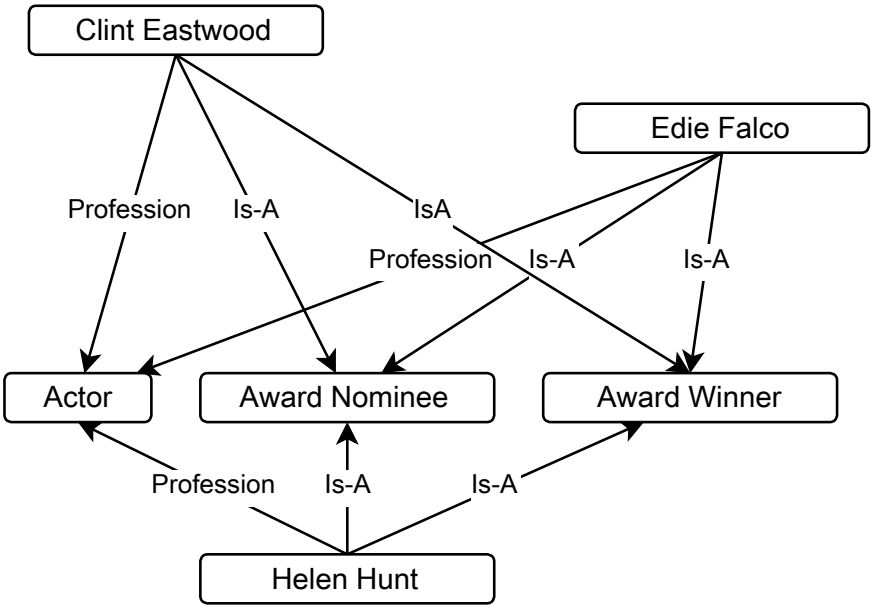


Figure 10: Example 3: output sub-graph, rank 1^{st}

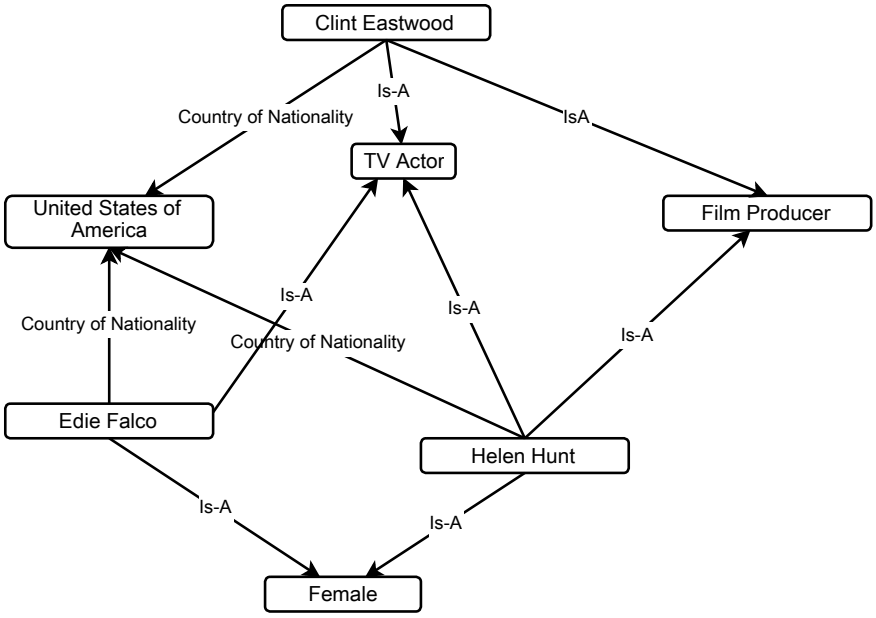


Figure 11: Example 3: output sub-graph, rank 2^{nd}

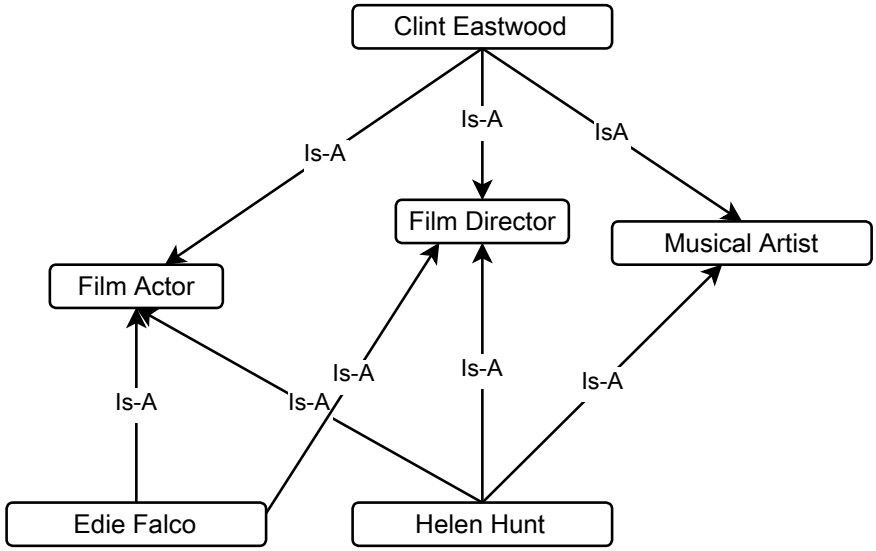


Figure 12: Example 3: output sub-graph, rank 3^{rd}

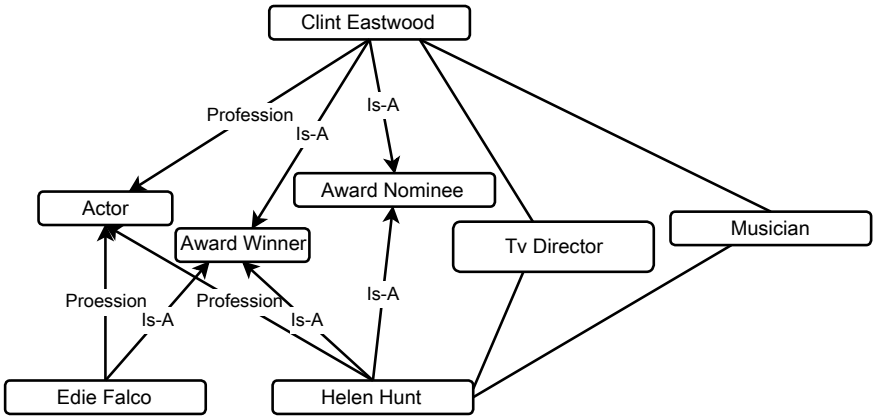


Figure 13: Example 4: ground truth query graph

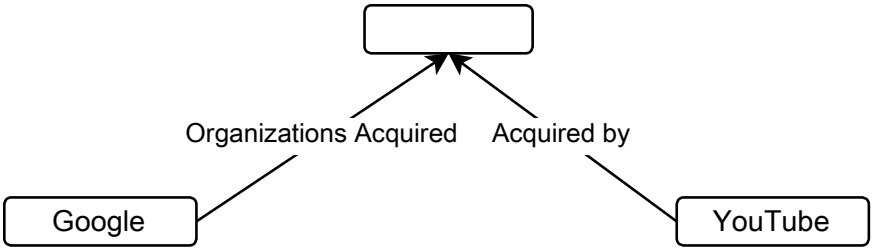


Figure 14: Example 4: output sub-graph, rank 1^{st}

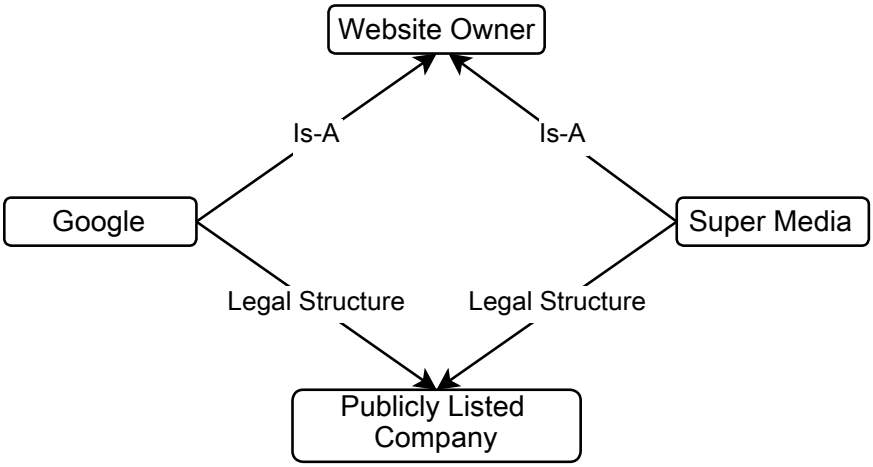


Figure 15: Example 4: output sub-graph, rank 2^{nd}

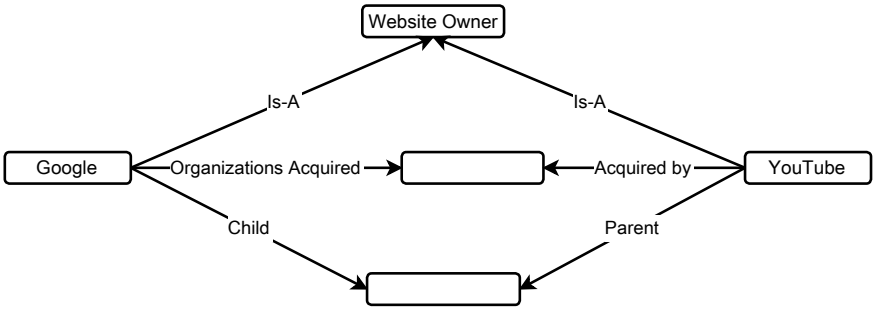
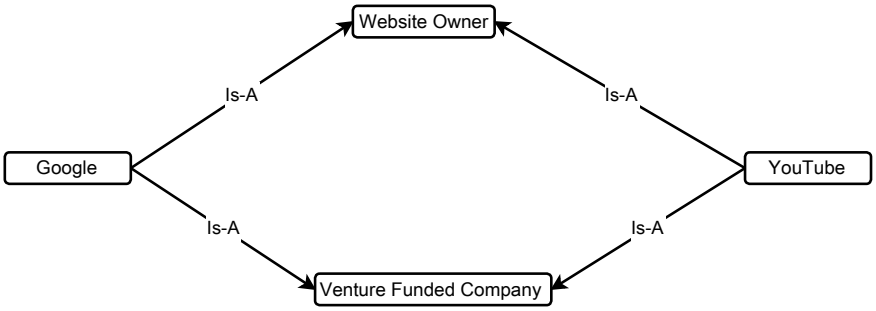


Figure 16: Example 4: output sub-graph, rank 3^{rd}



APPENDIX B

Queries used for the experiment

A. MING Queries

