

# Semantic Matching Between Documents Using Term Co-Occurrence Graph

**PRESENTED BY:**

**Beethika Tripathi (CSI5S004)**

**Hitesh Gupta (CSI5S039)**

**4<sup>th</sup> November 2015**



# Motivation



- Humans Understand Semantics!
- Many times news articles contain events which are similar to the events which have happened in the past. However, as the terms used in these two articles may be different, the pure lexical feature based similarity mining approach fails to identify the similarity many times.
- Hence, there is a need of unsupervised approaches which use features beyond the terms present in the documents to match them.

# Applications

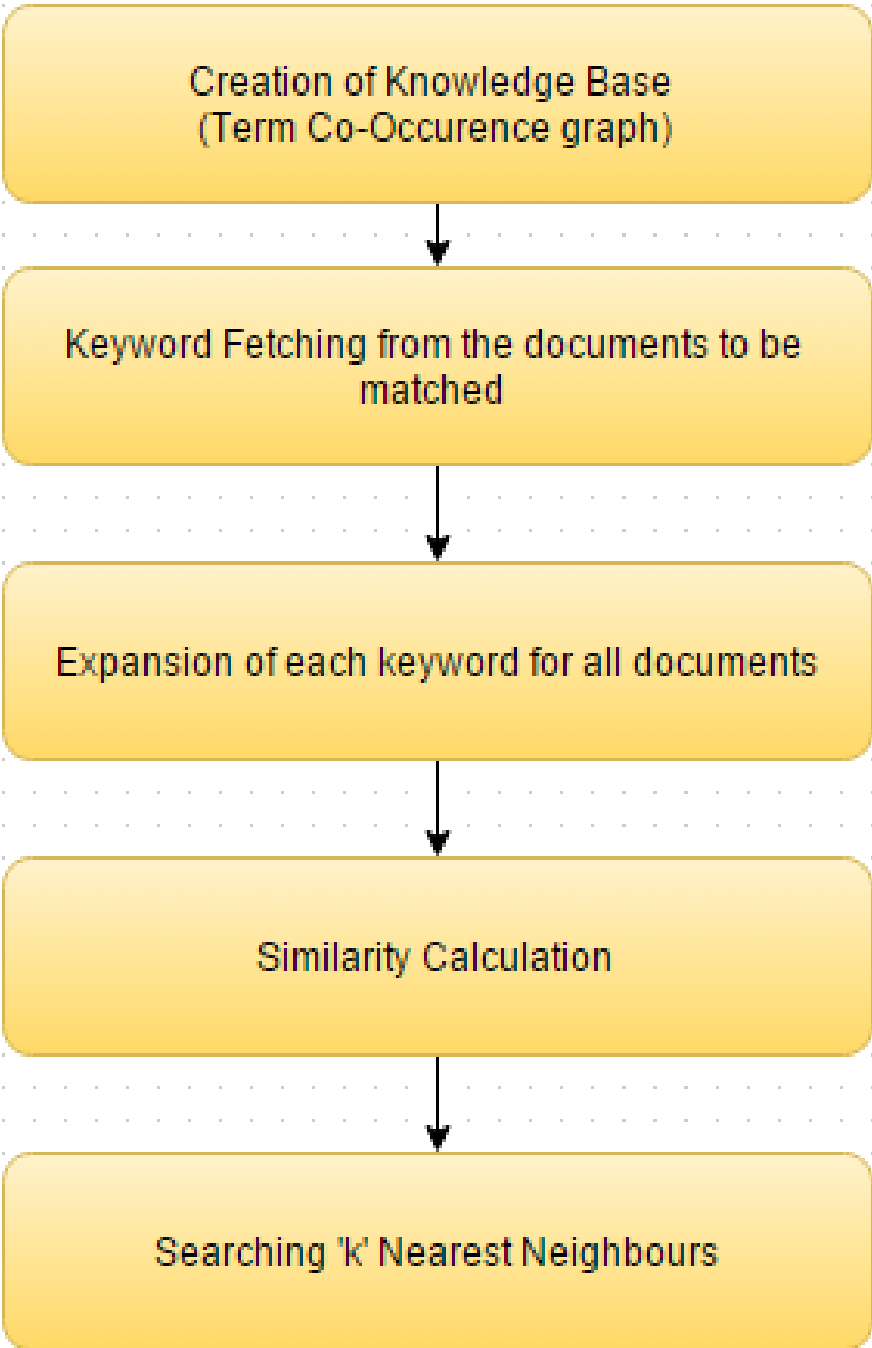
- Identification of similar judiciary cases, given a new case
- Identification of similar patents, given a new patent.
- Looking for similar medical cases to understand the treatment.
- Retrieval of a similar product using its description or reviews on an e-commerce website
- Relating with the previous news article, given a new news.
- Comparing movie plots.
- The literal terms in a user's query may not match those of a relevant document. In addition, most words have multiple meanings. Hence the user end up surfing through a lot of useless documents.

