Image annotation network

Mael Ogier

Département Informatique
INSA Lyon
2014-2015

Sous la responsabilité de :
Dr. David Coquil : Universität Passau
Dr. Elöd Egyed-Zsigmond : Département Informatique

Résumé De nos jours, le media image est l'un des plus utilisé, il est facile à capturer, relativement léger informatiquement parlant et parle à tous sans soucis de langage. Nombre d'entreprises utilisent des images tous les jours, à commencer par les médias. Elles ont donc besoin d'outils afin de retrouver la bonne image au bon moment. En tant qu'être humain, il nous est beaucoup plus facile de décrire ce que nous recherchons par nos propres mots plutôt qu'à l'aide d'histogrammes de distribution de couleurs. C'est pourquoi il est intéressant d'ajouter des annotations (ou tags) aux images afin de les décrire de façon plus naturelle. Avec ce projet, nous proposons un prototype d'enrichissement sémantique de ces annotations basé sur le web sémantique et sur une architecture des données en mode graphe. Cinq expériences sont présentées et les résultats sont évalutés et discutés. Enfin, quelques pistes sont données pour de futures recherches.

Abstract 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.

Keywords: image annotation, semantic enrichment, semantic web, graph model

1 Introduction

1.1 Context

This PFE 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.
I realised my project at the University of Passau during the last semester of my 5th year of study. During these 6 months, I had the amazing opportunity to discover the way of study in Germany and the organization of the University of Passau. There are 12,000 students in Passau (for 50,000 inhabitants) and 600 study computer science. The FIM faculty (Fakültät für Informatik und Mathematik) is composed of 24 chairs, I was part of Pr. Kosch's which employ master and doctors students from different countries (one of the student I shared the office with was from Tunisia).
I worked on my own on this project under the supervision of two tutors, Dr. David Coquil on the German side and Dr. Elöd Egyed-Zsigmond on the French side. They were a precious help during the whole PFE process in term of advices and suggestions of work.

1.2 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.
The 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 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 project 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.

1.3 State of the Art

Semantic Web 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.

Semantic Resources

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).
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.
DBpedia also provides useful web services and HTTP endpoints. DBpedia Spotlight, which highlight DBpedia concepts in an input text is described in [5]. The official DBpedia SPARQL endpoint[4] allows the user to send SPARQL queries to the online Virtuoso Triple Store by using the browser interface or by sending a HTPP request.

WordNet WordNet[5] is a lexical database of English which has been presented for the first time in 1995 in [6]. 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

[1] http://wiki.dbpedia.org/
[2] https://en.wikipedia.org/
[3] Complete classes tree : http://mappings.dbpedia.org/server/ontology/classes/
[4] http://dbpedia.org/sparql
[5] https://wordnet.princeton.edu/

distinguish among Types (common nouns, verbs...) and Instances (specific persons...). Synsets are interlinked 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 dataset as well as a specific library according to the code language he's using.

Similarity Measures

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.

Wu-Palmer evolved In [10], Zargayouna exposes a consequence of the Wu-Palmer formula which can, according to the context of use, gives false results. Indeed, in Wu-Palmer, 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). This can be an issue if one want to give an advantage to the father-child relation instead of the brother-brother one.
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.
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 Wu-Palmer. In the brother-brother relation yet, this part isn't null and the similarity decrease.

Existing approaches

Graph-cut based enrichment In [7], 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 [8]. 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.

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 [9], 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, Disjointness, Generic, Sibling and Instance Of. 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.

1.4 Method

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. Concept identification
2. Relation discovery
3. Candidates detection

As previously said, the two first steps will be similar as those presented in 1.3. The difference we want to propose 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 above.
The last step of our method is the detection of potential new tags by using the graphs previously created. Three experiments are proposed based on this. We also further investigated DBpedia's environment by directly using its sources (the Wikipedia's pages) for two other experiments.

2 Contribution

2.1 Architecture

In order to support the method presented in 1.4 some technology choices were made. In this section we will present the most important of them and especially detail the structure of our graph-based data model.

Technologies

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.

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[6], has a great scalability, a strong community and has its own query language : Cypher.

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.
We chose Apache JENA ARQ[7] to query the RDF-base schema of DBpedia. This solution is stable and maintained by a famous structure : Apache. Using it was really simple.
Regarding WordNet, we used JAWS[8] 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.

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[9] 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.

[6] http://db-engines.com/en/ranking/graph+dbms
[7] https://jena.apache.org/documentation/query/index.html
[8] http://lyle.smu.edu/ tspell/jaws/
[9] http://jsoup.org/

Stanford NLP Crawling web-pages is a thing, but extracting relevant data from it is another one. The Stanford NLP Research Group[10] 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.

Graph Structure The graph built through the algorithm we developed is an acyclic, directed graph. Its vertexes and edges come from our two chosen semantic resources.

