

# Targeted Factor Analysis over a period of time

Raghavender Sahdev and Forouq Khonsari

Department of Computer Science and Engineering

York University

Toronto, ON, Canada - M3J1L3

e-mail: {sahdev, khonsari}@cse.yorku.ca

**Abstract:** *In this project we aim to predict the trend of a particular factor. Here we focus on predicting the trend of violence. We propose 2 techniques to solve this - Word Embedding and Point wise Mutual Information (PMI). A violence score is assigned for each day using each of these approaches and a graph is plotted against days for a specific month. We use data from different sources from different parts of the world namely - news articles from various news channels, magazine articles and Television news. These news sources are taken from different parts of the world. We build/train the models from the xml files collected from these channels. We then validate our results from the ground truth provided. Finally we detect the violent peaks to find out which method is gives better results - PMI or word embedding.*

## 1. INTRODUCTION

### Objective of the project

In this project we aim to analyse a targeted factor from a given corpus. We want to analyse if the given factor increases, decreases or remains the same over a period of time. The factor can be anything like spread of infectious diseases, migration, violence, environmental threat or something else. For the simplicity of this project we focus on one of the factors – violence. After careful analysis of the data and applying appropriate data pre-processing techniques, we say that violence increases from day 1 to day 5 then drops and again rises towards the end of the month. We show this relationship with a plot of a violence score against the number of days.

Specifically we focus on extracting the violent information (violent content/events) from news articles and notice the changes of violence over a period of time. Thus, the main task of our project could be defined as extracting the articles related to violence and providing a measure to see how relevant each article is to violence. By having this task done, the only job remaining is combining this information with an element of time. To sum up, in the subsequent sections we focus on methods introduced to extract articles related to a pre-specified factor (violence here), and then we describe our approach used.

### Organization of the report

The initial part of the report gives brief introduction in section 1. We then present the relevant work done in this field in section 2. In section 3 we describe the approach we have used to solve our proposed problem. In this section we give a detailed analysis of pre-processing done with the data and the approaches used to come up with the violence scores. In section 4 we provide with empirical

results and visualize the results. In the section 5 we conclude the project and provide future research potential of the project. Finally we provide an appendix section which has the link to our code and provides a user's manual explaining what each code does. It also contains a sample input and output for the program. We also provide the limitations of the provided code and some known prerequisites for the input corpus.

## 2. RELEVANT WORK

Information extraction from text is a wide domain in data mining, which includes various tasks, like emotion detection [5], events extraction [7] or topic detection and tracking [Iraq paper], topic modelling and sentiment analysis. The Latent Dirichlet Allocation (LDA) proposed by Blei, D.M. et al [3] and Probabilistic Latent Semantic Analysis have dominated the field of Topic modelling for a long time. They can also be used for estimating continuous representations of words. However both of them suffer from a common problem inability to handle larger datasets due to increased complexity.

Recently Mikolov, T. et al [1] built a model to compute the word vector representations of words in large datasets. Their proposed approach can learn high quality word vector representations from a 1.6 billion words dataset in less than a day to generate vectors for a vocabulary of millions of words. Their model is inspired from Feed forward and recurrent neural networks. They propose two models in their paper – the Continuous Bag of Words Model and the Skip Gram model to compute word vectors. Once the word is represented in the form of vectors one can make interesting interpretations/relationships to find vectors similar to each other. Example we want to find a word similar to small in the same sense as big is similar to biggest, so we can simply compute the vector –  $X = \text{vector}(\text{"biggest"}) - \text{vector}(\text{"big"}) + \text{vector}(\text{"small"})$ . Then they find from the vocabulary the word closest to the vector X which gives smallest. Not only simple relations like this one they also find relations pertaining to countries and their cities/capitals –  $X = \text{vector}(\text{"France"}) - \text{vector}(\text{"Paris"}) + \text{vector}(\text{"Berlin"})$  gives them the vector corresponding to Germany. We use this approach to compute the word vector representations from our corpus.

Another renowned work is that of Church and Hanks [4] wherein they propose an objective measure based on the information theoretic notion of mutual information, for estimating word association norms from computer readable corpora. They compute the mutual information between 2 words by considering a window of a specific size. Smaller window sizes established fixed expressions (like idioms – bread and butter) and larger window sizes provided them with semantic concepts and other relationship that holds over larger scales.

One novel and unique method for event extraction is to use Cascaded Extraction Grammars which is used in [7]. In this method, a highly efficient extraction pattern engine is used, which is capable of matching thousands of patterns against MB-sized texts within seconds. This extraction pattern engine, uses predefined patterns and tries to find matches between the text and these patterns. In order to define these patterns, a grammar is needed. For creating grammar patterns a weakly supervised machine learning (ML) algorithm has been deployed. In particular, the pattern acquisition process involves multiple consecutive iterations of ML followed by manual validation. The method uses clusters of news articles. Each cluster includes articles from different sources about the same news story. An automatic procedure for syntactic expansion complements the learning, i.e. based on a manually provided list of words which have identical (or nearly identical) syntactic model of use (e.g. killed, assassinated, murdered, etc.) new patterns are generated from the old ones by substituting for each other the words in the list. Next step is the population of the semantic classes referenced by grammar rules. The task is also partially automated through application of machine learning techniques. A semi-supervised system is deployed which takes as input a small set of sample phrases belonging to a target semantic class (seed terms) and a unannotated corpus of news article, and then further populates the semantic class with new instances (terms). The learning

process consists of two steps: a 'Feature extraction and weighting' stage, where context features for a category C are selected among the terms co-occurring with any of the seed terms of C, and then weighted and ordered based on a Point-wise Mutual Information measure. The whole learning process is semi-supervised, as a minimal human intervention is required in the form of manual selection of context feature output between the two stages.

Another approach in information extraction has been proposed by Agrawal and An [5] which detects emotion from text. Here the emotion detection task is modelled as a classification problem where one or more nominal labels are assigned to a sentence from a pool of target emotion labels. In this paper a word-level analysis is performed by computing an emotion vector for the words by calculating their semantic relatedness to emotion concepts. The emotion of a word can be defined as a vector whose elements each represent the strength of the affinity of the word for an emotion category (happiness, sadness, anger, fear, surprise and disgust). Traditionally; the emotion vector is calculated by directly matching it against an affect dictionary. One of the shortcomings of this approach is that it cannot detect emotions if the sentence does not contain any obvious emotional keywords.

