Semantically Matching Between Documents Using Term Co-Occurrence Graph*

[Project Report]

Beethika Tripathi
CS15S004
beethikat@yahoo.co.in

Hitesh Gupta
CS15S039
hitesh3011@gmail.com

ABSTRACT

Documents like legal judgments, news articles, movie plots, patents might be lexically different but semantically similar or vice versa. Hence, lexical methods to identify the semantically similarity between such documents fails many times. Thus, we need to have approaches which use features beyond the just lexical features of the given documents. To solve the problem, we propose a approach called semantic matching.

In the current work, we attempt to solve two kinds of semantic matching problems namely – identification of semantic similarity (identifying the topical similarity score between the given pairs of documents) and semantic information retrieval (retrieving topical similar documents for a given document from a corpus). We take a term co-occurrence based approach to solve these problems. The term co-occurrence graph (TCG) acts as the knowledge base [1].

To identify the semantic similarity score between a pair of documents, we initially extract important terms from each document. Then we expand them using the knowledge base (TCG). We identify the common topics in the expanded space and the similarity between them. We evaluate our approach by comparing our results with the output of LSA.

General Terms

Documentation, Legal Aspects

Keywords

Cognitive models, Co-occurrence, Text mining, Syncretism

1. INTRODUCTION

Many times news articles contain events which are similar to the events which have happened in the past. However, as the terms used to in these two articles may be different, the pure lexical feature based similarity mining approaches many times fail to identify the similarity. For example, consider the following three text snippets.

1. "Forty-five people were killed after a private bus carrying them rammed into a culvert and its fuel tank caught fire in Mahabubnagar district of Andhra Pradesh early Wednesday morning. The Jabbar Travels' air-conditioned Volvo bus began its journey at 11pm on Tuesday from Kalasipalyam area in Bengaluru to Hyderabad and picked up 50 passengers along the way."
2. "Bankapura is a panchayat town in Haveri district in the state of Karnataka, India. It is in Shiggaon taluk, is just 2.5 km from the Pune-Bangalore national highway NH4, 22 km from Haveri town. Bankapura is about 45 km from Hubli-Dharwad. An historical site, Bankapura is famous for the Nagareshwara temple, Bankapura fort, The Bankapura Peacock Sanctuary. Baada, the birthplace of Kanaka Dasa is near to Bankapura."
3. "At least seven persons were charred to death and 21 others injured when a Mumbai-bound bus caught fire after hitting a road divider in Haveri district of Karnataka early this morning. According to reports, the mishap took place at around 3.20 a.m. near Kunimelli bridge, Bankapura Cross in Haveri district of central Karnataka. The bus, operated by Bangalore-based National Travels, was on its way from Bangalore to Mumbai." [3]

If need to identify a semantically similar pair of documents from this corpus using the state of the art methods like cosine similarity or latent semantic indexing based cosine similarity, then we find that they identify the snippet 2 and 3 as the similar documents as shown in the tables 1 and 2.

Text 1	Text 2	Cosine Similarity
Snippet 1	Snippet 2	0.37
Snippet 1	Snippet 3	0.069
Snippet 2	Snippet 3	0.019

Table 1: Similarity measurement between the pairs of text snippets using direct cosine measure

Text 1	Text 2	LSA Based Similarity
Snippet 1	Snippet 2	0.23
Snippet 1	Snippet 3	0.036
Snippet 2	Snippet 3	0.007

Table 2: Similarity measurement between the pairs of text snippets using the latent semantic indexing.

However, if a person does the same task, then we claim that she identifies snippets 1 and 3 as similar. This disparity in the decisions of a state of the art similarity

*As part of project, CS6740, Jul - Nov 2015

algorithm and human being is the due to the way they perform the matching. The state of the art algorithms match the documents based on the evident features (terms in this case) while human beings match documents using the knowledge they have.

Here, we take a step by step approach to topically compare the documents. In the first step, we identify the terms which are highly relevant to given documents. Then we expand the documents by expanding these terms using a term co-occurrence knowledge base. Finally, we identify the common topics between the two expanded documents and compare their similarity.