1. ADAM SMITH - GEORG WILHELM FRIEDRICH HEGEL
2. ADAM SMITH - JOHANN GOTTLIEB FICHTE - KARL WILHELM FRIEDRICH SCHLEGEL
3. ALBERT EINSTEIN - EDMUND HUSSERL
4. ALBERT EINSTEIN - FRIEDRICH NIETZSCHE
5. ALBERT EINSTEIN - WOLFGANG PAULI
6. ARTHUR SCHOPENHAUER - KARL MARX
7. ARTHUR SCHOPENHAUER - MORITZ SCHLICK - LUDWIG WITTGENSTEIN
8. BERTRAND RUSSELL - ALBERT EINSTEIN
9. CLINT EASTWOOD - HELEN HUNT - EDIE FALCO
10. DEMI MOORE - ASHTON KUTCHER - BRUCE WILLIS
11. EDWIN HUBBLE - ALBERT EINSTEIN
12. ERNEST RUTHERFORD - JOHANNES KEPLER

13. ERNST MACH - EDMUND HUSSERL - ADAM SMITH
14. GEORGE CLOONEY - LIAM NEESON - JAKE GYLLENHAAL
15. GEORG WILHELM FRIEDRICH HEGEL - HEINRICH HERTZ
16. GINA GERSHON - MICHAEL DOUGLAS - BRITTANY MURPHY
17. GOTTLOB FREGE - BERNARD BOLZANO