To calculate the semantics relatedness between two words, one assumption is that the affect words (adjectives, nouns, verbs and adverbs) that frequently co-occur together have the same emotional tendency. If two words co-occur more frequently, they tend to be semantically related. The Point wise Mutual Information (PMI) metric is used to measure the similarity between two terms by using the probability of co-occurrence.

A simple solution to derive the emotion vector from a word is to use the PMI score between the word and the word representing an emotion concept. However, since an emotion concept can often be expressed through various words (e.g. 'glad' or 'joy' for 'happiness'), one way is to use a few words rather than just one generic word representing the entire emotion category. The representative words in are the most commonly used synonyms taken from a generic thesaurus. A surprising result of this approach is that different common synonyms produce similar results.

The PMI scores between each document and each representative word of an emotion category are then used to compute the PMI score of each document word and the category. Then the category with the most score is picked as the category of the document.

In our report we use a similar idea as used above: if a document has lot of key words related to violence; it means it gets a higher violence score. Similarly if not many violent words appear in the document we say it to be unrelated to violence and it gets a low violence score. The next section explains this in detail.

### **3. DATA PRE-PROCESSING**

#### The Dataset and pre-processing

In this project we consider the Georgetown Data set which consists of news articles majorly from Mainstream Media channels and some are from Government websites. The dataset is very diverse in the sense that it has news articles from most countries in the world like Guyana, India, Iraq, Russia, UAE, Malaysia, Canada, Sierra Leone, Indonesia, Zimbabwe, Philippines, United States, United Kingdom, Israel and many more countries. Various news channels and many government sites were used to obtain this data. The data used here is in xml format which contains tags like Language, text, publication date, capture date, etc. We obtain the relevant tags like the text which contains the content of the articles and the publication date which tells when the article was published in the media. The dataset has articles which are for the most part in English but it also contains articles in Arabic, Persian, Chinese and 2-3 more languages. These articles were initially removed based on the ASCII values, however it was later decided not to remove these articles as some of the words

occurring in these articles could also be related to violence. So we do have some words from foreign languages in our vocabulary.

For pre-processing the data, we follow the following steps:

- Extract from the XML file the content of the tags Publication Date and text. Publication date tells us the date the document was published and text contains the content of the article.
- Convert all alphabets to lower-case so that while building the models we do not treat words like assault, ASSAULT and Assault differently.
- We then remove the punctuations from the documents. We consider the following punctuations - , . \_ - + = ~ ! @ # \$ % ^ & \* ( ) { } [ ] ; : ' " ? / and remove them.
- We do not remove the stop words as a part of pre-processing as the approach of word2vec which we describe next takes care of it.
- After doing the above steps we have all the files from a specified folder pre-processed, we save the new files in a separate folder.
- Finally we concatenate all the files content into a single text file and call it the pre-processed document.

### Seed Words

We generate a list of seed words related to violence. This seed is manually generated which considers the most common words which are related to violence. In the seed we include the following 75 words:

Civil, war, violence, violent, conflict, fight, killing, kill, battle, massacre, injury, butcher, explosion, bombs, bomb, corpse, abduction, ambush, suicide, bomber, rape, persecution, assassination, terror, militia, military, attack, assault, gang, crime, theft, clash, mortar, rocket, siege, blockade, shelling, shells, force, gun, soldiers, rebel, unrest, troubled, insurgent, revolutionary, dead, killed, crisis, victim, detention, arrest, detainee, prisoner, prison, indiscriminate, checkpoint, operation, torture, elements, execute, execution, withdrawal, feud, weapons, kidnapping, body, ransom, ammunition, martyr, injured, rounds, patrol, target, fighter

Later we use this list of seed words to compute the violence score. We compute the similarity between each of these words and the document words to come up with a violence score. We explain this later in the report.

## **4. DESCRIPTION OF THE ALGORITHMS**

After pre-processing the document and generating seed words, we have a single file from which we need to learn the model. This file contains all articles with their dates and content in each line. In this project we propose 2 techniques to compute the trend of violence:

1. Word Embedding
2. Point wise Mutual Information (PMI)

### Word Embedding

Word Embedding was proposed by Mikolov T. et al in [1]. Word Embedding have been successfully used to compute vector representations of words from a huge corpus. In this report we use the word2vec method using the deeplearning for java library as in [10]. Word2Vec is a two layer neural network that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. So we input our text file to the corpus and we get as out the vocabulary and each word in this vocabulary is represented by a 100 dimensional feature vector. We

train this model with a window size of 5 words and minimum frequency set as 10 words, and set the epoch = 1.

Word2Vec is very useful in the sense that it groups the vectors of similar words together in vector space. It detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as context of individual words. Word2vec can make highly accurate guesses about a word's meaning based on past appearances. These guesses can be used to establish relatedness of various words with a specific word. Word2Vec can be used in sentiment analysis.

The output of word2vec is a vocabulary in which each item has a vector attached to it, which can be used to further compute similarities between words using simple similarity measures like the cosine similarity. We compute the nearest words to violence and get very meaningful results which we show in the next section.

In the paper by Mikolov T. et al they call the word vectors as neural word embedding. Once each word is converted to a numerical vector one can perform powerful mathematical operations on it to detect semantic similarities. Neural word embedding can be simply thought of as translation of each vector to a new feature space. They can be thought of as an automatic encoder which encodes each word in the vocabulary based on its neighbouring words in the corpus. So closer words (like kill and assault) will have similar embedding whereas distant unrelated words (like happy and fighter) would have different embedding/representations. Two approaches have been proposed in the paper to compute these vector representations:

- Continuous Bag of Words model - This uses context to predict a target word
- Skip-Gram model – this uses a word to predict a target context.

In this project we use the skip gram model as it works better with large datasets.