2. BASELINE

2.1. LSA (Latent Semantic Analysis)

LSA is one of the most powerful techniques developed for finding similarity between documents. It assumes that words that are close in meaning will occur in similar pieces of text. It uses a matrix containing word counts per paragraph (rows represent unique words and columns represents each paragraph).The matrix is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) which is used to reduce the number of rows while preserving the similarity structure among columns. Words are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words. [5]

2.1. Problems with LSA –

- We cannot use LSA directly to find aspect based similarities like – story similarity and expression similarity.
- It works only on a local knowledge (patterns) and does not consider any external knowledge source to identify the similarity between the documents.
- For a document retrieval tasks, LSA reduces the dimensionality of the complete document and stores it as the index. If a new document needs to be added to the index, there is no way of partially changing the already available matrices apart from running the complete SVD again which is highly expensive and hence not practical. [2]

We used an online implementation of LSA to compare it with our results.

2.2. Textual Similarity

Lexical similarity is a measure of the degree to which the word sets of two given languages are similar. A lexical similarity of 1 (or 100%) would mean a total overlap between vocabularies, whereas 0 means there are no common words.

It is the most simple and oldest way of comparing two documents. We just check how many words are common between the two documents and output the result. It does not consider meaning or semantic for measuring similarity, hence end up giving inaccurate results.

3. SEMANTIC MATCHING

We define ‘syncretism’ as the deeper similarity in the meaning of the concepts under consideration. We can categories document matching into two types namely lexical matching and semantic matching. The lexical matching looks purely at lexical features of documents, runs some transformations on them to compare them. However, the semantic matching intends to identify the topics discussed in the document and matches documents based on the topics.

There have been many attempts like LSA (which attempt to use the lexical features across documents to identify similarity) and many ontology based document similarity mining approaches. They do not consider the two major factors in semantic matching. They are Noise due to proper nouns and equal importance of nouns and verbs.

3.1. Noise due to proper nouns

The semantic matching deals with matching the deeper meanings between documents under considerations. A part of deeper meaning of the documents gets derived from the actors of global context and the activities involved in the document. All the nouns except the proper-nouns behave as the actors with global context. The proper nouns have very local context. Similarly, all verbs represent the activities mentioned in a document. Hence, they contribute to the deeper meaning of the document. For example consider the statements – 1. Federer won the grand-slam. 2. Nadal won the grand-slam. 3. Djokovic lost the grand-slam 4. Nadal won a lottery. We claim that the first two are semantically similar statements than any other pair. This shows that local actors (in current case Federer and Nadal) are less significant compared to the global context knowledge element like ‘grand-slam’ and the action of ‘winning’. [3]

3.2 Verbs are as important as nouns

As discussed, nouns (except proper nouns) and verbs are the major contributors for the deep meaning.

We used a term co-occurrence graph as the knowledge to perform semantic matching. This can be created using any encyclopedic text corpus for general purpose semantic matching. If we want the matching to be performed using a domain specific knowledge, then we can use a domain specific corpus like reviews corpus of Flipkart, biological corpus, news specific corpus etc.

4. BASIC FLOW

The basic-flow of the document retrieval process has been stepwise described in the following section :

4.1. Term co-occurrence graph (TCG)

Formally, the undirected co-occurrence graph G is defined as, G = (T,C,w), where T is the set of all terms in the corpus and C is the set of all pair-wise co-occurrences across terms inferred from the episodes in the corpus. The function ‘w’ indicates the corresponding co-occurrence count in the corpus. This is the number of occurrence contexts across all documents in the corpus, that contain both elements of a co-occurring pair.

4.1.1. Dataset used :

Whole Clean Wikipedia text (as of Jan-2013)
Size = 8.21GB
No. of documents = 37804209
No. of unique terms = 708219 (after removing all stop-words and storing only adjectives, nouns or verbs)

4.1.2. Point Mutual Information(PMI)

It is the measure of association between the pair of words.
PMI(y,z) = Count(y,z)*N / (Count(i,z)*Count(y,i))
where,
Count(y,z) --> The number of co-occurrences of y and z
Count(i,z) --> The number of occurrences of z
Count(y,i) --> The number of occurrences of y
N --> The sample size.

