MEASURING SEMANTIC RELATEDNESS BETWEEN TWO WIKIPEDIA ARTICLES

Dhanya Raghu[1]
[1]Department of Computer Science and Engineering, PES Institute of Technology, Bangalore, India

ABSTRACT

This paper mainly focuses on estimating the relatedness and similarities between any two Wikipedia [1] articles. This paper describes various ways of determining the similarities. We hypothesize that by using some kind of properties of the Wikipedia articles, which can be internal or external, we can estimate the relatedness between Wikipedia articles. Each article is believed to have some kinds of internal properties and some external properties. Internal properties are those which are embedded inside the articles. It can be, for instance, have something to do with the content and text of the articles. External properties are those which are deduced or inferred from the articles. It can be, for example, the topic of the articles or even the closest distance between the two articles when plotted in a graph or in a category hierarchy. External properties include the properties associated with individual articles like topics (as mentioned), categories of the articles. Other techniques which are relevant when comparing the Wikipedia articles are cosine similarity, Jaccard similarity measure etc.

KEYWORDS

Wikipedia, semantic, relatedness, similarities

1. INTRODUCTION

Wikipedia is an information repository that is created by a community of users who bring in their knowledge and create articles. Wikipedia, an important source of information: These articles follow a certain schema, and the users can choose to provide additional information which is optional. Wikipedia is an authentic source of information and hence is consulted by many on a regular basis. People consult Wikipedia to gather information about topics of interest to them, and they can choose to edit and add additional information if they so desire.

Motivation: Wikipedia is a community database created by millions of users using a web 2.0 mechanism. Wikipedia contains a huge database with millions of pages out of which many are related to each other. The relationship between the wiki articles plays an important role in a further analysis of Wikipedia, it leads to semantic analysis and ultimately building a recommendation engine. This kind of relationship determining is used in online shopping websites to recommend related items the customer wants to buy.

Users login to Wikipedia and create pages of interest to them. When they create these pages which are referred to as “articles” in Wikipedia terminology, they need to identify related articles. The articles that are related to a given article contains information which is related to the current article. Users search for articles using the search option supported in Wikipedia, and they are interested in identifying related articles.

Additionally, many of the recommendation engines use Wikipedia as concept database, and they rely on the information networking provided in Wikipedia to make recommendations. For example, when a user is looking for information about a certain brand of paint, he/she would find it helpful if we could provide a set of related paints to him as well. These recommendation engines are core to several of the software based marketplaces, and they thrive on providing useful recommendations to their users. All the above-mentioned uses require a way to relate articles.

2. IMPLEMENTATIONS IN THIS PAPER

2.1 Dhanya (myMethod)

All wiki articles must belong to at least one category. Using the entire Wikipedia, input any two articles, determine the list of categories for each of them, and if they have one or more common categories, they are considered as related. For any particular category, the category hierarchy is built as a graph where the category hierarchy graph is traversed upwards towards a common parent until found. That article with the category name is the root and all its children are updated in the graph at each level and the relatedness between any two articles in the category hierarchy is calculated by determining the distance between the two articles (basically the number of edges) and a relatedness index is found. This relatedness index, however, cannot be compared with the other similarity measures because of the different methods applied for finding each similarity.

2.2 Cosine Similarity [3]:

It makes use of the textual content of any two Wikipedia articles and determines the relatedness in the form of cosine measure which tends to 1 as the relatedness increases. It uses the vector space model of documents modelled as vectors with Term Frequency (TF) and Inverse Document Frequency (IDF) counts.

2.3 Jaccard Similarity [4]:

Usually, this uses the common “bag of words” model, which is simplistic, but is sufficient for many applications. For any two datasets A and B, The Jaccard similarity is defined as JS(A, B) = |A union B|/ |A intersection B|. Here, A and B are two wiki articles. A more general approach is to shingle the document. This takes consecutive words and groups them as a single object. A k-shingle is a consecutive set of k words. So the set of all 1-shingles is exactly the bag of words model. An alternative name to k-shingle is a k-gram.