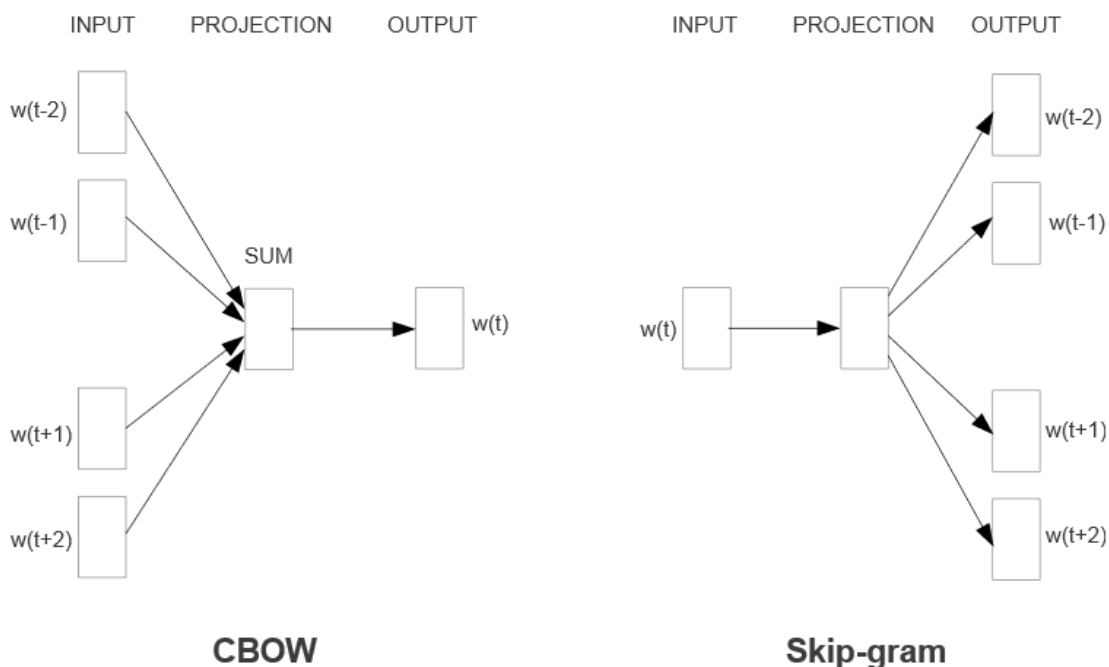


Figure 1: (image and content source [1], [8]) The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

When the feature vector assigned to a word cannot be used to accurately predict that word's context, the components of the vector are adjusted. Each word's context in the corpus is the teacher sending error signals back to adjust the feature vector. The vectors of words judged similar by their context are nudged closer together by adjusting their numbers in the vector.

We in this project encode each word as a 100 dimensional vector, which means we use 100 numbers to describe a word. Following shows a vector representation of a sample word trained from our corpus –

```
polluted      0.042375434190034866      0.01849209889769554      -0.16593559086322784
0.08191248029470444 -0.17923250794410706 -0.06226591765880585 -0.14931389689445496 -
0.10120388120412827 0.014830808155238628 -0.16163018345832825 0.1762140989303589 -
0.12769509851932526 -0.21592633426189423 -0.35278230905532837 -0.09322133660316467 -
0.08940312266349792 0.17831642925739288 0.2716498374938965 -0.01310174074023962
0.03199233487248421 0.219010591506958 -0.06445935368537903 -0.1961628496646881
0.07690036296844482 -0.031715407967567444 0.07739852368831635 0.1944802850484848
0.09778708964586258 0.022324467077851295 -0.11502872407436371 -0.060946378856897354 -
0.24064932763576508 0.03054555132985115 -0.10066346079111099 -0.012824032455682755 -
0.15709926187992096 0.08259209990501404 0.013623847626149654 -0.2396385222673416 -
0.24268339574337006 -0.1652454286813736 0.06022050231695175 -0.10756220668554306
0.19011732935905457 -0.13073405623435974 -0.11918039619922638 0.11501111835241318
0.06698443740606308 -0.19133920967578888 0.18716759979724884 0.17790670692920685 -
0.012576088309288025 0.11012710630893707 0.14500457048416138 -0.12210998684167862
0.04329126328229904 -0.05912068113684654 0.07546325027942657 -0.06311380863189697 -
0.21200692653656006 0.018347984179854393 -0.02472584694623947 -0.1025528684258461 -
0.5136263370513916 -0.07992315292358398 -0.15260663628578186 -0.054483864456415176
0.12295696884393692 0.14304566383361816 -0.3889305889606476 -0.07788556069135666
0.1511596143245697 0.12361390143632889 -0.15385989844799042 -0.037169139832258224 -
0.05754794552922249 0.2717892527580261 0.0686732828617096 0.020741520449519157
0.10472932457923889 -0.12593024969100952 -0.07872222363948822 0.04442039132118225 -
0.11891709268093109 -0.3044775128364563 0.13702474534511566 -0.029430247843265533 -
0.04287999868392944 0.03503192216157913 -0.05820851773023605 0.0824844017624855
0.14923492074012756 0.12303516268730164 0.015439540147781372 0.02271568588912487
0.07457729429006577 -0.10642362385988235 -0.03187739849090576 -0.12977780401706696 -
0.12253804504871368
```

We also have vector representation of Arabic words, Hindi words, Chinese words and some other languages words. Though the number of words from foreign languages is very less as compared to English but our model does have vector representation of frequent words from different languages. Some of the foreign language words representations:

```
अध्यक्षतामा -0.11560482531785965 0.06432882696390152 .....
снижает 0.09939511120319366 0.0114082181826 .....
```

A well trained set of word vector places similar vectors close to each other in the 100 dimensional space. To visualize it in 2 D this means having words like kill, alleged, accuse, rape, assault are together and words like diseases, food, infection, diarrhoea occur together. This would mean something as shown below. Here we show this in 2 dimensions since it is easy to visualize. Imagine the same in 100 dimension space. So similar vectors (words) in that space would be in closer vicinity of each other.