The dataset is traversed once, and PMI value for each pair of words is saved in a file which we use as our trained model.
Using a dictionary (a maps of keyword strings to the integer index of the node of the graph) we change the strings to integer values which reduces the file size drastically.
Also, we prune away those edges whose weight is < P, i.e. we don't consider an edge between the two nodes whose probability of occurring together is less than P.
For each node, we store its edges in a sorted order so that retrieval of the top-k edges doesn't take time.

4.2. Identification of locally important terms

Each document d_a is written to discuss some important topics. We need to identify the important terms from the document. Each document d_a is made up of a set of unique non-stopword terms. We can represent them as a set K_a. Formally,

$$K_a = \{ t_1^a, t_2^a, \ldots\ldots, t_k^a \}, \text{ where } |K_a| > 0$$

As there are two documents d_a and d_b, there will be two sets K_a and K_b are created.
We identify the nouns and verbs in the document separately with their frequencies.
In these, we remove all the stop words using a stop word list. We simplistically assume that the frequency of a word represents its importance in the document. We normalize the frequencies of nouns and frequencies of verbs independently and merge them. These normalized scores are called local importance values for the k terms in K_a. It is represented as (Y_a). Formally,

$$Y_a = \{ c_1^a, c_2^a, \ldots\ldots, c_k^a \}$$

where c_i^a is the importance score of term t_i^a .

- Part-Of-Speech tagger, a Java library by Stanford NLP has been used to identify which word is an adjective, noun or verb. [4]
- Importance Score = Frequency of the keyword in document / Frequency of all keyword in the document

4.3. Document Expansion

We expand the locally identified important terms on the given general knowledge to get a larger global context. The global context contains the other terms which are related to the locally identified terms.

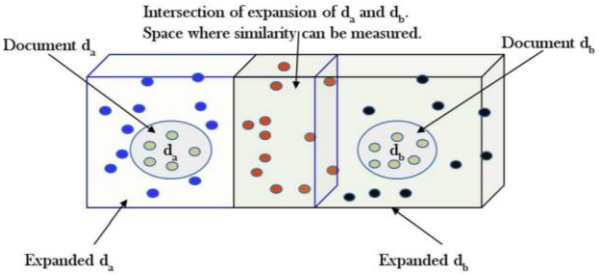


Figure 1: The block diagram of semantic matching. Each document is made up of terms. These terms are expanded to get the document expansion. If the documents are similar, then there will be an intersection in their expansion.

For document d_a, we create the semantic context of the closure of the terms in K_a on the the general knowledge corpus G (Wikipedia co-occurrence graph).
Formally, $U(K_a^*)$ can be defined as,

$$U(K_a^*) = \{S_a; E_a; w_a\}$$

where, $S_a = \{ s_1^a, s_2^a, \ldots\ldots, s_m^a \}$ is the set of all terms in the semantic context $U(K_a^*)$.
The term E_a is the set of edges whose both end points exist in S_a. The term w_a is a function which assigns edge weight to the edges.
For each of the k terms in K_a, we run a separate random walk on $U(K_a^*)$ as explained in the following algorithm :

Data: Co-occurrence graph G, documents to be matched
Result : Similarity value for the two documents.

1. **for** all documents
   extract all keywords from doc[i] removing stopwords and proper nouns
   for all keywords in doc[i]
   1. Search the dictionary for the keyword and map it to the node number.
   2. Go to the corresponding node in the term co-occurrence graph
   3. Calculate K = importance_score * Total_Edges
   4. Retrieve the top-K edges of that node and add it to the hashmap of doc[i]

   end
   Add the HashMap to the List.
   end
2. Access the List and find the HashMap of keywords for the two documents
3. Calculate the similarities using formulaes.
4. Return similarity results.

Algorithm 1: For calculating similarity using graph expansion for the keywords of the matched documents.