# Problem Statement

In this project, we attempt to solve two kinds of semantic similarity problems namely

- Identification of semantic similarity (identifying the similarity score between the given pairs of documents)
- Semantically similar information retrieval (retrieving similar documents for a given document from a corpus).

As we deal with the study of semantics, we define it as the deeper **similarity in the meaning** of the concepts under consideration.



```
graph TD; A[Creation of Knowledge Base  
(Term Co-Occurrence graph)] --> B[Keyword Fetching from the documents to be  
matched]; B --> C[Expansion of each keyword for all documents]; C --> D[Similarity Calculation]; D --> E[Searching 'k' Nearest Neighbours];
```

Creation of Knowledge Base  
(Term Co-Occurrence graph)

Keyword Fetching from the documents to be  
matched

Expansion of each keyword for all documents

Similarity Calculation

Searching 'k' Nearest Neighbours

# Basic Flow of our Project

Only for retrieving Top-k most similar documents to the specified document, we perform the last step, otherwise we just calculate similarity measure between the two given documents.

# Step I : Pre-processing - Creating Knowledge Base

- Dataset used : Whole Clean Wikipedia text (as of Jan 2013)  
Size = 8.21 GB  
No. of documents = 37804209  
No. of unique terms = 708219

The stop-words were already removed from this dataset and we store only the keywords which are either adjectives , nouns(except proper nouns) or verbs.

- $PMI(y,z) = \text{Count}(y,z) * N / (\text{Count}(i,z) * \text{Count}(y,i))$  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
- Point Mutual Information(PMI) is the measure of association between the pair of words.
- 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.

# Why only adjectives , nouns and verbs?

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.
- First two are semantically similar statements than any other pair.
  - 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.
  - Hence, considering proper nouns equal to other nouns is not desirable

# Processing done on Graph

- 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.