2.4 WikiMiner:

An online tool to estimate the similarities of wiki articles.
All of the above implementations have been experimented with selected article pairs given in the **Wikipedia standard dataset WordSim353** and the results are tabulated.

3. IMPLEMENTATION OF METHOD “DHANYA” (MYMETHOD)

The first method of determining the similarities of Wikipedia articles uses a graphical approach.
The wiki graph is depicted as G<V,E> where V=wiki article URL. Vi=wiki article(i) URL and the edge E =1 if between any two vertices Vi and Vj, there exists a connection. Basically, it lists the categories of both the pages and declares that they are related if they have one or more same categories. More the common categories more is the relatedness.

Open source tools used:

A. Jgrapht [5]: This tool provides the basic graph libraries needed for building the wiki graph and adding vertices, edges and also to find the shortest path between any two vertices. The vertices are basically URL objects and we have built a directed graph.

B. Httpclient [7]: This is a tool used to obtain the wiki articles in the form of XML strings from the HTTP server using the httpget() method.The content of the response entity is the wiki article that we need.

C. Wikixmlj [6]: This is an XML parser that parses the xml strings obtained by the http client and provides methods like getCategory() and getLinks() that we use to extract the categories and subcategories.

3.1 Algorithm:

Prerequisite: Availability of a Wiki graph in memory for performing the relatedness operations.

The procedure BuildWikiGraph() described loads the entire Wikipedia articles into an in-memory graph data structure.

- *FindIfRelated*: Finds out if the given two articles are related
- *FindRelatednessMeasure*: Find out how strongly the given two articles are related.

```
Wikigraph BuildWikiGraph()
{
Read the Wiki category
read the subcategories of the given category
for each subcategory   {
        read articles in each subcategory
        Update wiki graph
}
return Wikigraph
}

Algorithm FindIfRelated (article a1, article a2){
{
 read two articles (a1 and a2)
If a1.category == a2.category then the two articles are related as they belong to the same category
else they are not related
}

Algorithm FindRelatednessMeasure(article a1, article a2)
{
Find the distance between a1 and the category to which it belongs
d1 = distance(category, a1)
d2 = distance(category, a2)
relatednessMeasure = d1 + d2
}
```

Each wiki article is considered as a vertex in the wiki graph and the edges represent that the two wiki articles are related (connected). To find out whether two articles are related:

Input: Two wiki article names as strings
Output: print all the common categories thus proving that the two articles are related
else print: not related.

3.2 Steps:

Httpclient has the httpget method to obtain the wiki articles in html format as a stream of bytes. We can view the data retrieved from the http server by the getEntity() method of the response entity. But, we want the entities in xml format because wikixmlj is an XML parser. So, we use the httpget method to obtain the wiki articles as special export which is the wiki articles in xml format.

The xml strings of two wiki articles are fed as an input to the wikixmlj, which parses these xml strings and provides methods to extract links and categories from the parsed objects. We obtain all the categories of the two wiki articles in two string arrays. We, then use string comparison and print out all the common categories if any. The existence of a common category thus proves that the articles are related.

3.2.1 Building the wiki graph:

Input: one category
We again obtain this wiki page as an xml string. We use wikixmlj to parse the Wiki page and extract all the sub-categories by calling getLinks() method of the wikixmlj.

Once we get all the sub-categories in an ArrayList, for each sub-category, we extract its subcategories and so on. The wiki graph is constructed with the main category as the root, each sub-category is the parent of all the sub-categories extracted from it. The next part is to compute the distance between any two vertices. Since this wiki graph is a directed graph, it is possible to traverse the graph in only one direction which depends on the direction in which the edges are added.

So, here, we can traverse from the root towards its children vertices. So, to compute the total length of the path between any two vertices v1 and v2, we compute the distance between the root and v1 plus the distance between the root and v2.

Then, the relativeness index is found by the following formula:

```
for(int i=1;i<=pathlength;i++)
{
relativenessIndex = relativenessIndex + (1.0/(Math.pow(2,i)));
}
```

This formula is got by the following reasoning:

Each edge in the path between the two vertices is given a weight which starts from one vertex, and the weight is reduced by half at each edge in the path.