4.4. Similarity Calculation

We have used 3 types of similarity measures

4.4.1. Node Jaccard :

The node Jaccard similarity coffecient ($J_{G1,G2}$) between two graphs $G_1 = (T_1, C_1, w_2)$ and $G_2 = (T_2, C_2, w_2)$ is the ratio of the number nodes in the intersection of T_1 and T_2 to number of nodes in union of T_1 and T_2. Formally

$(J_{G1,G2}) = |T_1 \cap T_2| \ / \ (T_1 \cup T_2)$
= Size (Intersection of Top-K keywords of doc1 n doc2) / Size (Union of Top-K keywords of doc1 and doc2)

4.4.2. Edge Jaccard :

Numerator = Numerator +
min(Count_Of_Node(doc1),Count_Of_Node(doc2)) ;
if the nodes intersect;
= Numerator + 0 ;
if the nodes doesn't intersect;
Denominator = Denominator +
max(Count_Of_Node(doc1) ,Count_Of_Node(doc2));
if the nodes intersect;
= Denominator +
Count_Of_Node(doc1) + Count_Of_Node(doc2);
if the nodes doesn't intersect;

Edge Jaccard = Numerator / Denominator;

4.4.3. Cosine Similarity :

Term Frequency (TF) = Frequency of a word in the document / Total frequency of all the words in the document
Inverse Document Frequency (IDF) =
1 + log(Total no. of document / No. of document that contains that word) [3]
TFIDF = TF * IDF
For all words in doc 1 and 2, we find TFIDF and obtain nX1 vectors V_1 and V_2

Cosine Similarity : $V_1 . V_2 / |V_1| . |V_2|$

5. EXPERIMENTAL ANALYSIS

We collected 30 test cases for various sources on internet. They were divided into three themes - movies, news and encyclopedic articles.
There were 12 movie test cases, 9 news test cases and 9 encyclopedic test cases. Each test case consisted of three text snippets. Two of them were similar in semantics to each other.
The user evaluation was carried out to identify which out of LSA or semantic is better.
1. First set of users were asked whether the given pair of documents is the most similar pair out of the three snippets they have seen.
2. Second set of users were asked to identify the most similar pair of document in the pairs (document 1, document 2), (document 1, document 3) and (document 2, document 3) without telling which result is LSA's and which is our result
To avoid the bias in the reading order influencing the decision, we had randomly ordered the text snippets for each user.
Once the user evaluation results were available, we provided the same test cases as the input for the semantic matching algorithm, LSA based similarity and cosine based similarity algorithms.

5.1. Setup for experiment :

32GB RAM
Quad-Core PC each operating at 3.2 GHz
64bit Linux machine.
2TB HDD

The results of the semantic matching algorithm are shown in tables :

News1	Movie1	Encyclo1
News3 (0.70)	Movie2 (0.86)	Encyclo5 (0.85)
News4 (0.62)	Encyclo2 (0.74)	Encyclo3 (0.72)
Movie2(0.52)	Movie3 (0.71)	Encyclo2 (0.72)
News2 (0.48)	Movie6 (0.67)	News4 (0.67)

Table 3 : These are the top-4 results for the first document for every theme using our technique.
We can see that news document is similar to news, movies to movies and encyclopedia to encyclopedia documents. Hence our results our accurate. The similarity values are written in parenthesis.

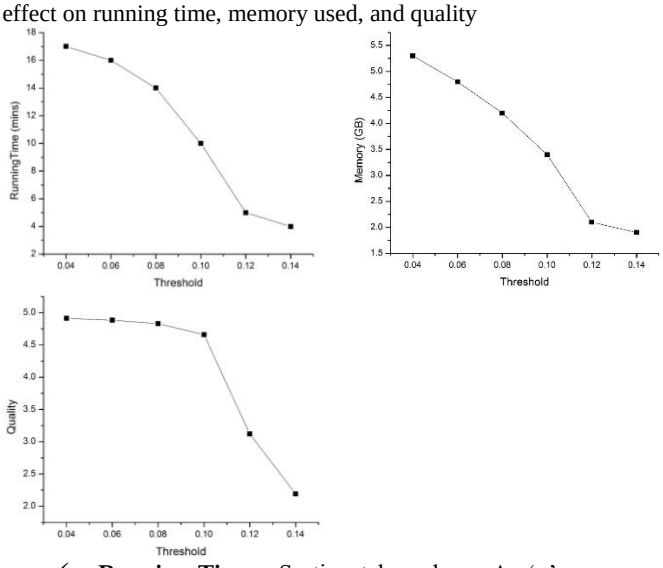
The results of the LSA algorithm are shown in figure :