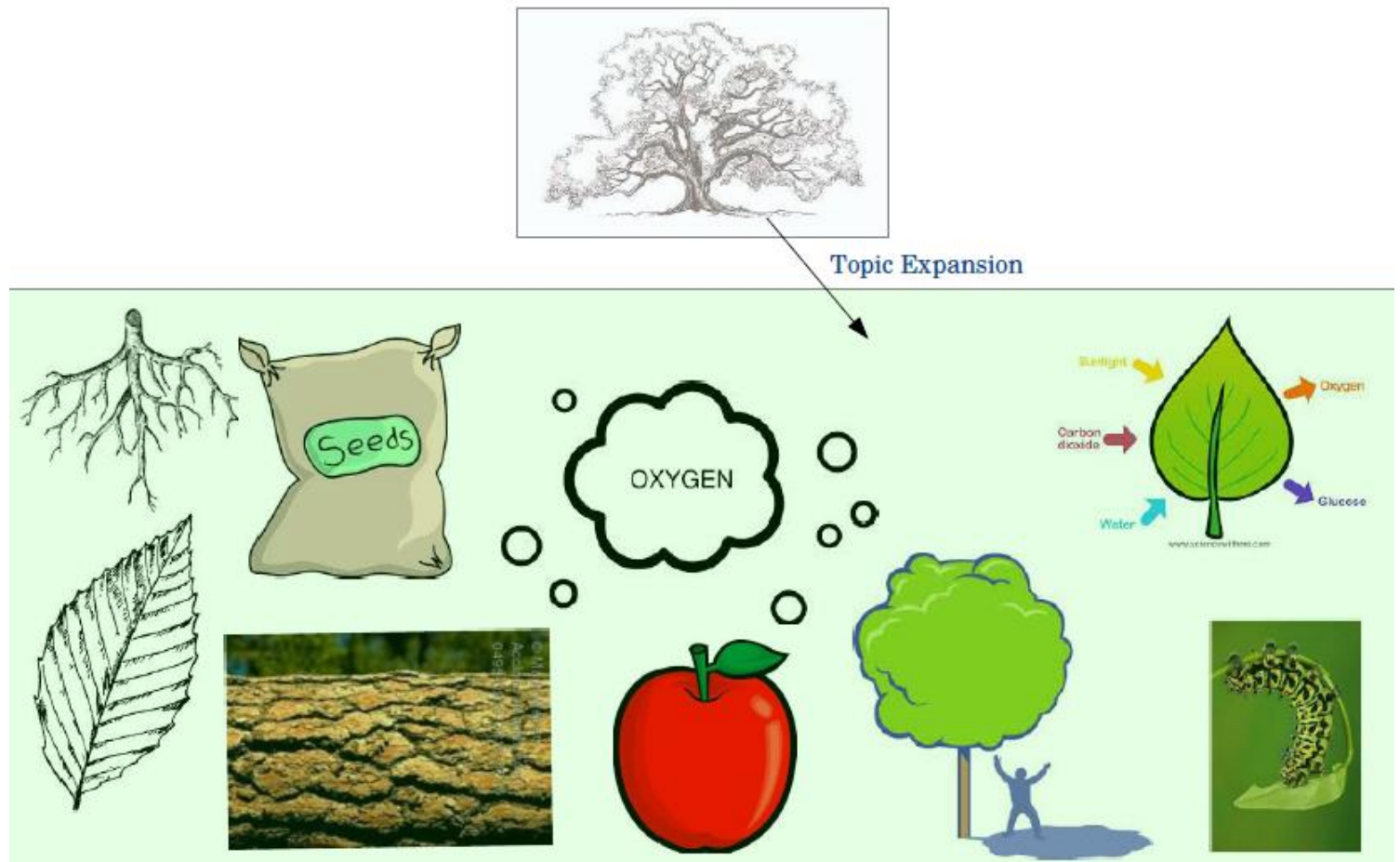


## Step-2 : Feature Extraction from our test documents.

- For each document whose similarity is to be measured, we find the keywords by removing the stop-words and considering only the adjective, nouns, and verbs
- Part-Of-Speech tagger, a Java library by Stanford NLP has been used to identify which word is an adjective, noun or verb.
- $$\text{Importance Score} = \frac{\text{Frequency of the keyword in document}}{\text{Frequency of all keyword in the document}}$$
- Each keyword of that document is stored in an HashMap (node value , importance score)

## Step-3 : Expansion of the keywords

- The human beings do not calculate the similarity between two events just based on the locally mentioned concepts. They bring in more number of related concepts from their general knowledge into their working memory and then compare them.

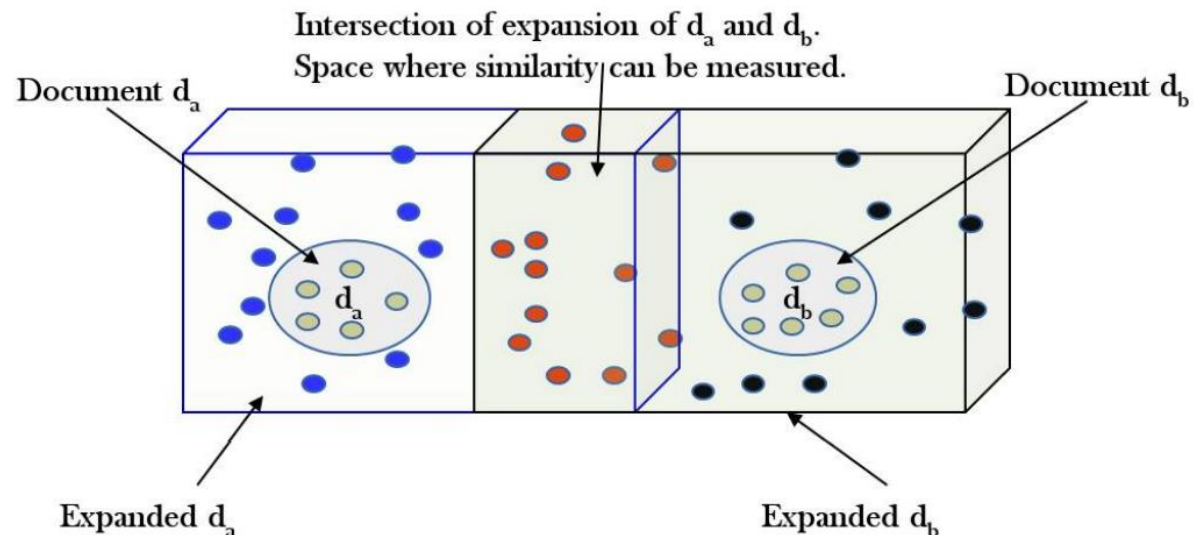


## Step-3 : Expansion of the keywords

➤ Similarly, we expand the locally identified important terms on the obtained knowledge base to get a larger global context. The global context contains the other terms which are related to the locally identified terms.

