INSA Lyon
Universitat Passau

Master Thesis

Image Annotation Network

Author:
Mael Ogier

Supervisors:
Dr. David Coquil
Dr. Elöd Egyed-Zsigmond

This thesis submitted in fulfilment of the requirements for the degree of Master of Science

in the

Informatique - Information und Kommunikation (IFIK)
Double Master Program

September 2015

Declaration of Authorship

"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web–the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines."

Sir Tim Berners-Lee

Acknowledgements

This master thesis was done in the context of the double master degree Informatique - Information und Kommunikation (IFIK), which brings together two Master programs : a degree in computer engineering at the National Institute of Applied Sciences in Lyon (INSA Lyon) and a Master in Informatik (Schwerpunkt : Information und Kommunikationssysteme) at the University of Passau.

First, I would like to thanks the initiators of the double Master IFIK for their valuable efforts and commitment to the german-french collaboration : Prof. Dr. Harald Kosch for the german side and Prof. Dr. Lionel Brunie for the french one.

An unmeasurable amout of thanks goes to my advisors, Dr. David Coquil and Dr. Elöd Egyed-Zsigmond for their precious help and advices during this thesis preparation.

I would also thanks my colleagues and friends for their support as well as my five german roommates who made my time in Passau such enjoyable.

Finally, I thanks the Dropkick Murphys for their energizing music and the worldwide coffee producers for this magical beverage they provide us.

INSA LYON
UNIVERSITAT PASSAU

Abstract

IFIK
Double Master Program

Master of Science

Image Annotation Network

by Mael OGIER

Image is a popular medium nowadays : it is easy to capture, can be really light on your electronic device and speaks to everyone without distinction of language. Lot of companies use image data every day and so need a good way to retrieve them. For a human-being, the more intuitive way to describe what he's looking for is by the use of his own words, not a distribution of colors or any low-level feature. Therefore, it is interesting to add annotations (also called tags) to images in order to describe them in an natural language way. In this work, we propose a semantic enrichment prototype based on the semantic web and graph models. Five experiments are presented and their results are discussed. Eventually we expose some future work which should be interesting.

Contents

List of Figures

List of Tables

Abbreviations

BFS	**B**readth-**F**irst **S**earch
DN	**D**irect **N**eighbors experiment
LCS	**L**east **C**ommon **S**ubsumer
RDF	**R**esource **D**escription **F**ramework
SL	**S**ub**L**ists experiment
SP	**S**hortest **P**ath
SPARQL	**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
URI	**U**niversal **R**esource **I**dentifier
WL	**W**hole**L**ist experiment
WP	**W**u-**P**almer measure
WP++	**W**u-**P**almer evolved measure
XML	e**X**tensible **M**arkup **L**anguage

Chapter 1

Introduction

1.1 Background

Image is a popular medium nowadays : it is easy to capture, can be really light on your electronic device and speaks to everyone without distinction of language.

In the all days life, people share their pictures on social networks in less than a blink of eye. In average, 70M of pictures are posted on Instagram each day and the users hit the "Like" button 2.5B times[1]. Other services like Picasa or Flickr exists but aren't as used as Instagram which is the favorite in the eyes of the teen public.
Companies also produce a lot of media data. Industry companies need their products' pictures, marketing and advertising studios use a lot of images in order to create new stuff for their client, . . . But the most consumer of media data are obviously mass media themselves : Newspapers, TV shows, news broadcasts are dealing with pictures at every moment of their day.

This huge production and consumption of images implies the need of an efficient way to store and search for the relevant one when the time comes. The best illustration to this need is to think of the nice but long moments one had with its relatives searching for the good picture of the new-born nephew in the family pictures album.
Since an image itself doesn't have a natural plain-text representation the best way to

[1]Stats from : https://instagram.com/press/

describe it is to add meta-data (data about the data) such as its date of creation, its dimensions or, and this is what this thesis is about, some tags.

There are a lot of ways if one wants to annotate pictures. We can do it manually, using our own words (like “Dad”, “Home” . . .), we can also analyze the raw picture, its pixel representation and compare some metrics (like the color histogram) to sample images in order to detect known concepts. Moreover, if the image already possesses annotations, we can enrich it semantically.

This field is so wide that it is impossible to speak about all the possibilities and technologies. In this study, we will focus on the last point and investigate the automation of the semantic enrichment. We will study the resources at our disposal and propose a solution keeping in mind the facts cited previously.

In the following section, we will present and discuss an application scenario to illustrate the motivation behind this thesis.

1.2 Motivation

NewsTV is a famous TV news channel which runs 24/7 and only speaks about the current news. It has lot of reporters worldwide, covering the important local news and sending their production to the main site in Paris, France.

The employees often need to consult older coverages in order to explain the context of the news, to make the necrology of a famous actor who recently died or to re-use common shots. Therefore, they need to query the central multimedia database management system using keywords they are familiar with like “Elections, France, 2007, José Bové”. But sometimes, their research aren’t so specific and they are looking for more generic pictures, let say “Land, Tree, Animal”.

The first kind of keywords had been tagged by the former reporter who produced the coverage but he logically didn't think to add generic terms. NewsTV needs something to do it automatically when a picture, or any media, is first added to its system with a few initial tags.

Details about which kind of technology can be used to achieve this automatic tagging will come in the following sections. To summarize, the goal of this thesis is to propose a running prototype and evaluate different methods of tagging. The questions that we will try to answer during this study are described in the following section.

1.3 Thesis Objectives

TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO

1.4 Thesis Outline

The remainder of this thesis will be organized as follows :

Chapter 2 - Semantic Web : presents the general concept of the semantic web and different semantic web resources, their structures, how to browse them and how are they used in the literature.

Chapter 3 - Measures : provides a solid background on semantic similarity and distance measures. We explore different metrics illustrating their pro/cons with examples.

Chapter 4 - Disambiguation : reviews the literature and assesses the most relevant ways to disambiguate a list of keywords which may be organize into sentences or not.

Chapter 5 - Existing Approaches : describes existing image annotation approaches as well as their architecture.

Chapter 6 - State of the Art Conclusion : summarizes the findings of the previous state of the art and opens the way to the presented contribution.

Chapter 7 - Proposed Methodology : presents the chosen methodology as well as some organizational points.

Chapter 8 - Proposed Architecture : details the technological choices by comparing them to their competitors and the chosen DBMS schema. Illustration figures will be presented.

Chapter 9 - Experiments : presents the chosen dataset, details some of the main algorithms and reviews the tests' results with the use of different evaluation methods.

Chapter 10 - Contribution Conclusion : summarize the findings of the presented research problem.

Part I

State of the Art

Chapter 2

Semantic Web

Our study is focus on semantic enrichment of an initial set of keywords which can be organized as sentences or not. It is important to first understand what is a semantic concept and how concepts are organized into ontologies.

In this section, we will present some general notions about semantic concepts and review several semantic resources, their hierarchical structures and how we access them.

2.1 Generalities

Linguistic semantics is the study of meaning that is used for understanding human expression through language. It is easy for two human-being to communicate (given that they speak the same language) and to understand what their partner say even if he's using a tricky turn of phrase. However, this task becomes way more difficult when it comes to the comprehension of the human language by a machine. How can the computer guess that "I am totally dead" means in fact "I am really tired" and that the speaker isn't actually dead ? Machines need structured resources to understand us and the Semantic Web is one of them.

The notion of "Semantic Web" has been mentioned for the first time by Berners-Lee et al in [1]. In this paper, they describe it as a Web which is readable by machines in opposite of most of Web's content which were designed for humans to read. The Semantic Web isn't a separate Web but an extension of the current one which will bring structure to the meaningful content of Web pages.

Two main technologies are used for the development of the Semantic Web : eXtensible Markup Language (short XML) and the Resource Description Framework (short RDF). XML allows everyone to create their own tags and to arbitrary structure their documents but gives no information about what this structure means. Meaning is provided by RDF which stores it in sets of triples which are composed of a subject, a predicate and an object. Those three components can be related to the subject, the verb and object of an elementary sentence. In [2], Miller presents a short introduction to the RDF standard and precises that a "Resource" can be any object which is uniquely identifiable by a Uniform Resource Identifier (URI).

The third basic component of the Semantic Web are collections of information called ontologies. An ontology is, in computer science, a document which defines the relations among concepts. Basically, Web ontologies are composed of a taxonomy, which defines classes of objects and their relations, and a set of inference rules.

In addition, the The New Oxford Dictionary of English defines the notion of "semantic concept" as : *An idea or thought that corresponds to some distinct entity or class of entities, or to its essential features, or determines the application of a term, and thus plays a part in the use of reason or language*.

Given those basic notions, we will now further detail four semantic web resources, their taxonomies and review some of their usage found in the literature.

2.2 Semantic Resources

2.2.1 DBpedia

DBpedia[1] is a project originally launched by two German universities (Berlin and Leipzig) and backed by an important community. It explores Wikipedia[2] and extract information from it which results on the creation of a multilingual, large-scale knowledge base. The extraction framework, all the available end-points as well as some facts and figures about the project are presented in [3].