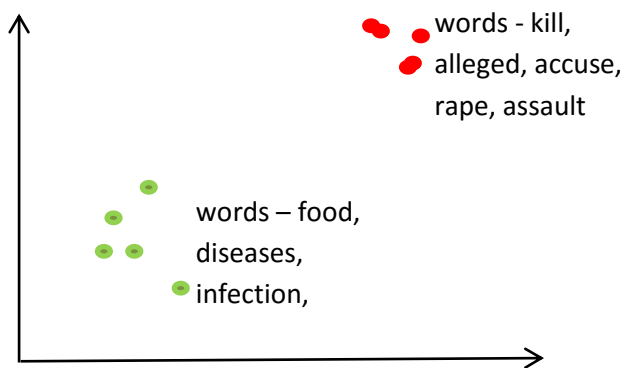


Figure 2: Closely separated words appear together in the vector space. (Here it is shown as 2 dimensions for the sake of simplicity)

After computing the vector representations from our corpus we find out the 100 nearest words based on their cosine Similarity – we get words like killings, anti-government, attacks, abuses, unrest, protest, uprising, kidnapping, extremists, genocidal, trafficking, aggression, violent, torture, crimes, sexual, rebellion, against, humanity, rape, repression, discrimination, etc. This makes a lot of sense as these are words which are actually associated with violence. One could also do a cluster analysis to find out similar words in the 100 dimensional feature space. Semantically similar words would appear in the same cluster.

We then use these word vectors to compute the violence score. We follow the following steps to compute the violence score

1. Pre-process the corpus to generate a text file containing all articles in separate lines.
2. Compute word embedding for each word using word2vec.
3. Compute the cosine similarity between each word in the vocabulary and each seed word to generate matrix of cosine similarities of size seed\_word\_size x vocabulary\_size.
4. For a particular document, iterate over all words to compute the cosine similarity of each word in the document with each of the seed words. This tells us how similar the words in the document are to violence. The seed contains all words related to violence, so if a document is talking a lot about violence, it would contain a lot of words related to violence. Example – consider the following 2 articles-

*Article1: killing rage voices grave compared killing rage written eamon collins former provisional ira intelligence officer brutally killed allegedly provisional ira giving evidence south armagh warlord tom slab murphy libel action voices grave partly based memoirs former provisional ira leader brendan hughes died harshly critical provisional ira published memoirs article compares contrasts books*

*and*

*Article2: according saudi british bank hsbc rise employment nonoil sector saudi economy weakening employment growth trend arab news*

In article 1 there are word like *killing, rage, brutally, killed, allegedly, died* which have a higher cosine similarity with the violence seed words whereas article2 contains words which have lower

cosine similarity with the violence seed words. So clearly as per our algorithm we assign article 2 a higher violence score and assign article2 a lower violence score.

5. After computing the cosine similarities of each word in the document with the seed, we assign each document a violence score based on the simple average. So the violence score of the  $k^{th}$  document  $d_k$  would be:

$$Violence\ Score(d_k) = \frac{1}{n} \sum_{i=all\ seed\ words} \frac{1}{d} \sum_{j=all\ document\ words} cosine\_similarity(s_i, w_j)$$

here  $s_i$  is the  $i^{th}$  seed word in the seed,  $w_j$  is the  $j^{th}$  document word,  $n$  is the number of seed words and  $d$  is the number of document words

6. At this stage each document is assigned a violence score. Now we compute the violence score for each day by averaging the violence scores for all documents belonging to a particular day.

$$Violence\ Score(day_d) = \frac{1}{D} \sum_{k=over\ all\ documents\ for\ day\ d} Violence\ Score(d_k)$$

where  $D$  is the number of documents on the  $d^{th}$  day

7. A day is said to be more violent if it has violent documents (documents having higher violence scores). A document is said to be violent if it has more violent words contained in it. A word is said to be more violent if it has a higher cosine similarity with the violent seed words. And the violent seed words themselves indicate violence as these words have been generated by human knowledge.
8. After computing the violence scores for each day we plot the trend of violence against the days and we then compare it with the ground truth. The plots are shown in the next section.

### Point wise Mutual Information (PMI)

Similar to the usage of word embedding in the previous section we use the Point wise Mutual Information as an alternative way to do what we did earlier – computing the violence score. PMI is also a measure to compute similarities between 2 words as is the cosine similarity for word embedding. We define the point wise mutual information between 2 words as follows:

$$PMI(x, y) = \frac{P(x, y)}{P(x) * P(y)}$$

where  $P(x, y)$  is the probability that the words  $x$  and  $y$  occur together,  $P(x)$  is the probability of occurrence of the word  $x$  in the corpus and  $P(y)$  is the probability of occurrence of the word  $y$  in the corpus

PMI has been extensively used by researchers in the field of information retrieval from corpuses as explained in section 2. We use PMI to compute similarity between seed words and document words and come up with a violence score. The PMI value between seed words and document words would be large for words that are related to violence as they occur more closely to each other in the documents. We follow very similar steps as for word embedding to compute the violence score:



1. Pre-process the corpus to generate a text file containing all articles in separate lines.
2. Compute the PMI value between each seed word and each vocabulary word to generate a matrix (of size seed\_word\_size x vocabulary\_size) containing the PMI values.
3. Compute the PMI value for each document by aggregating the PMI values for each document word with each of the seed words as was done for word embedding case.
4. After getting a PMI based violence value / score from step 3, we have a violence score for each document. We then compute the violence score for each day by aggregating the PMI scores over all documents for that specific day.
5. We then plot the violence scores against the days and compare it with the ground truth.

It should be noted the violence score computation between PMI and word embedding is the same. The only difference lies in the way in which the similarity matrix (of seed words and vocabulary) is generated. The over view of the method can be seen in figure 3 for a better understanding.