### PROCEDURE >>

- For each keyword, its edges in the knowledge base is referred and Top-K nodes are retrieved, where  $K = (\text{importance score} * \text{Total Edges})$
- These nodes can also be repeating as different keywords can have the same keyword in their Top-k
- We store the node-count pair in a HashMap showing which node appeared how many times in Top-K of all keywords



## Step-4 : Similarity Calculation of documents

We use 3 types of similarity measures

**1. Node Jaccard** = 
$$\frac{\text{Size (Intersection of Top-K keywords of doc1 n doc2)}}{\text{Size (Union of Top-K keywords of doc1 and doc2)}}$$

### 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;

## Step-4 : Similarity Calculation of documents

### 3. Cosine Similarity :

Term Frequency (TF) =  $\frac{\text{Frequency of a word in the document}}{\text{Total frequency of all the words in the document}}$

Inverse Document Frequency (IDF) =  $\frac{1 + \log(\text{Total no. of document})}{\text{No. of document that contains that word}}$

TFIDF = TF \* IDF

For all words in doc 1 and 2, we find TFIDF and obtain  $n \times 1$  vectors  $V_1$  and  $V_2$

Cosine Similarity :  $\frac{V_1 \cdot V_2}{|V_1| \cdot |V_2|}$

# BASELINE : Latent Semantic Analysis (L.S.A)

- LSA is based on one assumption: “the more two words occur in same documents, the more similar they are”.
- Constructs a co-occurrence matrix, where column names represent documents, row names - words and each cells[i][j] represents frequency of word[i] in document[j] using TF-IDF
- Singular Value Decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. It keeps only the “most important” vectors.
- Words are then compared by taking the cosine of the angle between 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.

## Problems with L.S.A.

- 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.

# Experimental Results

- We used a set of 30 documents for similarity, which has 10 types of categories and each category has 3 documents.
- First document from every category is compared with the rest 29 documents and top-5 documents are retrieved
- Each of these first document's top-5 results is then presented to user without letting him know which one is what method.
- This ground truth data will help us know that semantic matching actually gave us better results or not.
- We can see that for both L.S.A and our semantics based matching , top-5 documents retrieved are actually semantically similar to our query document

## ❖ SETUP

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



# L.S.A. Results

Crime_News_1	Movie_BodyGaurd_1	Story_Thief_1	Movie_Wanted_1	History_Missile_1
Crime_News_3 (0.70)	Movie_BodyGaurd_2 (0.86)	Story_Thief_2 (0.85)	Story_Thief_1 (0.72)	History_Missile_3 (0.59)
Crime_News_2 (0.52)	Story_Thief_2 (0.74)	Movie_Wanted_1 (0.72)	Movie_BodyGaurd_2 (0.70)	History_Missile_2 (0.56)
Movie_BodyGaurd_1 (0.52)	Story_Thief_1 (0.71)	Movie_BodyGaurd_1 (0.72)	Movie_BodyGaurd_1 (0.68)	News_Attack_2 (0.38)
Movie_Wanted_2 (0.52)	Movie_Wanted_1 (0.67)	Movie_BodyGaurd_2 (0.71)	Story_Thief_2 (0.67)	News_Attack_1 (0.31)
Story_Thief_1 (0.51)	Movie_Wanted_2 (0.65)	Movie_Wanted_2 (0.71)	Movie_Wanted_2 (0.64)	Mobile_Reviews_3 (0.26)
News_Attack_1	Mobile_Reviews_1	Scam_Info_1	Anti_Terrorist_1	News_BollyWood_1
News_Attack_2 (0.51)	Mobile_Reviews_2 (0.72)	Scam_Info_3 (0.68)	Anti_Terrorist_3 (0.67)	News_BollyWood_2 (0.66)
News_Attack_3 (0.47)	Mobile_Reviews_3 (0.69)	Scam_Info_2 (0.57)	Anti_Terrorist_2 (0.66)	News_BollyWood_3 (0.63)
Crime_News_3 (0.46)	Scam_Info_1 (0.54)	Anti_Terrorist_3 (0.51)	News_Attack_3 (0.46)	Crime_News_2 (0.58)
Story_Thief_1 (0.44)	Movie_Wanted_3 (0.46)	Anti_Terrorist_2 (0.50)	News_Attack_2 (0.43)	Movie_BodyGaurd_1 (0.53)
News_BollyWood_1 (0.43)	Scam_Info_2 (0.41)	Mobile_Reviews_3 (0.36)	Scam_Info_2 (0.41)	Crime_News_3 (0.52)