DBpedia's ontology is based on classes (320 items) which form a subsumption hierarchy, the root element being owl:Thing, with a maximal depth of 5[3]. Theses classes are described by a total of 1650 different properties, forming a large set of RDF triples (580 million extracted from the English version of Wikipedia).

Even though DBpedia is now a worldwide project and provides pages in 125 languages, the English one is still the most represented. We can indeed find 4.58 million of things[4] including 1,445,000 instances of the class *Person*, 735,000 places *Place*, 251,000 *Species* . . . The number of instances described in this language is about three time larger than the second and third language (French and German).

As well as any RDF-structured dataset, DBpedia can be requesting with SPARQL (which is an recursive acronym : SPARQL Protocol and RDF Query Language) queries. SPARQL allows the user to search, add, modify or delete RDF data available on the Internet, see [4] for more details about the language.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?class
WHERE { <E> rdf:type ?class }
```

Code 2.1: SPARQL Query : Search classes

[1]http://wiki.dbpedia.org/
[2]https://en.wikipedia.org/
[3]Complete classes tree : http://mappings.dbpedia.org/server/ontology/classes/
[4]http://wiki.dbpedia.org/about/facts-figures

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?superClass
WHERE { <C> rdf:subClassOf ?superClass }
```

CODE 2.2: SPARQL Query : Search superclasses

Codes 2.1 and 2.2 present two simple and generic SPARQL search queries wich return respectively the class(es) of a given entity E and the superclass(es) of a given class C using the *type* and *subClassOf* predicates.

DBpedia also provides useful web services and HTTP endpoints. DBpedia Spotlight, which highlight DBpedia concepts in an input text is described in [5] and further details about disambiguation using this service are presented in subsection 4.2. The official DBpedia SPARQL endpoint[5] allows the user to send SPARQL queries to the online Virtuoso Triple Store by using the browser interface or by sending a HTPP request. We learn in [3] that the average amount of hits per day of this endpoint is of 2,910,410 for the 3.8 dataset version.

We find lot of papers in the literature which mention DBpedia as an asset for systems based on the Semantic Web. If some of those papers are still in the research field, like [6] which proposes a music recommendation system built on top of DBpedia, others present "real" applications which are currently in use. We can cite for instance [7] which describes how the BBC[6] uses DBpedia to backbone its publications.

2.2.2 Geonames

GeoNames[7] is a geographical database which contains over 10 million geographical names. The most documented countries[8] are the United States of America (2,203,094 names), Norway (600,008) and China (526,456). All resources are categorized into one out of nine classes and further subcategorized into one out of 645 codes[9]. Obviously,

[5]http://dbpedia.org/sparql
[6]http://www.bbc.co.uk/
[7]http://www.geonames.org/
[8]http://www.geonames.org/statistics/
[9]http://www.geonames.org/export/codes.html

the root element of Geonames' hierarchy is mother earth.

Like DBpedia, the Geonames' ontology makes possible the addition of new resources to the World Wide Web. However, Geonames distinguish the feature's Concept from the RDF document about it. In consequence, each feature possesses two representation in Geonames. See the two following URIs as example :

URI1 http://sws.geonames.org/8015555/

URI2 http://sws.geonames.org/8015555/about.rdf

The first one stands for the "Notre Dame de Fourvière" church in Lyon, France. This URI is used when one wants to refers to the church itself. The second URI is the RDF document with what Geonames knows about the church, its latitude and longitude for instance, or some nearby locations.

In order to allow the user to browse its *Tremendous set of data* (Sir T. Berners-Lee), Geonames proposes a lot of REST-based web-services. In the case of image annotation, one in particular could be useful. Given that lot of recent numeric images contain EXIF data which embed GPS information such as the longitude and the latitude (see [8] for further details about the EXIF format), one could add geographic tags to the picture with the help of the *findNearbyPlaceName* service. Here is an example of usage :
Query : findNearbyPlaceName?lat=48.566&lng=13.43&username=demo

```
<geonames>
  <geoname>
    <toponymName>Passau</toponymName>
    <name>Passau</name>
    <lat>48.5665</lat>
    <lng>13.43122</lng>
    <geonameId>2855328</geonameId>
    <countryCode>DE</countryCode>
    <countryName>Germany</countryName>
    <fcl>P</fcl>
    <fcode>PPLA3</fcode>
    <distance>0.00041</distance>
  </geoname>
</geonames>
```

Code 2.3: Geonames' XML response

Given its specific domain, we found less use-cases of Geonames in the literature than DBpedia's. Some interesting papers have however been presented, like [9] which use several semantic resources including Geonames in order to detect named entities in "Agence France Presse" (AFP) wires.

2.2.3 WordNet

WordNet[10] is a lexical database of English which has been presented for the first time in 1995 in [10]. It is hosted by the Princeton University, currently running version 3.1 but there are no current plan for a future release due to limited staffing.

Its structure is based on the concept of "synset" (synonym set), a set cognitive synonyms. WordNet distinguish among Types (common nouns, verbs. . .) and Instances (specific persons. . .). Synsets are interliked using conceptual, semantic and lexical relations. The hierarchy is built by the use of the super-subordinate relation (or hyperonymy, hyponymy in WordNet's jargon). These relations implements the two directions of the "IS-A" expression. For instance, *fruit* is **hyperonym** of *apple* and *horse* is a **hyponym** of *animal*, the root element being "entity". Other relations are also provided, like the antonymy (opposite of synonymy) or the meronymy and its opposite holonymy which implements the "IS-PART-OF" relation : *finger* is a **meronym** of *hand*. All these relations are transitive.

This resource is useful if we are searching for entities. Since the maximal depth is of the ontology is of 16, the leafs are very detailed nous (tsetse-fly, Yukon white birch, . . .) but it also contains more general concepts (vehicle, animal, . . .). WordNet contains at the moment 155,287 unique strings incuding 117,798 nouns.

It exists several ways to browse this resource. An online interface allows the user to manually query the dataset and to navigate in it through hyperlinks. For software and research purposes, the user has to download one of the released version of WordNet's

[10]https://wordnet.princeton.edu/

dataset as well as a specific library according to the code language he's using.

Due to the lot of different relations between its synsets, WordNet has mostly been used for measuring semantic distances (see [11], [12] and [13]) or to disambiguate texts (in [14], [15] and [16]).

2.2.4 ImageNet

ImageNet[11] is an image database built on the WordNet hierarchy. It has been launch in 2009 and presented in [17]. Each of WordNet's synsets is depicted in ImageNet by a set of pictures (more than 500 in average). At the moment 14,197,122 images are referenced.

The stated aim of ImageNet is to serve as an asset for pedagogical and research purposes. It has for instance been used in [18] to measure the correlation between visual and semantic similarities or in a kindergarten in Canada to provide matching exercises to the children.

Despite the fact that this resource is not as used as the previous ones, it was important to cite it in order to highlight that semantic concepts and images can be related even tough they are different kinds of medium.

[11]http://www.image-net.org/

Chapter 3

Similarity Measures

This chapter aims to review several similarity measures. The first two sections describe semantic metrics when the last one shortly present a statistical way to measure concepts' relatedness. In order to illustrate those methods, we will used the figure 3.1 which is an handcrafted ontology.

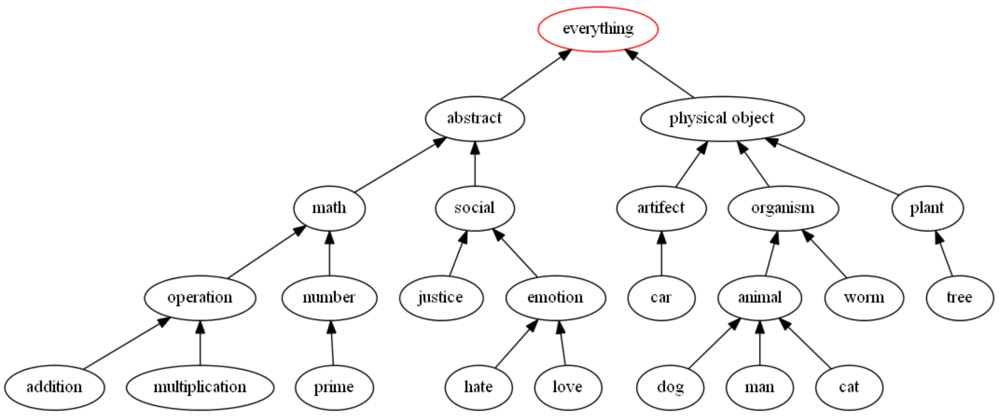

FIGURE 3.1: Sample ontology

3.1 Edge-based

3.1.1 Shortest Path

This measure is in fact a simple node-counting scheme (path). The similarity score is inversely proportional to the number of nodes along the shortest path between the concepts. The shortest possible path occurs when the two concepts are the same, in which case the length is 1. Thus, the maximum similarity value is 1.

One has to be very careful with the use of this metric because it initially doesn't take into account neither the kind of relations nor their direction. Therefore, two concepts which are hierarchically related may be as similar as two leafs. Based on figure 3.1 we can for instance see that :

$$sim_{SP}(addition, social) = sim_{SP}(addition, prime) = 1/4 = 0.25 \tag{3.1}$$

It means that, with regards to the Shortest Path metric, addition and social are as similar as addition and prime.