18. HARRISON FORD - ROBERT REDFORD - SALLY FIELD
19. HIDEKI YUKAWA - MAX PLANCK
20. ISAAC NEWTON - EDMOND HALLEY
21. ISAAC NEWTON - JAMES CLERK MAXWELL - WERNER HEISENBERG
22. JAMES CLERK MAXWELL - HIDEKI YUKAWA
23. JEANNE TRIPPLEHORN - JENNIFER ANISTON - DIANE LANE
24. JESSICA ALBA - LEONARDO DICAPRIO - BILLY CRYSTAL
25. JODIE FOSTER - TERI HATCHER - CHRISTINA RICCI
26. JOHANN AUGUSTUS EBERHARD - FRIEDRICH NIETZSCHE
27. JOHANN GOTTFRIED HERDER - PLATO - GOTTFRIED LEIBNIZ
28. KARL MARX - JEAN-PAUL SARTRE - LUDWIG WITTGENSTEIN
29. KEVIN SPACEY - HALLE BERRY - JULIA ROBERTS
30. LIV TYLER - DENNIS QUAID - TERI HATCHER
31. LUDWIG BOLTZMANN - RICHARD FEYNMAN
32. MAX HORKHEIMER - ARTHUR SCHOPENHAUER - HEINRICH HERTZ
33. MAX HORKHEIMER - BLAISE PASCAL - BERNARD BOLZANO