- We used an online application of LSA in comparing our documents
- We can see that most of the top-2 documents in every column belongs to the type of our query document only i.e most similar
- History\_Missile and Mobile\_Reviews are not at all similar (0.26)

# Node Jaccard Results

Crime_News_1	Movie_BodyGaurd_1	Story_Thief_1	Movie_Wanted_1	History_Missile_1
Crime_News_3 (0.211)	Movie_BodyGaurd_2 (0.320)	Story_Thief_2 (0.456)	Movie_Wanted_2 (0.286)	History_Missile_2 (0.1722)
News_BollyWood_1 (0.15)	Story_Thief_1 (0.271)	Movie_BodyGaud_1 (0.271)	New_Bollywood_2 (0.226)	History_Missile_3 (0.141)
Crime_News_2 (0.149)	Story_Thief_2 (0.230)	Movie_BodyGaud_2 (0.230)	Movie_BodyGaurd_2 (0.225)	News_Attack_2 (0.139)
Movie_Wanted_1 (0.147)	Movie_Wanted_2 (0.225)	Movie_Wanted_2 (0.225)	Movie_BodyGaurd_1 (0.225)	News_Attack_1 (0.135)
News_Bollywood_2 (0.143)	Movie_Wanted_1 (0.219)	Movie_wanted_1 (0.219)	Story_Thief_1 (0.213)	Antiterrorist_2 (0.132)

News_Attack_1	Mobile_Reviews_1	Scam_Info_1	Anti_Terrorist_1	News_BollyWood_1
News_Attack_2 (0.213)	Mobile_Reviews_2 (0.269)	Scam_Info_2 (0.208)	Anti_Terrorist_2 (0.236)	News_BollyWood_2 (0.207)
News_Attack_3 (0.196)	Story_Thief_3 (0.196)	Scam_Info_3 (0.171)	Anti_Terrorist_3 (0.190)	Movie_BodyGaurd_1 (0.190)
News_Bollywood_1 (0.151)	Mobile_Reviews_3 (0.195)	Anti_Terrorist_3 (0.159)	Scam_Info_1 (0.151)	Crime_News_2 (0.174)
News_Bollywood_2 (0.149)	Movie_BodyGaurd_3 (0.184)	Anti_Terrorist_1 (0.157)	History_Missile_2 (0.151)	Story_Thief_1 (0.167)
Anti_Terrorist_1 (0.145)	News_Attack_3 (0.183)	Anti_Terrorist_2 (0.144)	News_Attack_2 (0.146)	Movie_BodyGaurd_2 (0.167)

# Edge Jaccard Results

Crime_News_1	Movie_BodyGaurd_1	Story_Thief_1	Movie_Wanted_1	History_Missile_1
Crime_News_3 (0.228)	Movie_BodyGaurd_2 (0.363)	Story_Thief_2 (0.472)	Movie_Wanted_2 (0.286)	History_Missile_2 (0.173)
Crime_News_2 (0.164)	Story_Thief_1 (0.317)	Movie_BodyGaurd_1 (0.323)	New_Bollywood_2 (0.226)	News_Attack_1 (0.161)
News_BollyWood_2 (0.163)	Movie_Wanted_2 (0.266)	Movie_BodyGaurd_2 (0.294)	Movie_BodyGaurd_2 (0.225)	News_Attack_2 (0.159)
News_Bollywood_1 (0.159)	Movie_Wanted_1 (0.263)	Movie_Wanted_2 (0.259)	Story_Thief_1 (0.213)	Antiterrorist_2 (0.158)
Movie_Wanted_1 (0.157)	Story_Thief_2 (0.262)	News_Bollywood_2 (0.257)	Movie_BodyGaurd_1 (0.225)	Scam_Info_3 (0.147)