3.1.2 Wu-Palmer

In [19] Wu and Palmer (1994) introduce a new scaled metric in order to translate English verbs into Mandarin Chinese. This measure takes into account the Least Common Subsummer (LCS)[1] of the 2 considered concepts. The formula is :

$$sim_{WP}(c_1, c_2) = \frac{2 * depth(lcs(c_1, c_2))}{depth(c_1) + depth(c_2)} \tag{3.2}$$

This means that $0 < sim_{WP} <= 1$. The score can never be zero because the depth of the LCS is never zero since the depth of the root of a taxonomy is one. As for the shortest path, the score is one if the two input concepts are the same.
In our case, the results are the following :

$$sim_{WP}(addition, social) = 2 * 2/(5 + 3) = 4/8 = 0.5 \tag{3.3}$$

$$sim_{WP}(addition, prime) = 2 * 3/(5 + 5) = 6/10 = 0.6 \tag{3.4}$$

With *abstract* being the LCS of *addition* and *social* and *math* being the LCS of *addition* and *prime*. These results seems naively more normal than the one in equation (3.1).

[1]Sometimes called LCA : Lowest Common Ancestor

3.1.3 Evolved Wu-Palmer

In [20], Zargayouna exposes a consequence of the Wu-Palmer formula which can, according to the context of use, gives false results. Indeed, in (3.2), the similarity between two concepts is related to their distance to the LCS. Therefore, the more "general" the LCS is, the less similar the concepts (and inversely). With this configuration is it possible to observe the following result with c_2 being a child of c_1 and c_3 a brother node.

$$sim_{WP}(c_1, c_2) < sim_{WP}(c_1, c_3) \tag{3.5}$$

Which can be an issue if one want to give an advantage to the father-child relation instead of the brother-brother one. Here is an example with our sample ontology from figure 3.1 :

$$sim_{WP}(addition, math) = 2 * 3/(5 + 3) = 6/8 = 0.75 \tag{3.6}$$

$$sim_{WP}(addition, multiplication) = 2 * 4/(5 + 5) = 8/10 = 0.8 \tag{3.7}$$

In order to achieve what's mentioned above, the author create a virtual bottom node to which every leaf node is linked. Then, she integrates in the equation the maximum number of edges between this virtual node and the LCS as well as the number of edges between the LCS and each of the considered concepts.

$$sim_{WP++}(c_1, c_2) = \frac{2 * depth(lcs(c_1, c_2))}{depth(c_1) + depth(c_2) + spec(c_1, c_2)} \tag{3.8}$$

$$spec(c_1, c_2) = depth_{Bottom}(lcs(c_1, c_2)) * dist(LCS, c_1) * dist(LCS, c2)) \tag{3.9}$$

It is important to note that this modification of the Wu-Palmer formula rest on the fact that dist(LCS,c) can be null if one of the concepts is the LCS which occurs in the father-child relation. In this case, the third part of the denominator becomes null and the similarity is the same as in (3.2). In the brother-brother relation yet, this part isn't null and the similarity decrease. Let's take the same examples :

$$sim_{WP++}(addition, math) = 2 * 3/(5 + 3 + (3 * 0 * 2)) = 6/8 = 0.75 \tag{3.10}$$

$$sim_{WP++}(addition, multiplication) = 2 * 4/(5 + 5 + (2 * 1 * 1)) = 8/12 \ = 0.66 \tag{3.11}$$

The results are those expected, and we can conclude that this measure works but that the user really has to know which behavior he needs before choosing WP or WP++.

3.2 Node-based

3.2.1 Resnik

In [21] Resnik presents a new measure of semantic similarity in an "IS-A" taxonomy, based on a notion he call "information content" (IC). Its work is established on the following hypothesis: the more the information two concepts share in common, the more similar they are. In order to evaluate the amount of information shared by the considered concepts Resnik use their LCS' IC.

Given a chosen corpus, let p(c) the probability to encountering an instance of c. Note that this probability is monotonic as one moves up in the taxonomy. It means that if c_1 IS-A c_2, then $p(c_1) < p(c_2)$. For instance in our outology, $p(dog) < p(animal)$.
The Resnik's formula stands as :

$$sim_R(c_1, c_2) = log(p(LCS(c_1, c_2))) \tag{3.12}$$

We can not present results using our taxonomy because we would need a corpus to do so (like a book, composed of multiple sentences which contains words) but the original paper displays some in its Table 3.

3.2.2 Lin

Lin proposes in [22] a new measure based on Resnik's (see 3.2.1). It adds to Resnik's hypothesis the assumption that the similarity between c_1 and c_2 is also related to the differences between them; the more differences they have, the less similar they are.
Therefore, Lin modify the initial formula and adds a denominator :

$$sim_R(c_1, c_2) = \frac{log(p(LCS(c_1, c_2)))}{log(p(c_1)) + log(p(c_2))} \tag{3.13}$$

3.2.3 Jiang and Conrath

Jiang and Conrath use the same hypotheses than Lin in [23], but their formula isn't a ratio but a difference :

$$sim_R(c_1, c_2) = log(p(LCS(c_1, c_2))) - (log(p(c_1)) + log(p(c_2))) \tag{3.14}$$

3.3 Statistical similarity

It also exists statistic methods which evaluate the similarity of concepts. In [24], an algorithm is proposed based on the Jensen–Shannon divergence which measures the similarity between two probability distributions. It first calculates a probability distribution for each tag based on its co-occurrence with the most frequent words of the corpus and then compute the tags similarities.
This approach isn't semantic-based but it has been used as a comparison during our study.

Chapter 4

Disambiguation

Let's remember what we say about semantics in 2.1 : it is the study of meaning. The question that we will study here is the following : What happens when a word has several meanings ?

The disambiguation process aims to determine which one of the meanings is relevant in a specific context. In our case, we want to disambiguate the initial tags from a picture in order to propose relevant candidates. To achieve this, several approaches are presented with usages from the literature. Finally, we discuss its interest in the image annotation process and the issues it may rise.

4.1 How does it work

A word-sense disambiguation (or WSD) requires two inputs : a dictionary of senses and a corpus of language data to be disambiguated. Some WSD methods use machine learning therefore they also need a training corpus of language examples.
Let's consider the following sentences :

1. The *bass* line is too weak, we need to work on that !

2. I bought some grilled *bass* on the way home.

For the human brain, it's obvious that the first one refers to the musical context and the second to a special kind of fish. However, developing algorithms that replicate the this ability is often a difficult task. If we go further, it might be even more difficult to detect the precise meaning of *bass* in the first utterance since, in the musical context, it has two senses : the instrument and the sound.

In [25], Ide and Veronis present a nice state of the art of the disambiguation process. They explain that it takes two main steps :

1. Determination of all the different senses for every word relevant to the text under consideration
2. Assignation each occurrence of a word to the appropriate sense

Regarding step 1, *senses* can be pulled out different resources like our everyday dictionaries, groups of features (e.g synsets in WordNet, see 2.2.3) or transfer dictionaries, including translation in another language.
Step 2 is accomplished by reliance on two sources of information. The first one, already mentioned, is the *context* of the word which include information contained in the text in which the word appears and can also be filled by some extra-linguistic information about the text such as a situation. The second one is the external knowledge sources which can be used, such as encyclopedic resources, hand-made knowledge resources . . .
The authors then review all the WSD methods currently in use, see their paper for more information on the topic as well as the issues and problematic it rise.

4.2 Practical example : DBpedia Spotlight

Here we will jump from theory to reality and study how the DBpedia Spotlight[1] system process the WSD method in order to automatically annotate text documents with DBpedia (2.2.1) URIs. These explanations are pulled out of [5].

[1]http://dbpedia-spotlight.github.io/demo/

Before the disambiguation step, DBpedia Spotlight's algorithm detect entity names (or *surface forms* and pre-rank them with the use of the DBpedia Lexicalization dataset[2]. They add an important note : selecting fewer candidates can be good in terms of time cost but can also reduce recall if performed to aggressively.

After this selection phase, it use the context around the surface form (paragraphs) as information to find the appropriate disambiguation. DBpedia resources are modeled in a way that each resource is a point in a multidimensional space of words. They illustrate this process saying that this can be seen as an aggregation of all Wikipedia's paragraph mentioning the concept. Some metrics are computed :

- Term Frequency (TF) weight : represent the relevance of a word for a given resource
- Inverse Document Frequency (IDF) weight : represent the general importance of the word in the collection of DBpedia's resources

The authors explain then that IDF isn't that good for disambiguation (but pretty good for document retrieval). They propose a new weight called Inverse Candidate Frequency (ICF) in order to weight words based on their ability to distinguish between candidates for a given entity using this affirmation : The discriminative power of a word is inversely proportional to the number of DBpedia resources it is associated with.

At the end, given the chosen representation and the TF*ICF weights, the disambiguation process is similar as a ranking problem. They use the cosine similarity to rank vectors according to their context vector and the context surrounding the surface form.