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 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)

Edges There are 3 kind of edges :

- 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)

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 above) 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 Appendix D 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 the SPARQL requests of Appendix E through the JENA library. Code 1.4 if the resource is an entity and we need to find its class. Code 1.5 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.

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

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 build its generalization tree as presented above.

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).

2.2 Experiments

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

Implementation Explanation

WholeList – WL The two first experiments, WL and 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 code is available on Appendix A and the candidates detection algorithm on Appendix B.
The difference between WL and SL is located line 26 of Code 1.2 : 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 score between the considered candidate and the initial nodes as below :

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

The *score* function can be one of the measures we presented in 1.3, due to computer performances, we had to go with the Shortest Path distance. k is a chosen integer which has for consequence to favor (or not) really close nodes.

SubLists – 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 (1). This has for consequence that the very far nodes don't pollute our results by inserting noise.

DirectNeighbors – DN 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.

WikiLinks 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. 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.
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.

WikiContent The WikiContent expriment 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 Appendix C 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.

Results and Analysis Our experiments have been made using a database of images pulled out the website Flickr[11]. This base is composed of 10261 distinct unique tags, 55600 images, initially annotated with an average number of tags of 17.69, see Appendix F for the top 20 used tags.
For the purpose of our tests, we created a graph based on 1350 images. This gave a graph which contains 5212 nodes and 7552 relations (1612 VIRTUAL, 2339 EQUIV, 3592 PARENT). Thanks to the Gephi software[12], we were able to compute some statistics about our network and we learned that the diameter of our graph is of 20 and the average path length between two nodes is of 6.64. Two methods have been set in order to evaluate the results of our tests launched on 100 images with an initial set of tags of at least 7 items.

First, for the three graph-based experiments, we were able to compute a distance metric between a removed percentage (30%) of the initial tags and the candidates. This distance have been evaluated on a substantial number of test images (100). The average distance results are presented in Table 1. We observe that the DN experiment has better results than the list-based ones. These

[11] https://www.flickr.com/
[12] http://gephi.github.io/

are really close and this can be explained by the fact that their candidates are often the same but in a different order. These results are pretty good since our diameter (ie the longest shortest path between two nodes in our graph) is of 20.

The second method is an user-based evaluation, we asked several individuals to rate the generated candidates regarding the image and the initial tags. They were given a HTML file with 5 images to evaluate (see Appendix G for an example of one) and a CSV file to write their evaluation. The question they had to answer was *Does this tag describe the image ?* and they had to rate each tag on a scale of 1 (*Strongly Agree*) to 5 (*Strongly Disagree*) with an *Undecided* option (3). Each person only evaluated one experiment (WL, SL, DN, WikiLinks or WikiContent) and each experiment was evaluated by different people. We concatenated the results and computed several statistics presented in Table 2.

These results are interesting, it is very clear that the WikiContent experiment is the better one and among the graph-based ones, the DN experiment seems to be the more promising. We also wanted to see if the theory in which the firsts tags are the better ones was correct, and we can't really conclude anything based on these results since no rule seems to apply (only SL and WikiContent follow this pattern). In average, the global rating is around 3, which isn't bad but isn't so good as well. More has to be done in order to enhance our prototype.

3 Perspectives

Our experiments show promising results but it is important to notice that we only propose candidates that have a higher level of abstraction than our initial tags. This is explain by the fact that we "climb" the graph and never travel in the other direction. This can be a way to improve this work : given "general" candidates, it might be interesting to search in their specializations for good tags.
New resources could also be added to this work, like Geonames[13] for instance which provides nice services to detect geographic entities.
It might also be interesting to use other similarities measures, we were limited to the use of the Shortest Path length due to machine's performances.

4 Conclusion

In this study, we reviewed the literature about the semantic web and its technologies as well as the existing approaches for semantic enrichment. We then proposed a prototype which detect candidates tags for an image given an initial set of annotations. Experiments were made and their results were evaluated by two ways and then discussed. Eventually we presented some perspectives about this topic.
This project was a really nice experience. It allowed me to discover the research environment and some new technologies such as the semantic web, the SPARQL language or the graph-based databases. The fact that I worked alone on it was a great challenge to myself and I'm happy to see that some interesting results showed up.