News_Attack_1	Mobile_Reviews_1	Scam_Info_1	Anti_Terrorist_1	News_BollyWood_1
News_Attack_2 (0.264)	Mobile_Reviews_2 (0.285)	Scam_Info_2 (0.238)	Anti_Terrorist_2 (0.276)	News_BollyWood_2 (0.287)
News_Bollywood_1 (0.193)	Mobile_Reviews_3 (0.265)	Scam_Info_3 (0.201)	Anti_Terrorist_3 (0.230)	Movie_BodyGaurd_3 (0.198)
Crime_News_3 (0.185)	Story_Thief_3 (0.236)	Anti_Terrorist_3 (0.179)	History_Missile_2 (0.198)	Scam_Info_1 (0.171)
News_Bollywood_2 (0.184)	Movie_BodyGaurd_3 (0.194)	Anti_Terrorist_2 (0.134)	Scam_Info_1 (0.181)	Story_Thief_1 (0.167)
History_missile_2 (0.179)	News_Attack_3 (0.183)	Anti_Terrorist_1 (0.176)	Movie_BodyGaurd_2 (0.167)	Crime_News_2 (0.174)

# Cosine Similarity Results

Crime_News_1	Movie_BodyGaurd_1	Story_Thief_1	Movie_Wanted_1	History_Missile_1
Crime_News_3 (0.268)	Movie_BodyGaurd_2 (0.363)	Story_Thief_2 (0.472)	Movie_Wanted_2 (0.286)	History_Missile_2 (0.173)
Crime_News_2 (0.214)	Movie_Wanted_2 (0.287)	Movie_BodyGaurd_1 (0.323)	New_Bollywood_2 (0.226)	News_Attack_2 (0.161)
News_BollyWood_1 (0.183)	Story_Thief_1 (0.265)	Movie_BodyGaurd_2 (0.294)	Movie_BodyGaurd_2 (0.225)	News_Attack_1 (0.159)
News_Bollywood_2 (0.149)	Movie_Wanted_1 (0.243)	News_Bollywood_2 (0.257)	Story_Thief_1 (0.213)	Antiterrorist_2 (0.148)
Movie_Wanted_1 (0.157)	Story_Thief_2 (0.192)	Movie_Wanted_2 (0.199)	Scam_Info_3 (0.147)	Movie_BodyGaurd_1 (0.225)
News_Attack_1	Mobile_Reviews_1	Scam_Info_1	Anti_Terrorist_1	News_BollyWood_1
News_Attack_2 (0.264)	Mobile_Reviews_3 (0.245)	Scam_Info_2 (0.238)	Anti_Terrorist_3 (0.276)	News_BollyWood_2 (0.287)
Crime_News_3 (0.185)	Mobile_Reviews_2 (0.217)	Scam_Info_3 (0.201)	Anti_Terrorist_2 (0.230)	Movie_BodyGaurd_3 (0.198)
News_Bollywood_1 (0.173)	Story_Thief_3 (0.198)	Anti_Terrorist_3 (0.179)	History_Missile_2 (0.198)	Scam_Info_1 (0.171)
News_Bollywood_2 (0.164)	Movie_BodyGaurd_3 (0.194)	Mobile_Reviews_2 (0.157)	Scam_Info_1 (0.181)	Story_Thief_1 (0.167)
History_missile_2 (0.159)	Scam_Info_1 (0.171)	Anti_Terrorist_1 (0.136)	Scam_Info_3 (0.161)	Crime_News_2 (0.174)

# Ground Truth Data

We gave these top-5 lists of each document for each of the LSA, NJ, EJ and CS measures and the following results were obtained

	LSA	<u>Node Jaccard</u>	<u>Edge Jaccard</u>	Cosine Similarity
Crime_News_1	6	3	5	6
Movie_BodyGaurd_1	5	2	7	6
Story_Thief_1	7	5	4	4
Movie_Wanted_1	4	5	4	7
History_Missile_1	4	7	4	5
News_Attack_1	5	4	4	7
Mobile_Reviews_1	6	3	2	8
Scam_Info_1	4	5	7	4
Anti-Terrorists_1	4	7	8	1
News_bollywood_1	6	5	4	5
	51	45	49	53

# Another Example :

1. “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, off Bankapura Cross in Haveri district of central Karnataka. The bus, operated by Bangalore-based National Travels, was on its way from Bangalore to Mumbai.”
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. “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 11 pm on Tuesday from Kalasipalyam area in Bengaluru to Hyderabad and picked up 50 passengers along the way.”