To evaluate their approach, the authors picked 155,000 random ambiguous wikilinks (internal Wikipedia's links) and evaluate five methods. Their TF*ICF method shows promising results, since it matches correct sense at a score of 73.39% when the IF*IDF only has a score of 55.91%.

[2]http://wiki.dbpedia.org/lexicalizations

4.3 Our case

With the two previous sections, we can conclude that the disambiguation step is interesting and can be process in multiple ways. The results presented by DBpedia Spotlight's team are pretty good and encourage researchers to use this tool.

The facts that are important to remember about disambiguation before projecting it in our case is that the process needs resources and a context. Problem is that the image's annotations can't be considered as a context due to their organization and their content. Take these annotations from our dataset (9.1) for instance :

alley, building exterior, capital cities, city, city life, clear sky, colour image, day, dome, in a row, no people, outdoors, photography, travel destinations, window, czech republic, multi coloured, national landmark

First thing to note is that there is no grammar rules in the annotation format so it would be more difficult to define a proper context. Plus, as presented in [26], one of the interest of image annotation is to describe multiple semantics, resulting on a very diverse set of tags which make even more difficult the recognition of a context.
We will describe how we chose to use DBpedia Spotlight in 8.1.3 and what issues we faced.

Chapter 5

Existing approaches

We found in the literature lot of research which talk about multimedia tagging/annotation/labelling. These papers mainly use both the visual multimedia information (low-level features) and the textual information they might have at disposal (high-level features, meta-data). A review of the existing tagging-based applications is propose by Wand and al. in [27].
These works aren't our main focus, here we will present textual-only tagging approaches.

5.1 Mixing Statistics and WordNet

In [28], Jin and al. propose the integration of the WordNet (2.2.3) semantic resource in a statistic-based annotation process in order to remove irrelevant keywords.

Their algorithm is organized as follows : they first generate a set of keywords with the help of a statistical model called *Translation Model* (TM). Some of those candidates tags are relevant and some aren't.
In order to filter the irrelevant ones, they then compute several semantic similarity measures :

1. Lin - 3.2.2
2. Jiang and Conrath - 3.2.3

3. Banerjee and Pedersen - see [29]

Finally, they combine these metrics using Dempster-Shafer Theory ([30]) and the keywords with a resulting score under a chosen threshold are removed. Detailed method steps are presented in the original paper.

Concerning their results, they compare their *TMHD* proposed approach with a basic TM process. Based on a set of most frequently used keywords, they found that, on average, precisions values of TM and TMHD are respectively 14.21% and 33.11%. This indicates that TMHD is 56.87% better than TM. It is interesting to note that the recall score stays the same due to the fact that only irrelevant keywords are removed. They also compare *TMHD* to the use of individual measures with TM and the results aren't as good as those from their combination.
These results show the power of knowledge-based data and similarity measures when it's added to a statistical model.

5.2 Graph-cut based enrichment

In [31], Qian and Hua expose their graph-based approach of the tag enrichment process. They represent each initial tag of their corpus as a node and interlink them (using *n-links*). The weight of those n-links can be seen as the similarity between the two linked nodes, computed by the help of the Google distance [32]. They add two virtual nodes called *sink* and *source*. Then, they link all nodes to one of these virtual nodes using *t-links*.

The aim of their approach is to split all the tags into two distinct sets S (containing the source node) and T (containing the sink one) by assigning the labels s (source) if the tag is relevant to the image and t (sink) if not to the nodes. Then, they determine how many tags are relevant to the image by solving the combinational optimization problem through the graph.

This paper is really short and not very clear but it gives good ideas about the tags' representation as a graph and how to interlink them. According to the authors, the results are satisfactory.

5.3 Enrich Folksonomy Tag Space

Folksonomies are typical Web 2.0 systems that allow users to upload, tag and share content such as pictures, bookmarks . . . In [33], Angeletou and al. envisaged tag space enrichment with semantic relations by exploring online ontologies. Their method is composed of two phases :

- Concept identification
- Relation discovery

The first step is achieved by extracting concepts from online ontologies in which the local concept label matches the tag. In order to exploit all meanings, the authors retrieve all the potential semantic terms for each tag and then discover relation between them in the second phase. This means that no disambiguation is processed but it is a consequence of the relation discovery phase.
This phase consist of the identification of the relation between two tags *T1* and *T2*. Four kind of relations are distinguished :

- Subsumption relation : *T1* subClassOf *T2*
- Disjointness relation : *T1* disjointWith *T2*
- Generic relation : Property1 hasDomain *T1* and Property1 hasRange *T2*
- Sibling relation : *T1* and *T2* share a common ancestor
- Instance Of relation : *T1* instanceOf *T2*

These relations can be found by two ways : a relation can be declared within an ontology or, if no ontology contain such relation, one is made by crossing knowledge from different ontologies.

The author then present different experiments as well as some issues rose during this phase. One in particular is important to keep in mind : when users tags resources, especially pictures, they tend to tag them with specific vocabulary, mainly instances rather than *abstract* concepts. This can result on lot of "semantic noise" : tags which can't be match with concepts from online ontologies.

This paper is really interesting and approaches the topic in a very general point of view, which ensure the flexibility of its implementation. We will see in 7 that this method perfectly adapt to our study.

Chapter 6

Conclusion

In this State of the Art part, we've reviewed the generalities about the Semantic Web and presented several online semantic resources. We've also study the implementation of different similarity measures based on a generic ontology as well as their advantages and drawbacks regarding the the potential use-case. Then, we've shortly explain how the disambiguation process is achieve and how the previous measures can be used in it. Finally we've presented three approaches for semantic enrichment.

We will now propose a contribution to the topic by compiling all the notions we previously studied.

Part II

Contribution

Chapter 7

Proposed Methodology

Here we will present the chosen methodology, highly based on the three approaches presented in 5. This methodology aims to support different kind of experiments in order to try to answer our thesis questions previously stated in 1.3.

7.1 Global process

With this work we want to propose a prototype which semantically enrich images given an initial set of tags. This prototype will be based on 3 steps :

1. Concepts identification
2. Relations discovery
3. Candidates detection

As previously said, the two first steps will be similar as those presented in 5.3.

We also want to use a lemmatization process in order to group together the different inflected forms of a word so they can be analyzed as a single item. This will be achieve by using a hand-made tool based on regular expressions.

In order to compute semantic metrics, we will add virtual nodes to the graph, fulfiling the requirement of the Wu-Palmer evolved measure presented in 3.1.3. More details about the graph structure will be presented in 8.2.

We will set up different experiments, some based on the graph structure and some not. They will be run on a concrete images database and evaluated by several ways. Their results will then be presented, compared and discussed.

7.2 Knowledge bases

The difference we want to propose between our tool and the approaches presented in 5 is to use several online ontologies in order to detect concepts and to create relations between them. We selected two resources : DBpedia and WordNet, already presented in 2.2.1 and 2.2.3. We will need ways to query these ontologies and browse them. In order not to get two separated graphs, we will create interlinks between concepts from both of the resources.
Eventually, the candidates we will propose as new annotations will also come from both of the ontologies.

7.3 Data representation

As in 5.2, we want to represent our data as a graph. We will extract concepts from our chosen ontologies and save them as vertexes of our graph. We will interlink them using the already existing relations in their respective ontologies and add new ones we will detect by ourself. The graph structure is detailed in 8.2 and an example is available Figure 8.1.

Chapter 8

Proposed Architecture

8.1 Technology Choices

8.1.1 Java Language

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is also a cross-platform language which means that it would be possible to use it without any recompilation needed. Java is very easy to use, well documented a has the support of a large community (more than 9 million developers reported). Therefore, lot of libraries are available, we will present some of them below.

8.1.2 Neo4j

Graph-based databases are very intuitive to work with and allow the user to model the world as he experience it. The model schema isn't rigid and the user can edit it at anytime, adding new entities or new kind of relationships. Neo4j is an open-source graph database, implemented in Java (so cross-platform), maturing for 15 years and currently running version 2.2. It is the most popular graph database nowadays[1], has a great scalability, a strong community and has its own query language : Cypher.

[1]http://db-engines.com/en/ranking/graph+dbms

8.1.3 DBpedia Spotlight

As presented in 2.2.1, DBpedia propose a nice tool to automatically annotating mentions of DBpedia resources in text : DBpedia Spotlight. We chose to use it even if we can't exploit its full potential since we don't have a context. Its usage is really simple and performances are quite ok given our use-case. See Code 8.1 for a sample response.

8.1.4 Semantic Resources Libraries

We needed to access the chosen online ontologies (DBpedia's and WordNet's) from our prototype. To achieve this, we used the fact that Java is very popular and lot of libraries are available.

JENA We chose Apache JENA ARQ[2] to query the RDF-base schema of DBpedia. This solution is stable and maintained by a famous structure : Apache. Using it was really simple.

JAWS Regarding WordNet, we used JAWS[3] which has been developed and is maintain by a member of the Southern Methodist University (Dallas, Texas). Its last version is a bit old but this isn't an issue since WordNet's upgrades have also stopped. This library was also deeply intuitive and easy to use.