34. MAX PLANCK - JAMES CLERK MAXWELL - NIELS BOHR
35. MAX PLANCK - WERNER HEISENBERG - ENRICO FERMI
36. MAX WEBER - GEORG WILHELM FRIEDRICH HEGEL - ERNST MACH
37. MICHAEL DOUGLAS - BILLY BOB THORNTON - KIM DELANEY
38. NIELS BOHR - ERNEST RUTHERFORD - MAX BORN
39. NIELS BOHR - MICHAEL FARADAY - MAX BORN
40. PAM GRIER - MATT DAMON - SHARON STONE
41. PAUL DIRAC - ENRICO FERMI - MAX BORN
42. PLATO - BLAISE PASCAL - GOTTLOB FREGE
43. PLATO - FRIEDRICH NIETZSCHE - BERTRAND RUSSELL
44. RUDOLF CARNAP - THOMAS ABBT - MAX HORKHEIMER
45. SANDRA BULLOCK - JENNIFER ANISTON - KEVIN SPACEY
46. SARAH MICHELLE GELLAR - SALMA HAYEK - VIGGO MORTENSEN
47. SIGOURNEY WEAVER - WINONA RYDER - MICHAEL KEATON
48. STEPHEN HAWKING - JOHANNES KEPLER
49. TOM SIZEMORE - AL PACINO - JENNIFER GARNER
50. UMA THURMAN - JAKE GYLLENHAAL - JENNIFER GARNER

B. AOL Queries

1. ALICE WALKER - EVERYDAY USE
2. BILL COSBY - THE COSBY SHOW

3. STEPHENIE MEYER - TWILIGHT
4. ANSON WILLIAMS - HAPPY DAYS - POTSIE
5. GOOGLE - YOUTUBE; ACQUISITION
6. AN OFFICER AND A GENTLEMAN - AWARD WINNER
7. AN OFFICER AND A GENTLEMAN - LOUIS GOSSETT, JR
8. BEETHOVEN - IMMORTAL BELOVED
9. GOO GOO DOLLS - BLACK BALLOON
10. BLACKBOARD OF MY HEART - HANK THOMPSON
11. CHICAGO BULLS - MICHAEL JORDAN
12. CRIMINAL MINDS - SHEMAR MOORE
13. DAN BROWN - THE DA VINCI CODE - TOM HANKS
14. DREW BARRYMORE - PRODUCER
15. DWYANE WADE - GARY PAYTON
16. GEORGE BUSH - LAURA BUSH
17. GRETCHEN MOL - THE NOTORIOUS BETTIE PAGE
18. YOUR MAN - JOSH TURNER
19. YOU'VE GOT A WAY - SHANIA TWAIN; BY
20. CLUE DUNGEONS & DRAGONS - CLUEDO - DUNGEONS & DRAGONS - GARY GYGAX
21. SECRETARY OF STATE - WILLIAM F. GALVIN
22. TIM BERNERS-LEE - WORLD WIDE WEB - MACWWW - WORLD WIDE WEB CONSORTIUM

23. SANTA CLARA - YAHOO! - DEL.ICIO.US
24. TABLET COMPUTER - IPAD - APPLE INC.
25. DAVID STEWARD - WORLD WIDE TECHNOLOGY; CHAIRMAN
26. MICHAEL J FOX - PARKINSON;FOUNDATION
27. X-MEN - FILM - ROGUE - WOLVERINE
28. CAMILLA BELLE - ACTOR
29. ROMEO AND JULIET - WILLIAM SHAKESPEARE; BY
30. MILEY CYRUS - TISH CYRUS - BILLY RAY CYRUS - SINGER - SONG-WRITER - ACTOR

References

[1] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference* (Berlin, Heidelberg, 2007), ISWC'07/ASWC'07, Springer-Verlag, pp. 722–735. 1

[2] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (2008), ACM, pp. 1247–1250. 1, 40

[3] Faloutsos, C., McCurley, K. S., and Tomkins, A. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2004), KDD '04, ACM, pp. 118–127. 2