Which two are similar?

<b>Node <u>Jaccard</u> Similarity</b>		
<b>Snippet 1</b>	<b>Snippet 2</b>	<b>Snippet 3</b>
Snippet 3 (0.2676)	Snippet 1 (0.0869)	Snippet 1 (0.2676)
Snippet 2 (0.08695)	Snippet 3 (0.0833)	Snippet 2 (0.0833)

Snippet  
1<sup>st</sup> and 3<sup>rd</sup>  
are similar

<b>Edge <u>Jaccard</u> Similarity</b>		
<b>Snippet 1</b>	<b>Snippet 2</b>	<b>Snippet 3</b>
Snippet 3 (0.1763)	Snippet 1 (0.0676)	Snippet 1 (0.1786)
Snippet 2 (0.0356)	Snippet 3 (0.0481)	Snippet 2 (0.0561)

<b>Cosine Similarity</b>		
<b>Snippet 1</b>	<b>Snippet 2</b>	<b>Snippet 3</b>
Snippet 3 (0.1702)	Snippet 1 (0.0496)	Snippet 1 (0.1702)
Snippet 2 (0.0496)	Snippet 3 (0.0461)	Snippet 2 (0.0461)

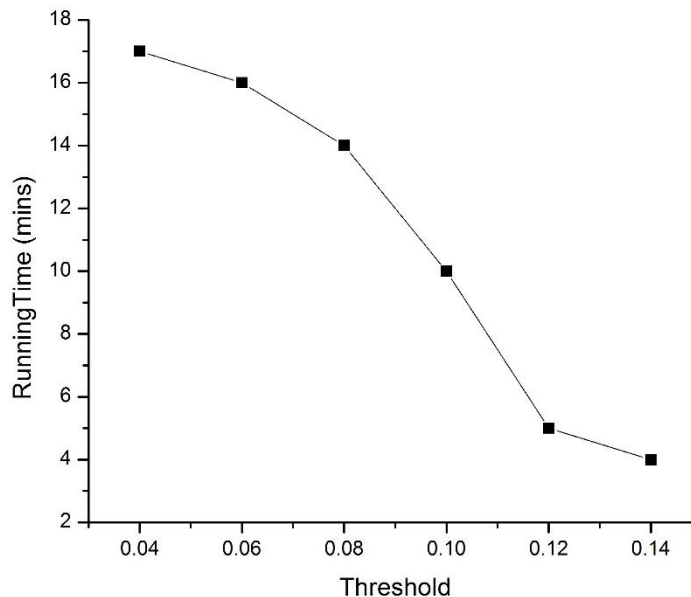
<b>LSA</b>		
<b>Snippet 1</b>	<b>Snippet 2</b>	<b>Snippet 3</b>
Snippet 3 (0.57)	Snippet 3 (0.35)	Snippet 1 (0.57)
Snippet 2 (0.16)	Snippet 1 (0.16)	Snippet 2 (0.35)

<b><u>Cosine Similarity without graph based approach</u></b>		
<b>Snippet 1</b>	<b>Snippet 2</b>	<b>Snippet 3</b>
Snippet 2 (0.23)	Snippet 1 (0.23)	Snippet 2 (0.007)
Snippet 3 (0.036)	Snippet 3 (0.007)	Snippet 1 (0.036)

Our  
Solution  
Agrees!!