8.1.5 JSoup

Our two last experiments are based on Wikipedia's web-pages. Therefore we needed a way to crawl and extract content from them. The JSoup[4] library was a perfect asset to achieve this. It is open-source, implements the WHATWG HTML5 specification, and parses HTML to the same DOM as modern browsers do. It also allows the user to build specific queries to access particular elements in the DOM.

[2]https://jena.apache.org/documentation/query/index.html
[3]http://lyle.smu.edu/ tspell/jaws/
[4]http://jsoup.org/

8.1.6 Stanford NLP

Crawling web-pages is a thing, but extracting relevant data from it is another one. The Stanford NLP Research Group[5] has released several libraries in different programming languages including Java. Those libraries can achieve many things such as sentence segmentation, Part-of-speech (POS) tagging, named entities recognition and so on. . . We used the POS Tagger to extract nouns from Wikipedia paragraphs.

8.2 Graph Structure

8.2.1 Vertexes

Each vertex represents a semantic concept (virtual or real).
Virtual ones are concepts created in purpose to perform operations. Here we're talking about 2 different kinds of nodes :

- Base concepts : those are created in order to represent the originals tags. Their URIs follow this pattern : "base:TAG" (ex : base:dog) and their *TAG* value is lemmatized (dogs -> dog).
- Top/Bottom concepts : these two concepts (virtual:top and virtual:bottom) are essential in the compuation of the Wu-Palmer evolved measure (see 3.1.3).

Real nodes are semantic nodes linked to entities, classes (DBpedia) or synsets (WordNet):

- WordNet's nodes pattern : "Wordnet:TAG" (ex: Wordnet:plant)
- DBpedia's nodes pattern : "DPedia's concept URI"
  (ex: http://dbpedia.org/resource/London)

8.2.2 Edges

There are 3 kind of edges :

[5]http://nlp.stanford.edu/

- VIRTUAL : represent virtual links between nodes, not present in any of the semantic resources, created by the algorithm
- EQUIV : represent an equivalence between nodes from 2 different ontologies (ex : http://dbpedia.org/resource/Dog and Wordnet:dog)
- PARENT : represent a semantic link which is of type "IS-A" (implemented by the rdfs:subClassOf predicate in DBpedia and the hyperonym/holonym relation in WordNet)

8.2.3 Construction

Now that we have described our graph, we will present how it is built.

Given an initial set of input tags, the first step of our construction process is to detect semantic concepts among them. To achieve this, we use DBpedia Spotlight (shortly presented in 8.1.3) and its REST endpoint. It takes as input a string (here the list of tags, separated by commas) and returns a JSON object containing detected semantic concepts, see Code 8.1 for a sample response. We then use the JAWS library and request all input tags to our WordNet database. This returns us a list of synsets.

We have now to create the hierarchical tree for each of this base concepts and stop when we reach the ontology's root. In order to do so, we again split the task according to the ontology the concept comes from. For DBpedia's, we send SPARQL requests through the JENA library. Code 8.2 if the resource is an entity and we need to find its class. Code 8.3 if the resource is a class and we so need to find it's superclass.
It is a bit more simple with the WordNet's concepts. Since we have the initial synsets, we can easily navigate into them and extract hypernyms. This whole process is recursive, for each new concept we start again.

The last step of our process is the detection of equivalences. To do so we start from the DBpedia's nodes, get their label and request WordNet. If a correspondence is found, then we create the equivalent node, interlink it via an EQUIV edge to the initial one and

```
{
  "@text": "obama,sea,wolf,germany",
  "@confidence": "0.0",
  "@support": "0",
  "@types": "",
  "@sparql": "",
  "@policy": "whitelist",
  "Resources":   [
        {
      "@URI": "http://dbpedia.org/resource/Michelle_Obama",
      "@support": "321",
      "@surfaceForm": "obama",
      "@offset": "0",
      "@similarityScore": "0.143932044506073",
      "@percentageOfSecondRank": "0.7407412852273438"
    },
        {
      "@URI": "http://dbpedia.org/resource/Sea",
      "@support": "1016",
      "@surfaceForm": "sea",
      "@offset": "6",
      "@similarityScore": "0.11291095614433289",
      "@percentageOfSecondRank": "0.5072430234623665"
    },
        {
      "@URI": "http://dbpedia.org/resource/Wolf_%28film%29",
      "@support": "50",
      "@surfaceForm": "wolf",
      "@offset": "10",
      "@similarityScore": "0.12193222343921661",
      "@percentageOfSecondRank": "0.888975663754396"
    },
        {
      "@URI": "http://dbpedia.org/resource/Germany",
      "@support": "106627",
      "@surfaceForm": "germany",
      "@offset": "15",
      "@similarityScore": "0.10691326856613159",
      "@percentageOfSecondRank": "0.9705061222373553"
    }
  ]
}
```

Code 8.1: DBpedia Spotlight sample response

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
SELECT ?o1 ?o2 where {GRAPH <http://dbpedia.org> { CONCEPT_URI rdf:type ?o1
FILTER regex(?o1, '^^http://dbpedia.org/ontology/(?!Wikidata:)')
OPTIONAL{ ?o1 rdfs:label ?o2
FILTER(langMatches(lang(?o2), 'EN'))}}}
```

Code 8.2: Find entity's class

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?o ?o2 WHERE{GRAPH <http://dbpedia.org> { CONCEPT_URI rdfs:subClassOf  ?o
FILTER regex(?o, '^^http://dbpedia.org/ontology/(?!Wikidata:) || ^^http://www.w3.org/2002/07/owl#
     Thing')
OPTIONAL{ ?o rdfs:label ?o2
FILTER(langMatches(lang(?o2), 'EN'))}}}
```

Code 8.3: Find class' superclass

build its generalization tree as presented above.

An extract of our constructed graph is presented in figure 8.1. It comes from the browser interface of Neo4j and we can here see all kind of nodes and relationships. More precisely, the bottom-left node is *virtual:bot* and the top-right is *virtual:top*. Between them, we chose to display two initial tags (*base:chapel* and *base:cologne cathedral*) and their generalizations. We can observe an *EQUIV* relation between the red (*http://dbpedia.org/ontology/Building*) and the green (*Wordnet:building*) circles. Both WordNet and DBpedia ontologies' root are displayed in the blue circles (*Wordnet:entity* and *http://www.w3.org/2002/07/owl#Thing*) and linked to our *virtual:top* root.

8.2.4 Pro-Cons

This graph-based model approach has several benefits : it is more intuitive to imagine the inheritance of a concept, the computation of basic metrics such as shortest path between two concepts or their Least Common Subsumer (LCS) is really easy, the schema isn't rigid and has, in fact, evolved during the project . . .

However, our choice also has drawbacks : No disambiguation system has been implemented which means that all parents of a node are added to the graph making it always bigger and slower to browse. Given the graph's size and my machine performances, the integration of the Wu-Palmer evolved measure wasn't possible (but implemented and tested on smaller graphs).

8.3 Code and Procedure Explanations

This work has been split into several Java projects, each completing a special task. In this section we will present these projects and detail their main classes.

8.3.1 TreeGenerator

This project achieve the creation of a data model representing our hierarchical concepts organization. We chose to create a Java object-based data model in order to be able to do different things with it, like importing it in a graph database.
It is split into one main class and two packages :

- **treegenerator.model** : This package contains all the classes that represent the entities of our project. These classes are organized as an inheritance schema where :
  - **OnlineConcept** is the mother class which define the common attributes and initialize the *parents* and *childs* vectors. These vectors store all the concepts that are linked to this entity, as generalizations or specializations.
  - **DBpediaConcept** inherit from OnlineConcept and define specific constructors which can either take a JSON object or an URI as input.
  - **WordNetConcept** inherit from OnlineConcept, defines a specific constructor and stores the Concept associated synset.
  - **TypeTerm** enumerate all the concept's types.

  Each Concept can have multiple parents and multiple childs. Class diagram is available in Figure 8.2.
- **treegenerator.services** : This package contains three tools that we use for the data-model creation:
  - **SpotlightConnection** : This class operates the connection to DBpedia Spotlight's REST service, sending the initial tags as an input text and returning a JSON object.
  - **WordNetConnection** : This class query our download WordNet version by retrieving the synsets associated to an input keyword.
  - **Inflector** : This class performs the lemmatization process we presented in 7.1. It is based on a list of regular expressions and extends an initial work from *chuyeow*[6].

[6] https://github.com/chuyeow

- **TreeGenerator** : The main class, which contains methods to generate the data-model. Few are important and are presented below :
  - **run** : the main method, initialize all the parameters and start the construction process.
  - **runAndGenerateGEXF** : same as above but also generate a GEXF[7] file containing our data model.
  - **getAncestors** : define all of our process' steps presented in 7.1
  - **recursionTerms** : the recursive function which is called for every new concept in order to build its generalization.

8.3.2 GexfParserForNeo4jDB

We won't detail this project since it has only be an interim project. Its aim is to parse a GEXF file and create a model in a Neo4j database. It is split into two packages :

- **gexfparserforneo4jdb.gexf** : Parse the GEXF file into a Java Graph object.
- **gexfparserforneo4jdb.neo4j** : Connects to the Neo4j database and transform the Java Graph object in a Neo4j graph model.