4. OTHER METHOD IMPLEMENTATIONS

4.1 Cosine Similarity:

It basically calculates the cosine product between two document vectors.

Implementation: Initially, the documents between which the cosine similarity has to be found are listed in a folder. Each document is inputted into the algorithm and all the contents of the input document are read. Then, the string builder class is invoked and the document is parsed, meaning the document is tokenized into individual terms and the duplicate terms are eliminated. Calculation of the tf-idf matrix: Term Frequency also known as TF measures the number of times a term (word) occurs in a document. In reality, each document will be of different size. On a large document, the frequency of the terms will be much higher than the smaller ones. Hence, we need to **normalize** the document based on its size. So, divide the term frequency by the total number of terms. Then, the inverse document frequency matrix is found. The documents, sometimes have some terms occurring repeatedly, but these terms might not actually be relevant in determining the similarity. On the other hand, some terms, though are less frequent, may be relevant in determining the similarity. So, there is a need to weigh up the relevance of these less frequent terms and weigh down the relevance of high-frequency irrelevant terms. Logarithms are useful here. Consider some documents with the term computer in them.

Ex: IDF(**computer**) = 1 + log(Total Number Of Documents / Number Of Documents with term **computer** in it).

The next step is to multiply the tf and idf values and these products become the parameters for finding the cosine similarity.

4.2 Jaccard Similarity:

This makes use of the k-shinglings technique and is the ratio of the common shinglings between the documents and the total number of distinct shinglings in the two documents put together. A bag of words is 1-shingle. The k-shingle approach is better. K-shingle is a sequence of k characters. The formula for jaccard similarity: J(A, B) = |A union B| / |A intersection B|.

5. EXPERIMENTS AND RESULTS:

Table.1 showing the tabulated results of all similarity measures.

Word 1	Word 2					
			1	2	3	4
		Human	myMethod	Cosine	Jaccard	wikiMiner
tiger	cat	0.705	0.9375	0.880935305	0.556294569	0.603
tiger	tiger	1	1	0.99999999	1	0.82
computer	keyboard	0.7355	0.96875	0.569367	0.1334525	0.66
computer	internet	0.7311	0.9375	0.8608416	0.480286	0.56
plane	car	0.53	0.75	0.6367835	0.1756188	0.48
stock	jaguar	-0.00888	0.96875	0.805172	0.35298	0.32
telephone	communication	0.7222	0.9375	0.719259	0.4604139	0.61
football	soccer	0.9	0.75	0.9291183	0.474487451	0.86
physics	chemistry	0.7	0.9375	0.5120144	0.81396668	0.77
drug	abuse	0.65	0.875	0.539589	0.33099	0.73
money	cash	0.90555	0.75	0.869559	0.32376	0
professor	cucumber	-0.07666	0	0.69789	0.3999499	0.43
king	cabbage	-0.08555	0	0.7793045	0.392904	0.23
vodka	gin	0.8	0.9375	0.8657988	0.4694249	0.78
tiger	cat					

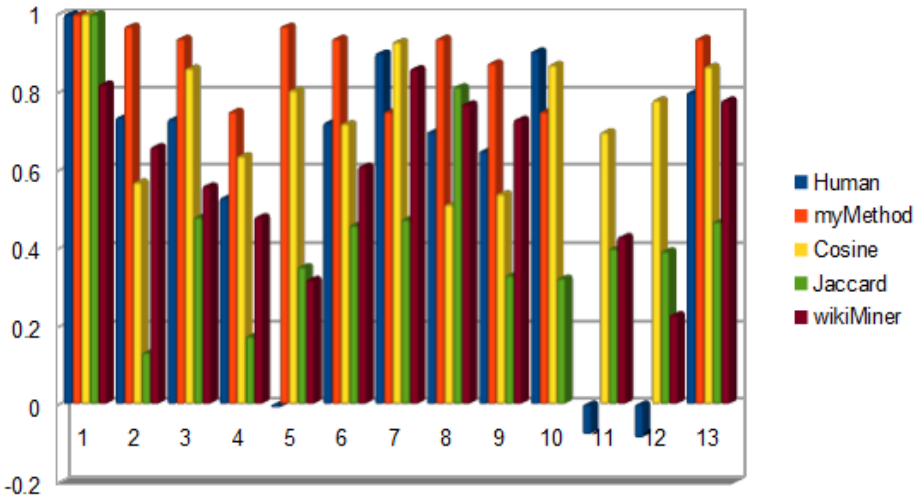

Figure.1 showing the results of the various methods of similarity measures for Wikipedia Articles.