News1	Movie1	Encyclo1
News4 (0.75)	Movie2 (0.84)	Encyclo5 (0.86)
News3 (0.68)	Encyclo5 (0.76)	News4 (0.81)
Encyclo2 (0.57)	News3 (0.71)	Encyclo2 (0.76)
Movies2 (0.48)	Movie6 (0.63)	News3 (0.71)

Table 4 : These are the top-4 results for the first document for every theme using LSA

The user's choices showed that our results were more accurate than LSA. Also we solved the shortcomings of the LSA technique for semantic matching of documents.

While storing the edges in the graph, we pruned away those edges whose weight is less than 'P'
This means that those two node which has such a low weight edge between them, they just cannot occur together.
While varying the value of this threshold, we check its effect on running time, memory used, and quality



- ✓ **Running Time** : Sorting takes nlogn. As 'n' (number of edges) decrease, our method will take lesser time. Hence it is obvious that on not taking all the edges, it will decrease the time taken. There is a sudden decrease at threshold 0.12 because lots of edges are getting pruned away as most of them has weight between 0.10 and 0.12
- ✓ **Memory :** As the number of edges decrease (by increasing threshold), lesser memory will be required to store and hence less memory is used

at big thresholds which is evident from the graph. As most of the edges has weight between 0.10 and 0.12 , there is a sudden decrease in use of memory at threshold 0.12 because lots of edges are getting pruned away

- ✓ **Quality :** If the number of edges are being decreased, the quality of our system decreases since the expansion in our knowledge base will not be accurate. We notice that quality is not much affected till 0.10 threshold because the edges having weight less than 0.10 were already not much considerable for the results.
- ✓ We also tried running cricket focussed documents over a Biological Knowledge base and it gave completely irrelevant results, which was expected since such knowledge graph is useless for the unrelated documents as expansion does not produce useful learning.

6. APPLICATIONS

1. Identification of similar judiciary cases, given a new case
2. Identification of similar patents, given a new patent
3. Looking for similar medical cases to understand the treatment.
4. Retrieval of a similar product using its description or reviews on an e-commerce website
5. Relating with the previous news article, given a new news.
6. Comparing movie plots.
7. Predicting what user actually wants so that he doesn't end up surfing through a lot of useless documents. [1]

7. CONCLUSION

- We solved these two kinds of semantic matching problems namely
  - ➢ Identification of semantic similarity (identifying the similarity score between the given pairs of documents)
  - ➢ Semantic information retrieval (retrieving similar documents for a given document from a corpus).
- Also, our experiment results shows that cosine similarity of the semantic matching is the best technique comparing it with the ground truth.
- We tried running it with documents that are totally irrelevant to the knowledge base and as expected, we got inaccurate results.
- So, always try to use a general-most possible knowledge base to teach the computer.

8. FEEDBACK FROM FLIPKART

- They liked the idea of matching two documents semantically using a knowledge graph i.e. by making the machine learn.
- They wanted us to run our system over review documents and product description documents.
- Since the most similar review documents were fetched correctly by our system, they were highly impressed by the result and wanted us to continue the work and send them a URL of a working application.
- They wanted us to use a specific knowledge base of review documents to train the system, so that more accurate results can be obtained.
- They wanted us to properly document our idea so that they could suggest it in their company for retrieving the most similar reviews to a given one, so that user do not need to read every review.

9. REFERENCES

[1] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, R. A. Harshman, Indexing by latent semantic analysis, JAsIs 41 (6) (1990) 391–407.

[2] C. H. Ding, A similarity-based probability model for latent semantic indexing, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 1999, pp. 58–65.

[3] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis., in: IJCAI, Vol. 7, 2007, pp. 1606–1611.

[4] C. J. van Rijsbergen, A theoretical basis for the use of co-occurrence data in information retrieval, Journal of documentation 33 (2) (1977) 106–119.

[5] L. Huang, D. Milne, E. Frank, I. H. Witten, Learning a concept-based document similarity measure, Journal of the American Society for Information Science and Technology 63 (8) (2012) 1593–1608.