[13] http://www.geonames.org/

WL	SL	DN
3.95	3.90	3.55

Table 1. Average distance between removed tags and candidates

	WL	SL	DN	WikiLinks	WikiContent
Avg. global rating	3.30	3.29	3.09	3.25	2.55
Avg. 1st candidate rating	2.70	2.33	2.73	3.13	1.93
Avg. 2nd candidate rating	3.45	2.96	3.6	2.67	2.19
Avg. 3rd candidate rating	3.15	3.33	3.13	2.53	2.82

Table 2. User-evaluation results

References

1. Berners-Lee, Tim, James Hendler, and Ora Lassila. "The semantic web." Scientific american 284.5 (2001): 28-37.
2. Miller, Eric. "An introduction to the resource description framework." Bulletin of the American Society for Information Science and Technology 25.1 (1998): 15-19.
3. Lehmann, Jens, et al. "DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia." Semantic Web Journal 5 (2014): 1-29.
4. Prud, Eric, and Andy Seaborne. "Sparql query language for rdf." (2006).
5. Mendes, Pablo N., et al. "DBpedia spotlight: shedding light on the web of documents." Proceedings of the 7th International Conference on Semantic Systems. ACM, 2011.
6. Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38.11 (1995): 39-41.
7. Qian, Xueming, and Xian-Sheng Hua. "Graph-cut based tag enrichment." Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011.
8. Cilibrasi, Rudi L., and Paul Vitanyi. "The google similarity distance." Knowledge and Data Engineering, IEEE Transactions on 19.3 (2007): 370-383.
9. Angeletou, Sofia, et al. "Bridging the gap between folksonomies and the semantic web: An experience report." (2007).
10. Zargayouna, Haïfa, and Sylvie Salotti. "Mesure de similarité dans une ontologie pour l'indexation sémantique de documents XML." IC 2004 (2004): 249-260.

Appendix A BFS code

```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 1.1. Java BFS implementation

Appendix B Lists-based candidates detection algorithm

```
1  public void findNewTags(ExecutionEngine engine){
2    // Init
3    HashMap<Node,BFSTraverser> traversers = new HashMap<Node, BFSTraverser>();
4    HashMap<Node, ArrayList<Node>> ilots = new HashMap<Node, ArrayList<Node>>();
5    ArrayList<PairNodeScore> tagsCandidats = new ArrayList<PairNodeScore>();
6    boolean tousParcoursTermines = false;
7    for(Node tagNode : RunExperiments.nodes.values()){
8      traversers.put(tagNode, new BFSTraverser(tagNode));
9      ilots.put(tagNode, new ArrayList<Node>());
10   }
11   // Parcours
12   Transaction tx = RunExperiments.graphDb.beginTx();
13   try {
14     while(tagsCandidats.size()<nbCandidates && !tousParcoursTermines){
15       tousParcoursTermines=true;
16       for(BFSTraverser traverser : traversers.values()){
17         if(traverser.hasNext()){
18           Node currentNode = traverser.next();
19           Node currentRoot = traverser.getRootNode();
20           ilots.get(currentRoot).add(currentNode);
21           if(!containsNode(currentNode, tagsCandidats)){
22             List<Node> occurences = intersection(ilots, currentNode);
23             if(occurences.size()>1){
24               PairNodeScore p = new PairNodeScore(currentNode, 0.0);
25               if(!RunExperiments.nodes.containsKey(p.getLabel())){
26                 double globalScore = computeScores(currentNode,ilots, engine);
27                 p.setScore(globalScore);
28                 tagsCandidats.add(p);
29               }
30             }
31           }
32           if(traverser.hasNext())
33             tousParcoursTermines=false;
34         }
35       }
36     }
37     tx.success();
38   } finally {
39     tx.close();
40   }
41   candidates = tagsCandidats;
42   Collections.sort(candidates);
43 }
```

Code 1.2. Candidates detection algorithm

Appendix C NLP Part-of-speech labels

1. CC Coordinating conjunction
2. CD Cardinal number
3. DT Determiner
4. EX Existential there
5. FW Foreign word
6. IN Preposition or subordinating conjunction
7. JJ Adjective
8. JJR Adjective, comparative
9. JJS Adjective, superlative
10. LS List item marker
11. MD Modal
12. NN Noun, singular or mass
13. NNS Noun, plural
14. NNP Proper noun, singular
15. NNPS Proper noun, plural
16. PDT Predeterminer
17. POS Possessive ending
18. PRP Personal pronoun
19. PRP$ Possessive pronoun
20. RB Adverb
21. RBR Adverb, comparative
22. RBS Adverb, superlative
23. RP Particle
24. SYM Symbol
25. TO to
26. UH Interjection
27. VB Verb, base form
28. VBD Verb, past tense
29. VBG Verb, gerund or present participle
30. VBN Verb, past participle
31. VBP Verb, non 3rd person singular present
32. VBZ Verb, 3rd person singular present
33. WDT Whdeterminer
34. WP Whpronoun
35. WP$ Possessive whpronoun
36. WRB Whadverb

Appendix D DBpedia Spotlight response sample


```
{
  "@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 1.3. Sample response

Appendix E SPARQL requests

```
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 1.4. 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 1.5. Find class' superclass

Appendix F Flickr DB top used tags

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

Appendix G Sample image to evaluate

Initial tags : [plant, flower head, fragility, cornflower, freshness, photography, petal]

New tags

flower flowering genus material term tepal hundred