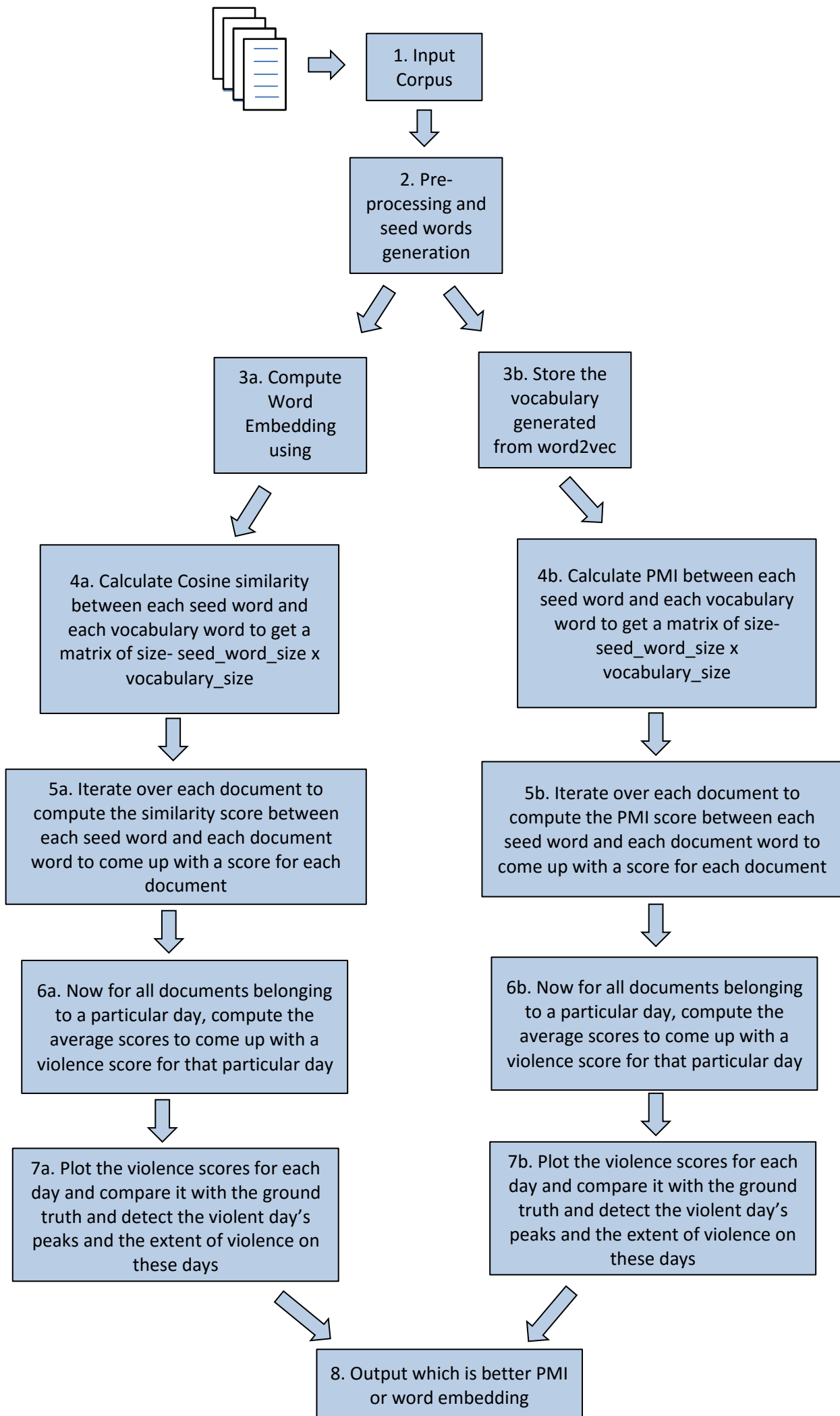


Figure 3: Overview of our proposed Alogrithm

## 5. EXPERIMENTAL RESULTS

In this project we considered a corpus containing 35,240 xml files. Each file contains a news article. We ran this corpus through our proposed methods in the previous section. After running word2vec on this corpus we obtained a vocabulary of size 50,678 distinct words. We truncate all words based on their minimum frequency of 10. Vocabulary size before truncations was 377,301 words. The total number of word occurrences in the corpus was 14,933,868. We set the number of epochs as 1 for the network model. These files covered articles over a period of one month (July 2013).

We got the following graph of the violence score vs. days for the case of word embedding and PMI as shown below:

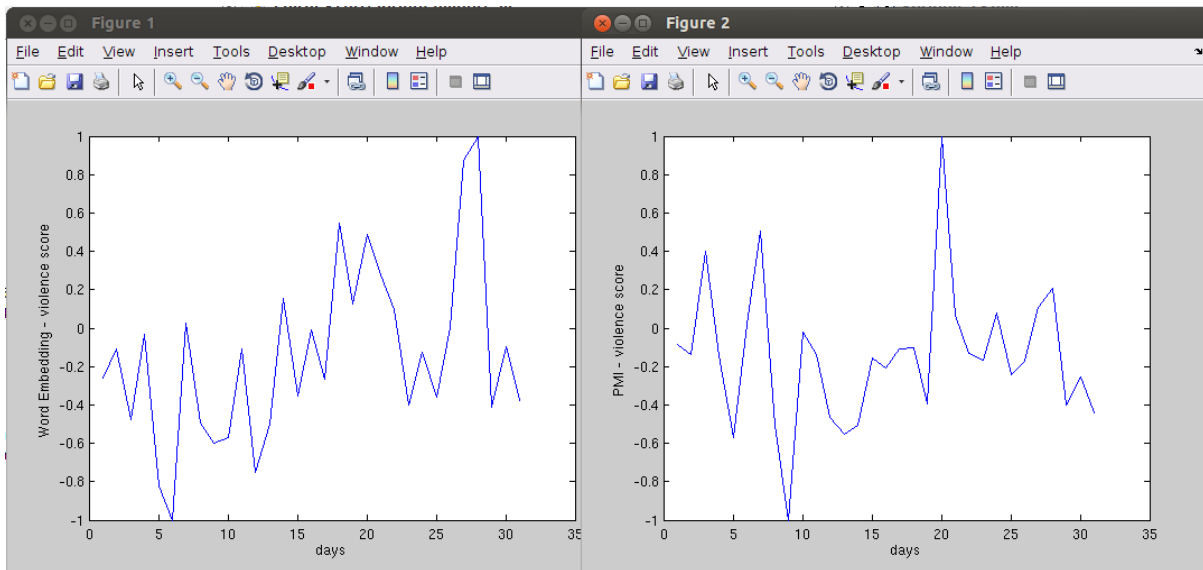


Figure 4: Plot of Violence Score against days for (Word Embedding - left) and (Point wise Mutual Information – right)

Now we compare our results with the ground truth. In these plots we can ignore the negative violence scores and set them to zero which we do in the evaluation part in the next section.