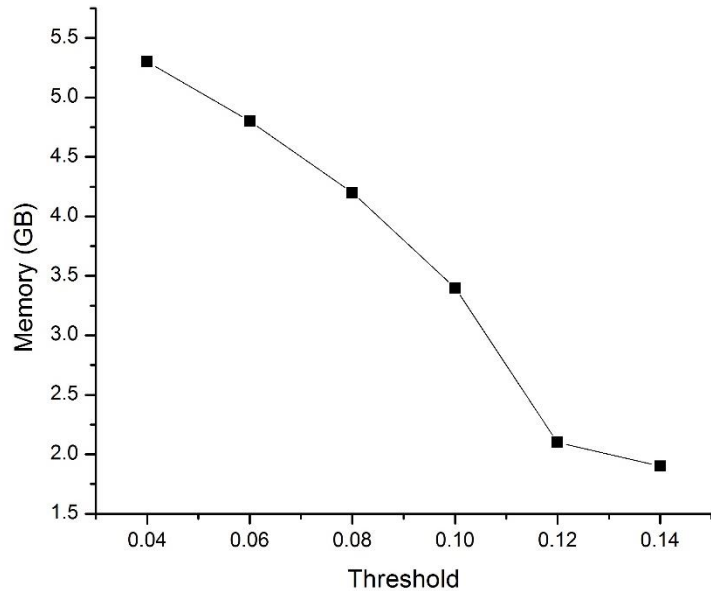
# Analysis

- Since many people voted for Cosine and LSA answer's to be correct, we can say that (not surely) they are the best technique.
- 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



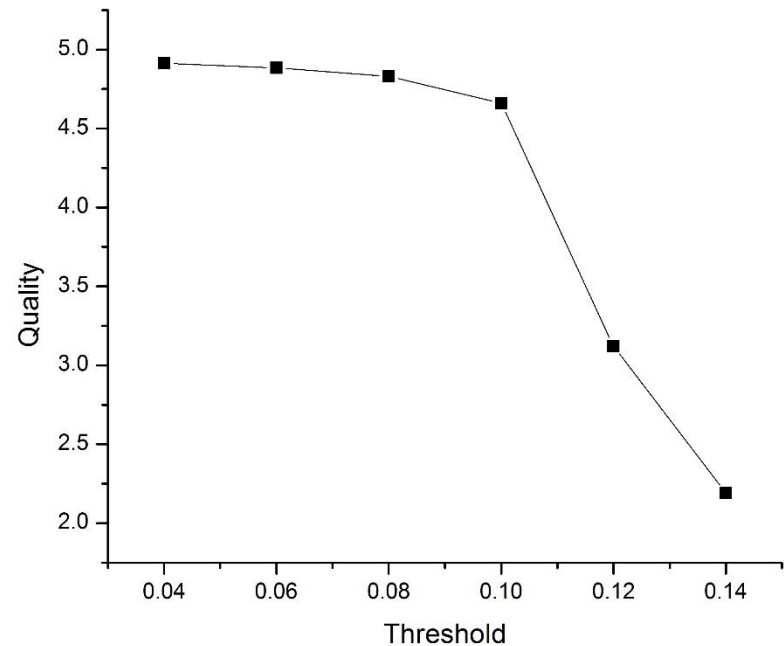
Its obvious that on not taking all the edges, it will affect the Time Complexity. Sorting takes  $n \log n$ . As 'n' (number of edges) decrease, our method wil take lesser time.





Same reason!

As the number of edges decrease (by increasing threshold), memory will be freed and hence less memory is used at big thresholds



If the number of edges are being decreased, the quality of our project decreases since the expansion in our knowledge base will not be accurate.

We also tried running cricket focussed documents over a Biological Knowledge base and it gave completely irrelevant results.

- ✓ Always ensure that the knowledge base is relevant to the documents being matched.
- ✓ Try using a general document corpus for creating a sufficient knowledge base.

# Conclusion

- ❖ In this project, we attempt to solve two kinds of semantic similarity problems namely
  - Identification of semantic similarity (identifying the similarity score between the given pairs of documents)
  - Semantically similar information retrieval (retrieving similar documents for a given document from a corpus).
- ❖ Also, our experiment results shows that Cosine and LSA are the best technique comparing it with the ground truth. But the answer set of each technique was more or less similar, so we cannot be sure.
- ❖ We tried running it with documents that are totally irrelevant to the knowledge base and as expected, we got rubbish results.
- ❖ So, always try to use a general-most possible knowledge base to teach the computer.

# Some general questions...

## 1. Any previous work?

Yes, Extracting Semantics from Co-occurrences, DKE, July 2014

## 2. Why can't existing things be not used over this?

We already told about the drawbacks of LSA. It does not expand it's knowledge base

## 3. When does the proposed technique fails?

When documents to be matched are completely irrelevant to the knowledge base, hence there will be no useful expansion.

## 4. How to make it better?

By using some better similarity calculation functions, if available

By ensuring that the knowledge base is always sufficient for the documents to be matched.

# THANK YOU