8.3.3 SemObsNeo4j

This project achieve several tasks. First it allows the user to connect to an external source of data (CSV file or PostgreSQL database) and run the generalization tree step by calling the TreeGenerator project (8.3.1).
Then it make use of the Java data-model created by TreeGenerator and directly parse it into a Neo4j graph without using an interim GEXF file. This whole creation phase can be repeated as much as the user wants, the nodes won't be created twice since we check for the existing ones before any upload operation.
Eventually, the project also run our experiments and compute the results into several files.
It is split into two packages each containing a lot of classes we will now present :

[7]http://gexf.net/format/

- **semopsneo4j** : This package contains the main class of our project and several useful hand-made tools.
  - **SemObsNeo4j** : This class loads the properties of our project, initializes the Neo4j database if needed and starts the chosen task.
  - **SemDistance** : This class performs the calculation of semantic measures and propose several ways to export its results as matrices.
  - **DataReader** : This class connects to the external source of data and parse it into a Java Map object in which a key (the image's ID) is associated with an ArrayList of its tags.
  - **ModelImporter** : This class uses the data model created by TreeGenerator in order to build our graph model. It also checks for already existing concepts to avoid duplication.
  - **BFSTraverser** : This class implements a Breadth-First Search (BFS) algorithm using our graph structure. It is used in our graph-based experiments presented in 9.3.2.
  - **WikipediaCrawler** : This class crawl the content of a Wikipedia's page based on an input tag. Obviously, nothing is done if the requested URL doesn't exist.
  - **NLPParser** : This class performs several operations provided by the Standford NLP library already presented in 8.1.6. We mainly use the Part-Of-Speech module which associate a label to each word.
  - **PairNodeScore** : This class is a hand-made pair structure which contains a Node object and a double field. This allows us to store the detected candidates in our experiments and perform operations on them.
- **experiment** : This package contains our experiments classes and a main class for the whole evaluation step. We won't detail the experiments algorithms since it will be done in the next chapter. The experiments classes are organized as follows :
  - **RelevantTagsExperiment** : The mother abstract class, define all common attributes such as the list of considered initial tags, the execution time or the selected candidates.

This class also contains the export methods, more about the exports will be presented in the next chapter.

- **RelevantTagsExperimentDN** : Inherits from RelevantTagsExperiment and implements a direct parents search.
- **RelevantTagsExperimentLists** : Inherits from RelevantTagsExperiment and implements a search based on lists of initial tags.
- **RelevantTagsExperimentWL** : Inherits from RelevantTagsExperimentLists and implements a candidate's global score computation based on the whole list of initial tags.
- **RelevantTagsExperimentSL** : Inherits from RelevantTagsExperimentLists and implements a candidate's global score computation based on sub-lists of initial tags.
- **RelevantTagsExperimentWikiLinks** : Inherits from RelevantTagsExperiment and implements search among internal Wikipedia's links.
- **RelevantTagsExperimentWikiContent** : Inherits from RelevantTagsExperiment and implements a search among all nouns in a Wikipedia's page first paragraph.

Class diagram of our experiments classes is available on Figure 8.3.

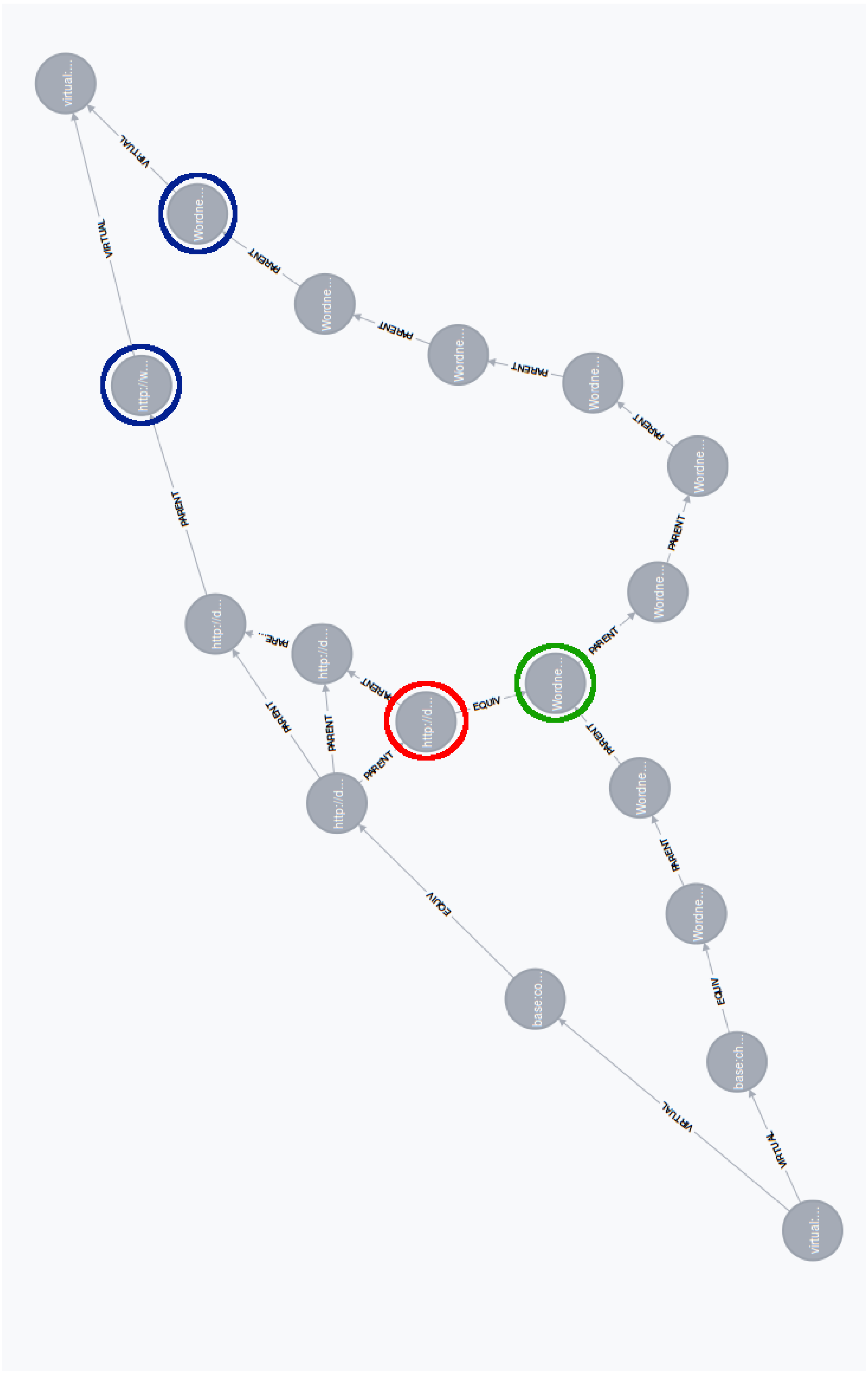

FIGURE 8.1: Graph extract

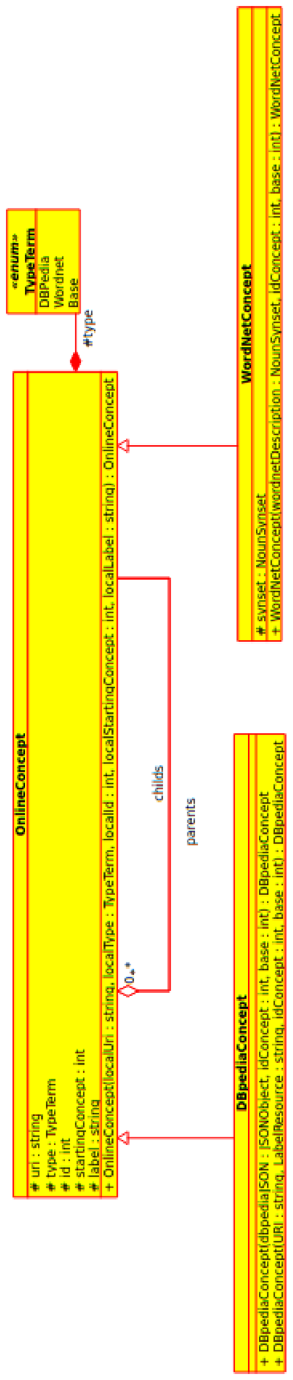
«enum»
TypeTerm
DBPedia
Wordnet
Base
#type
OnlineConcept
# uri : string
# type : TypeTerm
# id : int
# startingConcept : int
# label : string
+ OnlineConcept(localUri : string, localType : TypeTerm, localId : int, localStartingConcept : int, localLabel : string) : OnlineConcept
0..*
childs
parents
WordNetConcept
# synset : NounSynset
+ WordNetConcept(wordnetDescription : NounSynset, idConcept : int, base : int) : WordNetConcept
DBpediaConcept
+ DBpediaConcept(dbpediaJSON : JSONObject, idConcept : int, base : int) : DBpediaConcept
+ DBpediaConcept(URI : string, LabelResource : string, idConcept : int, base : int) : DBpediaConcept