[4] Fang, L., Sarma, A. D., Yu, C., and Bohannon, P. Rex: explaining relationships between entity pairs. *Proc. VLDB Endow. 5*, 3 (Nov. 2011), 241–252. 12, 30

[5] Gabrilovich, E., and Markovitch, S. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence* (San Francisco, CA, USA, 2007), IJCAI'07, Morgan Kaufmann Publishers Inc., pp. 1606–1611. 12

[6] Guo, J., Xu, G., Cheng, X., and Li, H. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 267–274. 14

[7] Han, X., and Sun, L. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (Stroudsburg, PA, USA, 2011), HLT '11, Association for Computational Linguistics, pp. 945–954. 2, 9

[8] Han, X., Sun, L., and Zhao, J. 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* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 765–774. 2, 8, 14, 17, 19

[9] Han, X., Sun, L., and Zhao, J. 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* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 765–774. 20, 21

[10] He, H., Wang, H., Yang, J., and Yu, P. S. Blinks: ranked keyword searches on graphs. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), SIGMOD '07, ACM, pp. 305–316. 16, 26

[11] Hu, J., Wang, G., Lochovsky, F., Sun, J.-t., and Chen, Z. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web* (New York, NY, USA, 2009), WWW '09, ACM, pp. 471–480. 11

[12] Kasneci, G., Elbassuoni, S., and Weikum, G. Ming: mining informative entity relationship subgraphs. In *Proceedings of the 18th ACM conference on Information and knowledge management* (New York, NY, USA, 2009), CIKM '09, ACM, pp. 1653–1656. 2, 26, 27, 33, 40

[13] Kasneci, G., Ramanath, M., Sozio, M., Suchanek, F. M., and Weikum, G. Star: Steiner-tree approximation in relationship graphs. In *Proceedings of the 2009 IEEE International Conference on Data Engineering* (Washington, DC, USA, 2009), ICDE '09, IEEE Computer Society, pp. 868–879. 2

[14] Khan, A., Wu, Y., Aggarwal, C. C., and Yan, X. Nema: fast graph search with label similarity. In *Proceedings of the 39th international conference on Very Large Data Bases* (2013), PVLDB'13, VLDB Endowment, pp. 181–192. 2

[15] Milne, D., and Witten, I. H. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence* (2008). 21

[16] Pound, J., Hudek, A. K., Ilyas, I. F., and Weddell, G. Interpreting keyword queries over web knowledge bases. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (New York, NY, USA, 2012), CIKM '12, ACM, pp. 305–314. 10

[17] Pound, J., Ilyas, I. F., and Weddell, G. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (New York, NY, USA, 2010), SIGMOD '10, ACM, pp. 423–434. 2, 11

[18] Ramakrishnan, C., Milnor, W. H., Perry, M., and Sheth, A. P. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explor. Newsl. 7*, 2 (Dec. 2005), 56–63. 2

[19] Shen, W., Wang, J., Luo, P., and Wang, M. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web* (New York, NY, USA, 2012), WWW '12, ACM, pp. 449–458. 2, 9

[20] Suchanek, F. M., Kasneci, G., and Weikum, G. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. 1, 10

[21] Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A large ontology from wikipedia and wordnet. *Web Semant. 6*, 3 (Sept. 2008), 203–217. 1, 10

[22] Tong, H., and Faloutsos, C. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2006), KDD '06, ACM, pp. 404–413. 2, 33, 34

[23] Tong, H., Faloutsos, C., and Pan, J.-Y. Fast random walk with restart and its applications. In *Proceedings of the Sixth International Conference on Data Mining* (Washington, DC, USA, 2006), ICDM '06, IEEE Computer Society, pp. 613–622. 33, 34

[24] Vivaldi, J., and RodrÃıguez, H. Finding domain terms using wikipedia. 14

[25] Wang, Y., Wang, L., Li, Y., He, D., Liu, T.-Y., and Chen, W. A theoretical analysis of ndcg type ranking measures. *CoRR abs/1304.6480* (2013). 40, 41

[26] Wu, W., Li, H., Wang, H., and Zhu, K. Q. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2012), SIGMOD '12, ACM, pp. 481–492. 1

[27] Zheng, Z., Si, X., Li, F., Chang, E., and Zhu, X. Entity disambiguation with freebase. In *The 2012 IEEE/WIC/ACM International Conference on Web Intelligence (WI'2012)* (2012). 2, 43