### Evaluation

In this section we present the generation of ground truth and the evaluation/validation method. We here scored every document manually using the same criteria. The score of a document is in the scale of 0 to 5 and it means how much each document is related to violence. The criteria are as followings:

1. If the document just included wounded/injured it gets a lower score than when it includes killed/dead. If it includes both will get the same score as killed/dead. Example: *Three were killed and at least 13 wounded by a bombing in Bayaa, Baghdad.*
2. Kidnapped has a lower score than wounded.
3. If the news is just about the refugees it will get the lower score than kidnapped. Example: *UN in Amman announce the number of Iraqi refugees have increased, and that most left in fear of recent terrorist blasts in recent months in Baghdad and elsewhere.*
4. If none of the above happens the score of 1 is given to the document like: *Protesters gather in Al Mijar al Kabier, Maysan to protest the poor services, in addition to bad agriculture and a lack of irrigation projects. OR Anbar Governor Qassim al-Fahdawi receives approval to have*

*the army forces protect some polling stations in cities across Anbar and a curfew will be imposed on Election Day.*

5. If the document is not about the violence at all it will get 0. exp: *Dust storms cover Kurdistan, and a hospital received a patient affected by the storms.*

At last, the scores for all the documents per day are added together and the result is the score of the date. The following table contains the documents for July 2013 and their scores.

<b>Date</b>	<b>Score</b>	<b>Key words associated with the document</b>
3-Jul-13	4	civilian target, Diyala, explosive
3-Jul-13	5	bridges, Muthanna, Ninewa, security target
3-Jul-13	5	civilian target, military, Ninewa, security target
8-Jul-13	0	Religious Holiday
11-Jul-13	0	Baghdad, foreign government relief, medical relief
11-Jul-13	0	crops, international trade
13-Jul-13	5	Anbar, border conflict
13-Jul-13	5	Baghdad, civilian target, explosive, religious
13-Jul-13	5	children, civilian target, Diyala, explosive
13-Jul-13	4	Anbar, civilian target, explosive
13-Jul-13	5	Baghdad, civilian target, explosive, men, youth
13-Jul-13	5	Baghdad, civilian target, explosive, leisure
14-Jul-13	5	civilian target, Dhi Qar, explosive, shopping
14-Jul-13	0	
14-Jul-13	5	civilian target, Diyala, explosive, roads
14-Jul-13	4	Anbar, civilian target, gunfire
14-Jul-13	0	Republic Day
14-Jul-13	5	Basra, explosive, government target, political
14-Jul-13	4	explosive, Ninewa, police, security target
14-Jul-13	5	Baghdad, civilian target, explosive, roads
14-Jul-13	5	babel, civilian target, explosive, religious
14-Jul-13	5	civilian target, explosive, Karbala, religious, Shia
14-Jul-13	5	explosive, government target, Ninewa, roads
14-Jul-13	5	civilian target, dining, explosive, shopping, Wasit
16-Jul-13	4	Basra, civilian target, dining, explosive
16-Jul-13	5	Basra, government employee, gunfire
16-Jul-13	5	civilian target, Diyala, Explosive, men, youth
16-Jul-13	5	air attack, civilian target, security
21-Jul-13	2	Baghdad, civilian unrest, independent militias, prisons
21-Jul-13	0	Diwaniya, government relief, heavy rain
24-Jul-13	5	Baghdad, civilian target, Salah Ad Din
24-Jul-13	4	Babel, explosive, professional target
24-Jul-13	5	civilian target, roads, Salah Ad Din
24-Jul-13	4	Baghdad, explosive, government employee, government target, oil
24-Jul-13	5	civilian target, explosive, Ninewa, roads
24-Jul-13	5	Baghdad, civilian target, explosive
25-Jul-13	0	government relief, money, Syrian refugees

Table1: Ground truth generated using the articles for July 2013

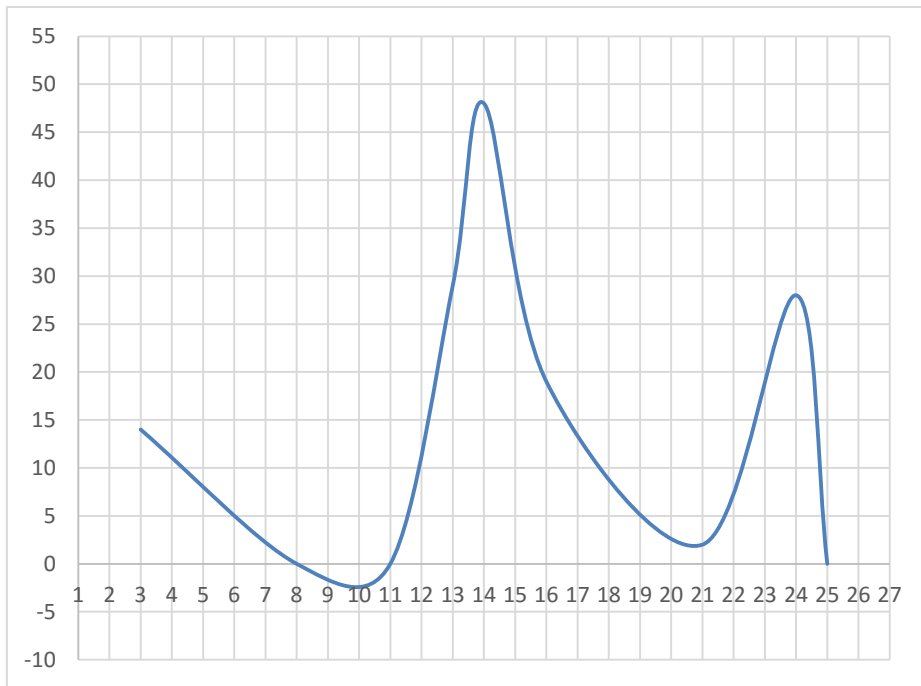


Figure5: Plot of ground truth for the period of July 2013

Now we compare the ground truth with our proposed methods for violence score generation using Word Embedding and Point Wise Mutual Information.