FIGURE 8.2: TreeGenerator class diagram

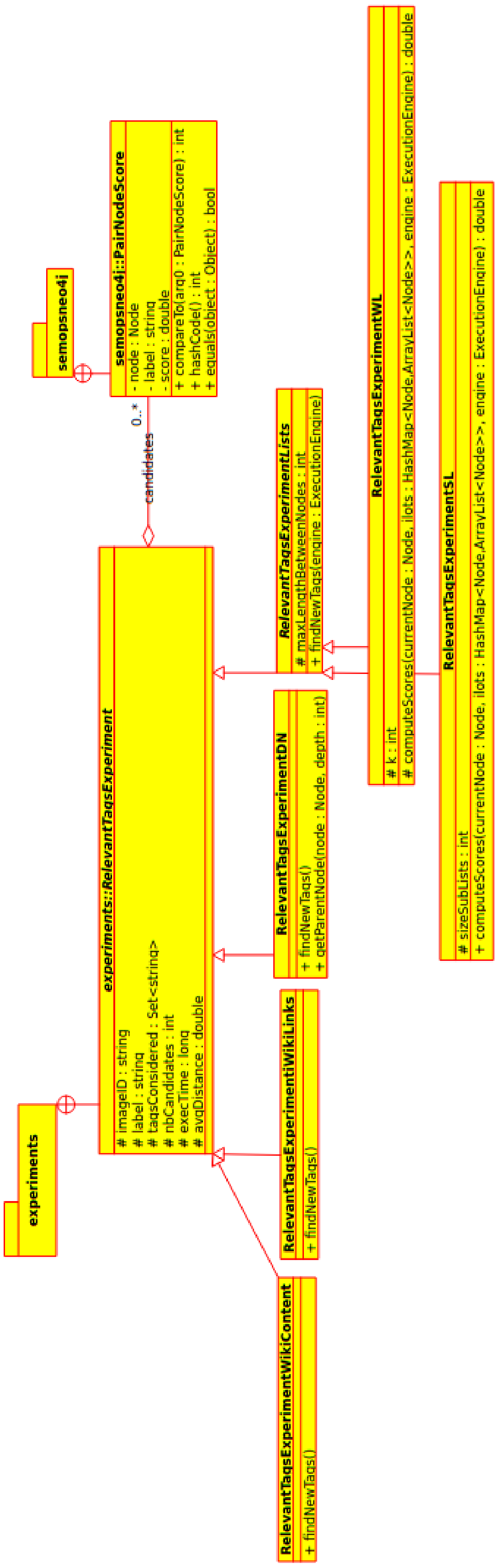
semopsneo4j
semopsneo4j::PairNodeScore
- node : Node
- label : string
- score : double
+ compareTo(arg0 : PairNodeScore) : int
+ hashCode() : int
+ equals(object : Object) : bool
0..*
candidates
experiments
experiments::RelevantTagsExperiment
# imageID : string
# label : string
# tagsConsidered : Set<string>
# nbCandidates : int
# execTime : long
# avgDistance : double
RelevantTagsExperimentLists
# maxLengthBetweenNodes : int
+ findNewTags(engine : ExecutionEngine)
RelevantTagsExperimentWL
# k : int
# computeScores(currentNode : Node, ilots : HashMap<Node,ArrayList<Node>>, engine : ExecutionEngine) : double
RelevantTagsExperimentSL
# sizeSubLists : int
+ computeScores(currentNode : Node, ilots : HashMap<Node,ArrayList<Node>>, engine : ExecutionEngine) : double
RelevantTagsExperimentDN
+ findNewTags()
+ getParentNode(node : Node, depth : int)
RelevantTagsExperimentWikiLinks
+ findNewTags()
RelevantTagsExperimentWikiContent
+ findNewTags()

FIGURE 8.3: Experiments class diagram

Chapter 9

Experiments

As previously said, we implemented five experiments : three are based on our graph structure and two directly use the content of Wikipedia's pages. We will now present the dataset we used, the experiments' implementation, the results we obtained and discuss them.

9.1 Dataset

Our experiments have been made using a database of images pulled out the website Flickr[1] by fellow students[2]. Some statistics about the database are presented in Table 9.1.

Nb. of images	Distinct unique tags	Avg. nb. of initial tags
55600	10261	17.69

TABLE 9.1: Database statistics

[1]https://www.flickr.com/

[2]Thanks to Adela Neacsu, Sorana Capalnean, Iler Viraragavane and Gaëtan Deshayes

The top 20 used tags are also available below.

1. photography : 55002
2. colour image : 49734
3. outdoors : 49129
4. no people : 44892
5. day : 37337
6. sky : 23964
7. travel destinations : 17713
8. cloud : 16906
9. tree : 12890
10. scenics : 12134
11. tranquility : 11668
12. tranquil scene : 10526
13. landscape : 10303
14. building exterior : 9941
15. beauty in nature : 9686
16. people : 8198
17. sea : 8197
18. capital cities : 8144
19. close-up : 8083
20. reflection : 7393

For the purpose of our tests, we created a graph based on 1350 images that are located around Berlin (Germany). The PostgreSQL query we used to extract them is available in Code 9.1.

```sql
SELECT
  imagefiltred.id,
  imagetagfiltred.tag
FROM
  gettytag,
  imagefiltred,
  imagetagfiltred
WHERE
  imagefiltred.id = imagetagfiltred.imageid AND
  gettytag.text = imagetagfiltred.tag AND
  gettytag.gettytag = true AND
  imagefiltred.lat >= 50 AND
  imagefiltred.lat <= 55 AND
  imagefiltred.lon >= 5 AND
  imagefiltred.lon <= 1;
```

CODE 9.1: PostgreSQL query to retrieve images around Berlin

This gave us a graph which contains 5212 nodes and 7543 relations, their distributions are presented in Tables 9.2 and 9.3. An example of a Cypher query counting the WordNet's nodes is also presented in Code 9.2.

Virtual	**Base**	**WordNet**	**DBpedia**
2	1619	2613	978

TABLE 9.2: Nodes' types distribution

VIRTUAL	**PARENT**	**EQUIV**
1612	3592	2339

TABLE 9.3: Edges' types distribution

```
MATCH (w:Concept) WHERE w.uri=~"Wordnet:.*" RETURN COUNT(w)
```

CODE 9.2: Cypher query to count WordNet's nodes

Thanks to the Gephi software[3], we were able to compute some statistics about our network and we learned that the longest shortest path between two nodes (also called graph's diameter) of our graph is of 20 and the average path length between two nodes is of 6.64.

9.2 Computer Characteristics

The computer we will use for our operations has the following characteristics :

1. OS : Ubuntu 14.04 LTS (64 bits)
2. RAM : 3,7 Go
3. CPU : Intel Celeron(R) CPU 847 @ 1.10GHz x 2

9.3 Algorithms Explanation

The algorithms we will present in this section are supported by the experiments classes we detailed in 8.3.3.

[3]http://gephi.github.io/

```
MATCH path=(node1 {uri: URI_NODE1})-[:PARENT|:EQUIV*]->
lca
<-[:PARENT|:EQUIV*]-(node2 {uri:URI_NODE2})
RETURN lca
ORDER BY length(path)
LIMIT 1
```

CODE 9.3: Cypher query to retrieve two nodes' LCS

9.3.1 Matrices exports

As presented in 8.3.3, we developed a way to calculate semantic distances between nodes and export the results as matrices. To achieve this, we implemented two measures : the Shortest Path distance (see 3.1.1) and the evolved Wu-Palmer algorithm (see 3.1.3). This last implementation also implies the development of a Cypher query which retrieves two nodes' LCS. See Code 9.3 for the complete query.

With our architecture, it is possible to compute a distance matrix for each image based on its initial tags or to compute a bigger and global matrix taking all images' tags into account.

These exports are a support to the comparison of our implemented semantic distances and other metrics one would like to use, like the statistical matrices produced by the tool presented in 3.3.

The computation of such matrices cost around 20 seconds for 15 tags on the machine presented in 9.2.