Example output for tiger and cat wiki articles: It finds the shortest path between tiger article and the common category page (root) and the same is repeated for wiki article cat. Then, these distances are added and relatedness index is calculated. For instance, consider wiki articles- tiger and cat.

Shortest path from
http://en.wikipedia.org/wiki/Lists_of_animals to http://en.wikipedia.org/wiki/tiger

size of the path list is 2

[(http://en.wikipedia.org/wiki/Lists_of_animals:
http://en.wikipedia.org/wiki/endangered_species),
(http://en.wikipedia.org/wiki/endangered_species : http://en.wikipedia.org/wiki/tiger)]
Shortest path from
http://en.wikipedia.org/wiki/Lists_of_animals to http://en.wikipedia.org/wiki/Cat

size of the path list is 2

[(http://en.wikipedia.org/wiki/Lists_of_animals:
http://en.wikipedia.org/wiki/List_of_domesticated_animals
(http://en.wikipedia.org/wiki/List_of_domesticated_animals : http://en.wikipedia.org/wiki/Cat)]

Relativeness Index is 0.9375

6. EXPLANATION OF THE RESULTS:

Human*:* It is the standard, accepted and a precise value of similarity obtained from the wikiSim 353 dataset. So, it is used as a standard to compare the output of other similarity measures.

Dhanya*:* This method is the first method explained in the paper. The values of similarity obtained from this method cannot be compared with the other measure because, it finds the distance between two wiki article nodes in a category hierarchy. It adds up all the weights of the edges in the shortest path between the two nodes and gives the total distance between the two nodes. So,

lesser the distance, more the similarity, unlike the other methods where greater the measure, greater is the similarity.

Cosine*:* This is the output of the cosine similarity measure. It is mostly accurate and close to the human values in most of the resulting output.

Jaccard*:* The output of jaccard similarity measure. This is less accurate than cosine similarity.
WikiMiner*:* This is also accurate, but not all articles are recognised by this online tool.

7. FUTURE WORK:

1. Import the entire Wikipedia graph into Neo4j and run the shortest path algorithm which gives a possible measure of similarity.
2. Using Apache Lucene to find probabilistic similarity measures between the articles.
3. Using R, perform topic modeling and use the topics to determine the similarity.

8. ACKNOWLEDGEMENT:

I, Dhanya Raghu, would sincerely like to thank my project guide, Dr. Raghu A, for guiding this project into a successful one.

9. CITATIONS:

1. http://wikipedia.org
2. Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi, Wolfman, and Eytan Ruppin, "Placing Search in Context: The Concept Revisited", ACM Transactions on Information Systems, 20(1):116-131, January 2002
3. https://en.wikipedia.org/wiki/Jaccard_index
4. https://en.wikipedia.org/wiki/Cosine_similarity
5. http://jgrapht.org/javadoc/
6. https://code.google.com/p/wikixmlj/
7. http://hc.apache.org/httpclient-3.x/userguide.html
8. https://www.aaai.org/ **Papers** /AAAI/2006/AAAI06-223.pdf
9. www.cs.technion.ac.il/~gabr/**papers**/ijcai-2007-sim.pdf
10. www.cms.waikato.ac.nz/.../ **papers** /08-DM-IHW-Semantic_ **relatedness** .pdf

Author

I am a Computer Science and Engineering undergraduate at PES Institute of Technology, Bangalore, India. My research interests in Computer Science include Data and Graph Mining, Cloud computing and Machine Learning.