We here detect the peaks which indicate the violent days in the ground truth and our plotted graphs.

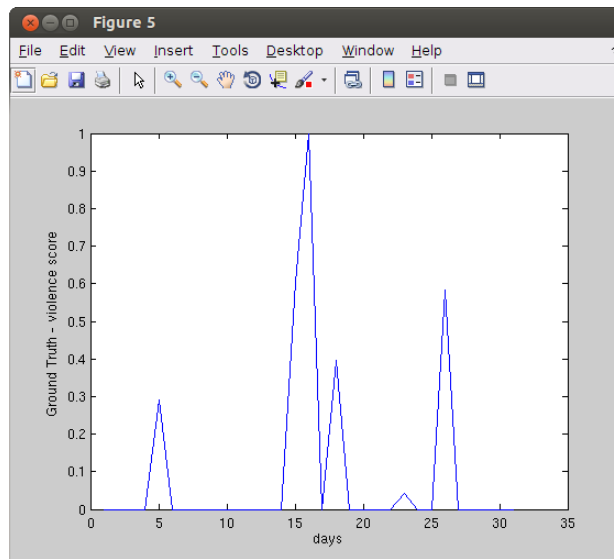


Figure 6: Ground Truth Violence Scores indicating the peaks on the days which had violent news articles

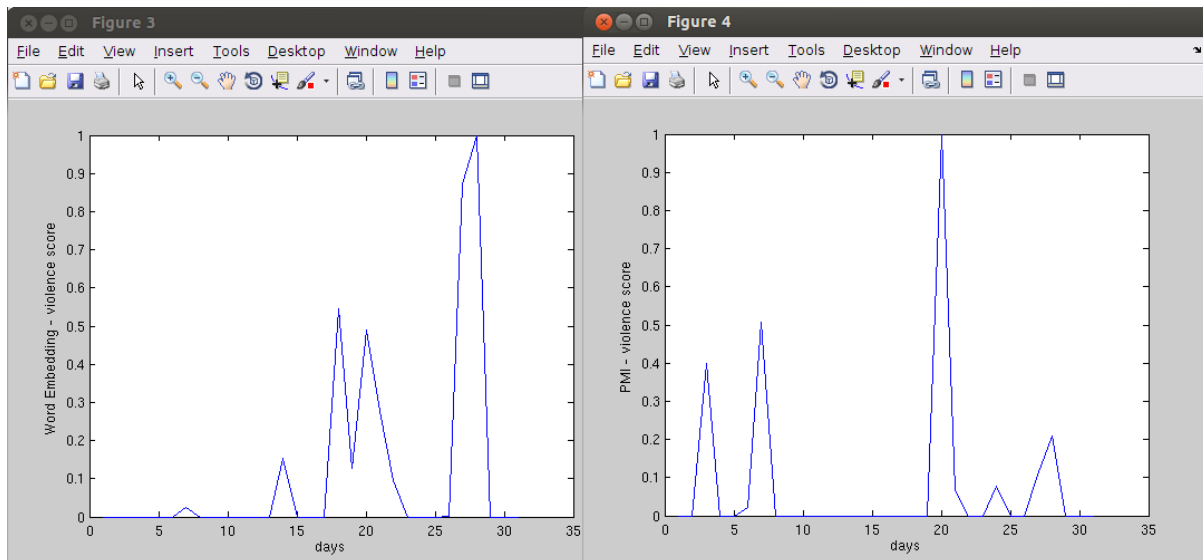


Figure 7: Plot of violence scores using Word Embedding (left) and PMI(right)

### Discussion of the results

We plot the graphs here to indicate the violence scores above. As it can be seen the trend of violence is similar. The days on which violence occurs in the ground truth are – 4 to 6, 14 to18 and 22 to 27.

For the word embedding we get results having violent days as 6 to 8, 12 to 15, 16 to 23 and 26 to 29. Here it appears as if the days are shifted by a factor of 2. This can be explained in the sense that the data used to generate ground truth was the captured date and our proposed method used publication dates, so its shifted by a factor of 2-3 days

For PMI, we get the violent days as 2-4, 6 to 8, 19 to 22, 23 to 25 and 26 to 29.

We are able to detect violent days and the extent of violence on those days better through the word Embedding case as it matches more with the ground truth in terms of the number of peaks. A better analysis would have been using Pearson correlation here, however due to simplicity of the project and insufficiency of data we avoid it here. Using Pearson correlation with more data would make more sense. And while computing Pearson correlation we also need to shift the dates by 2-3 days or use capture dates for both the ground truth and our program.

## 6. CONCLUSION

In this project we proposed a novel method for computing a factor score (here violence). The factor score was computed based on the fact that if a document has words semantically similar in meaning to the factor under consideration (violence here), then we say that particular document is talking more about that factor. We used this intuition to propose 2 methods to compute the violence scores for all documents. We then used each of the proposed methods (word embedding and Point wise Mutual Information) to generate a metric for computing violence scores. We compute the scores over a period of one month. We then analysed the scores and compare it with the ground truth that we generate based on human common sense. We find in from this project that word embedding perform better than Point wise Mutual Information.

### Findings from the Project

We believe this method can be generalized for computing the trend of any factor not only violence but also factors like infectious diseases, environmental threat, weather, infrastructure, science and technology, etc.

We can use this approach to analyse trends for the above mentioned factors. We just need to build a seed for each of the above factors and can then use our proposed methods in exactly the same way to analyse the factor's trend over a period of time. In this report for simplicity only one months' data was analysed. In practice one can easily analyse the trend for over several months from the sufficient data. Our algorithms can handle huge data up to a million of words in vocabulary and millions of text files. The bottle neck for our algorithm is in word2vec program which takes maximum amount of time. Other parts take do not much image.