9.3.2 Graph-based experiments

The two first experiments, WholeList (short WL) and SubLists (short SL), are based on the same algorithm of candidates detection. This algorithm make use of Breadth-First search (BFS) traversers in order to browse the graph starting from the initial tags' nodes. The BFS implementation is available on Code 9.4 and the candidates detection algorithm on Code 9.5.

```java
// java.utils imports omitted for brevity

import org.neo4j.graphdb.Direction;
import org.neo4j.graphdb.Node;
import org.neo4j.graphdb.Relationship;

/*
 * Breadth First Search Traverser
 */
public class BFSTraverser {
  protected Queue<Node> queue;
  protected List<Node> visitedNodes;
  protected Node rootNode;

  public BFSTraverser(Node rootNode) {
    this.queue = new LinkedList<Node>();
    this.rootNode = rootNode;
    this.queue.add(this.rootNode);
    this.visitedNodes = new ArrayList<Node>();
  }

  //Getters and setters omitted for brevity

  /*
   * Process BFS algorithm and return the last reached Node
   */
  public Node next(){
    Node removedNode = queue.remove();
    visitedNodes.add(removedNode);

    Iterator<Relationship> itRel = removedNode.getRelationships(Direction.OUTGOING).iterator();
    while(itRel.hasNext()){
      Node currentParent = itRel.next().getEndNode();

      if(!visitedNodes.contains(currentParent)){
        queue.add(currentParent);
      }
    }
    return removedNode;
  }
}
```

Code 9.4: Java BFS implementation

9.3.2.1 Lists - WL

The difference between WL and SL is located line 26 of Code 9.5 : the score computation function doesn't take the same things into account. Let's detail this.
In the WL experiment, we compute a global score using all the initial tags (Whole List). This global score is composed of all the scores between the considered candidate and the initial nodes as follows :

$$globalScore(currentNode) = \sum_{0 \leq n < nbNodes} \frac{1}{score(currentNode, nodes(n))^k} \tag{9.1}$$

The *score* function can be one of the measures we presented in 3. Due to computer performances we had to go with the Shortest Path distance. k is a chosen integer which

```java
public void findNewTags(ExecutionEngine engine){
  // Init
  HashMap<Node,BFSTraverser> traversers = new HashMap<Node, BFSTraverser>();
  HashMap<Node, ArrayList<Node>> ilots = new HashMap<Node, ArrayList<Node>>();
  ArrayList<PairNodeScore> tagsCandidats = new ArrayList<PairNodeScore>();
  boolean tousParcoursTermines = false;
  for(Node tagNode : RunExperiments.nodes.values()){
    traversers.put(tagNode, new BFSTraverser(tagNode));
    ilots.put(tagNode, new ArrayList<Node>());
  }
  // Parcours
  Transaction tx = RunExperiments.graphDb.beginTx();
  try {
    while(tagsCandidats.size()<nbCandidates && !tousParcoursTermines){
      tousParcoursTermines=true;
      for(BFSTraverser traverser : traversers.values()){
        if(traverser.hasNext()){
          Node currentNode = traverser.next();
          Node currentRoot = traverser.getRootNode();
          ilots.get(currentRoot).add(currentNode);
          if(!containsNode(currentNode, tagsCandidats)){
            List<Node> occurences = intersection(ilots, currentNode);
            if(occurences.size()>1){
              PairNodeScore p = new PairNodeScore(currentNode, 0.0);
              if(!RunExperiments.nodes.containsKey(p.getLabel())){
                double globalScore = computeScores(currentNode,ilots, engine);
                p.setScore(globalScore);
                tagsCandidats.add(p);
              }
            }
          }
          if(traverser.hasNext())
            tousParcoursTermines=false;
        }
      }
    }
    tx.success();
  } finally {
    tx.close();
  }
  candidates = tagsCandidats;
  Collections.sort(candidates);
}
```

Code 9.5: Candidates detection algorithm

has for consequence to favor (or not) really close nodes.

9.3.2.2 Lists - SL

The SL experiment is based on the same approach than his brother WL. The difference is that, when it comes to global score computing, we only take into account a determined number of initial nodes. This results on a small modification of the algorithm : before any global score calculation we store all the *score* results and only use some of them in equation (9.1). This has for consequence that the very far nodes don't pollute our results by inserting noise.

9.3.2.3 Direct Neighbors

This last graph-based experiment version is more direct : we search for direct neighbors of the input nodes. In concrete terms, for each of the input nodes, we store all the parents nodes with a maximal distance of 2. We count how many times each parent is found and return the most frequent ones. This method is expected to give good results when the initial tags are close semantically speaking (boat, sea, sail . . .) but can return abstract results in the case of a very diverse set of tags, and even more if this set is small.

9.3.3 Plain-text experiments

We will now present the last two experiments which are directly based on the content of Wikipedia's pages, extracted with the help of the JSoup library (see 8.1.5). The process is basically the same : we search for wikipedia's pages matching the URL *https://en.wikipedia.org/wiki/+TAG*, if found we extract the content from the first paragraph and then operate the candidate detection step.

9.3.3.1 WikiLinks

For the WikiLinks experiment, we only care about internal links (called wikilinks). We access them using CSS queries :

- Access the 1st paragraph : *div#mw-content-text > p*
- Access wikilinks : *a[href =/wiki/(?!Help:)(?!File:)(?!Wikipedia:)]*

Here again, we store each of the links and sum its occurrences, we then propose as final candidates the most frequent ones.

9.3.3.2 WikiContent

The WikiContent experiment is very similar to the WikiLinks one. The difference is that we consider here all the nouns in the first paragraph of the Wikipedia's page. In order to achieve this, we extract the first paragraph with the same CSS query presented above

and then parse its content using the NLP library. The Part-of-speech module gives us a list of tokens as well as their POS label (see below for the complete list of POS labels) and we then only keep the nouns (NN* labels : common and proper nouns, singular and plural).
Here again, we store each of the links and sum its occurrences, we then propose as final candidates the most frequent ones.

- CC Coordinating conjunction
- CD Cardinal number
- DT Determiner
- EX Existential there
- FW Foreign word
- IN Preposition or subordinating conjunction
- JJ Adjective
- JJR Adjective, comparative
- JJS Adjective, superlative
- LS List item marker
- MD Modal
- NN Noun, singular or mass
- NNS Noun, plural
- NNP Proper noun, singular
- NNPS Proper noun, plural
- PDT Predeterminer
- POS Possessive ending
- PRP Personal pronoun
- PRP$ Possessive pronoun
- RB Adverb
- RBR Adverb, comparative
- RBS Adverb, superlative
- RP Particle
- SYM Symbol
- TO to
- UH Interjection
- VB Verb, base form
- VBD Verb, past tense
- VBG Verb, gerund or present participle
- VBN Verb, past participle
- VBP Verb, non 3rd person singular present
- VBZ Verb, 3rd person singular present
- WDT Whdeterminer
- WP Whpronoun
- WP$ Possessive whpronoun
- WRB Whadverb

9.4 Results and Analysis

9.4.1 Evaluation methodology

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

Chapter 10

Conclusion

10.1 Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

Bibliography

[1] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. 2001.

[2] Eric Miller. An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25(1):15–19, 1998.

[3] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 5:1–29, 2014.

[4] Eric Prud'Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[5] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.

[6] Alexandre Passant. dbrec—music recommendations using dbpedia. In *The Semantic Web–ISWC 2010*, pages 209–224. Springer, 2010.

[7] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. Media meets semantic web–how the bbc uses dbpedia and linked data to make connections. In *The semantic web: research and applications*, pages 723–737. Springer, 2009.

[8] Jelena Tešić. Metadata practices for consumer photos. *MultiMedia, IEEE*, 12(3): 86–92, 2005.

[9] Rosa Stern and Benoît Sagot. Détection et résolution d'entités nommées dans des dépêches d'agence. In *Traitement Automatique des Langues Naturelles: TALN 2010*, 2010.

[10] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[11] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*, volume 2, pages 2–2, 2001.

[12] Ray Richardson, A Smeaton, and John Murphy. Using wordnet as a knowledge base for measuring semantic similarity between words, 1994.

[13] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

[14] Philip Resnik. Disambiguating noun groupings with respect to wordnet senses. *arXiv preprint cmp-lg/9511006*, 1995.

[15] Ellen M Voorhees. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 171–180. ACM, 1993.

[16] Satanjeev Banerjee and Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing*, pages 136–145. Springer, 2002.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[18] Thomas Deselaers and Vittorio Ferrari. Visual and semantic similarity in imagenet. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1777–1784. IEEE, 2011.

[19] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[20] Haïfa Zargayouna, Sylvie Salotti, et al. Mesure de similarité dans une ontologie pour l'indexation sémantique de documents xml. *IC 2004*, pages 249–260, 2004.

[21] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[22] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

[23] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[24] Hatem Mousselly-Sergieh, Elöd Egyed-Zsigmond, G Gianini, M Döller, Harald Kosch, and Jean-Marie Pinon. Tag similarity in folksonomies. INFORSID, 2013.

[25] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational linguistics*, 24(1):2–40, 1998.

[26] Xueming Qian, Xian-Sheng Hua, Yuan Yan Tang, and Tao Mei. Social image tagging with diverse semantics. *Cybernetics, IEEE Transactions on*, 44(12):2493–2508, 2014.

[27] Meng Wang, Bingbing Ni, Xian-Sheng Hua, and Tat-Seng Chua. Assistive tagging: A survey of multimedia tagging with human-computer joint exploration. *ACM Computing Surveys (CSUR)*, 44(4):25, 2012.

[28] Yohan Jin, Latifur Khan, Lei Wang, and Mamoun Awad. Image annotations by combining multiple evidence & wordnet. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 706–715. ACM, 2005.

[29] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *IJCAI*, volume 3, pages 805–810, 2003.

[30] Glenn Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[31] Xueming Qian and Xian-Sheng Hua. Graph-cut based tag enrichment. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1111–1112. ACM, 2011.

[32] Rudi L Cilibrasi and Paul Vitanyi. The google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):370–383, 2007.

[33] Sofia Angeletou, Marta Sabou, Lucia Specia, and Enrico Motta. Bridging the gap between folksonomies and the semantic web: An experience report. 2007.