### Limitations and possible extensions

The limitations of our proposed algorithm could be the following:

- Our method solely relies on the quality of the seed generated; if one generates seed of poor quality we may not get good results. So if one needs to consider a factor say 'relief' one needs to consider all possible words related to relief. This could be based on experts knowledge in that field.
- Another possible limitation of our project could be the amount of time taken to compute the word embedding. In this project we used a Ubuntu based system which had 12 cores and 24 GB RAM. So one clearly needs a strong system which can handle parallelization of loops in an efficient way. If the same program is run on a system with a single core one would increase the running time to roughly 12 times which is really slow.

Possible Extensions of this project include the following:

- Here we considered the data from just one month. Analysing data over several months is an interesting extension to this project. We have data for over 8 months and we plan to extend out proposed approach over these months analyse the trends for violence over this period of time.
- For the simplicity of the project we considered only analysing one factor – 'Violence'. One can extend our proposed approach to analyse various factors like infectious diseases, environmental threat by generating an appropriate seed for it.

- Generation of the seed can be altered to see if there is an improvement in the analysis. We can use the data from the corpus to build a seed which is better than an expert human could generate based on his knowledge.
- One can alter the way in which the factor scores are generated maybe giving more weightage to the number of documents talking about violence.
- Computing the Pearson correlation with more data to see which method correlates more mathematically gives a more concrete proof to word embedding better than PMI.



## REFERENCES:

1. Mikolov T. et al (2013) Efficient Estimation of Word Representations in Vector Space. In *arXiv preprint arXiv: 1301.3781*
2. Bouma, G. (2009) Normalized (Pointwise) Mutual Information in Collocation Extraction. In *proceedings of the Biennial GSCL Conference*
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. *The journal of Machine Learning research* 3 (2003) 993-1022
4. Church K.W. and Hanks P. (1990) Word Association Norms, Mutual Information, and Lexicography in *Journal of Computational Linguistics* 16(1): 22-29 (1990)
5. Agrawal A. and An A. (2012) Unsupervised Emotion Detection from Text using Semantic and Syntactic Relations, proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'12), Macau, China, December 4-8, 2012. 346-353
6. Davoudi H. (2014) Bringing insides into a Corpus: Topics Trends Detection and Labelling. Project report at School of Computer Science and Engineering, York University.
7. Piskorski, J. et al. (2011) Online News Event Extraction for Global Crisis Surveillance, *Lecture Notes in Computer Science, Volume 6910, 2011*
8. Deeplearning for java library – [deeplearning4j.org](http://deeplearning4j.org)

## APPENDIX

### User Manual

For this project implementation we have made our code publically available on github. The link to our code is: <https://github.com/raghavendersahdev/FactorAnalysis>

In this project we have written the following files:

- PreProcess.java – this file pre-processes all the files in a particular folder. This parses the xml data and extracts only the text content of each article and the publication date and time. We remove the punctuations, stop words and convert the corpus to lower case. We then save the pre-processed files in a new folder. We also give the user an option to remove the files whose content is not in English. However we do not remove the foreign languages files in our project.
- PreProcess2.java – this file pre-processes reads all the pre-processed files generated by PreProcess.java and stores them in a single text file. This is required as word2vec program expects a single file to learn the model for the vector representations of the word.
- RemoveInvalid.java – this file removes the invalid xml files which has illegal characters like &#. This program is required to pre-process huge corpuses because sometimes the xml files have illegal character sequences in them, so this program takes care of that.
- Word2VecRawTextExaample – this file has been modified from the code by [1],[10]. This file is responsible for computing the word vector representations from the single text file generated by PreProcess3.java. This files stores the word vectors in a text file word2vec.txt so that it can later be used by other programs. We also generate the vocabulary from this file.
- Similarities.java – this file reads the word vector representations and computes the cosine similarity of each of the seed words with each of the words in the vocabulary to generate a matrix of size seed\_words\_size x vocabulary\_size. In this file after generating the matrix we compute the violence score for each document based on the method explained in the report.
- GenerateScores.java – this files reads the document scores generated by the Similarities.java file and computes the violence score for each day. It stores the violence scores for each day in a text file from which we plot the graph of violence scores vs. days.
- PMI\_new.java – this file computes the Point wise Mutual Information between each of the seed words and the vocabulary words generated by the word2vec code. It then computes the violence scores for each document, these scores are then used by GenerateScores.java file to compute the violence score for each day.

It should be noted that the code is well commented and the API for the code can be generated using the javadoc generation option in eclipse IDE which clearly explains what functions are there in each class and what each class and its functions do.

### Descriptions of your System Design

In this part we explain how to run our program. So one should follow the following steps to execute our code:

Step 1: Run the PreProcess.java file to pre-process the files. Input to this program is the path to the corpus which contains all the files. Output path for the pre-processed files should be specified by the user.

Step 2: Run the PreProcess2.java to concatenate all the files generated in step 1 in a single text file.

Step 3: Run the Word2VecRawTextExample.java to compute the word vector representations.

Step 4: Run the Similarities.java to compute the scores for documents based on word embedding OR run the PMI\_new.java to compute the score based on PMI

Step 5: Run the GenerateScores.java to compute the violence scores for each day.

Data Structures used in the code were HashMaps, ArrayLists and arrays. HashMaps reduced the time complexity of the code drastically since the complexity of extracting an item from a hash map is  $O(1)$ . However it requires more dynamic memory on the RAM which is not a problem for us.

### Time Complexity of the Program

The following codes were run on an ubuntu machine with 24GB RAM and 12 cores.

- PreProcess.java – took 120 seconds for 35,312 files
- PreProcess2.java – took 7 seconds for 35,312 files
- Word2VecRawTextExample.java – took 300 seconds for 35,312 file
- Similarities.java – took 40 seconds for 35,312 files
- GenerateScores.java – took 1 second for 35,312 files

### Sample Input and Output

A sample input to our program is the path to the corpus. Sample outputs are

- Vocabulary of words
- Violence scores for each day
- Plot of violence scores vs. days (shown in the plots earlier in the report)

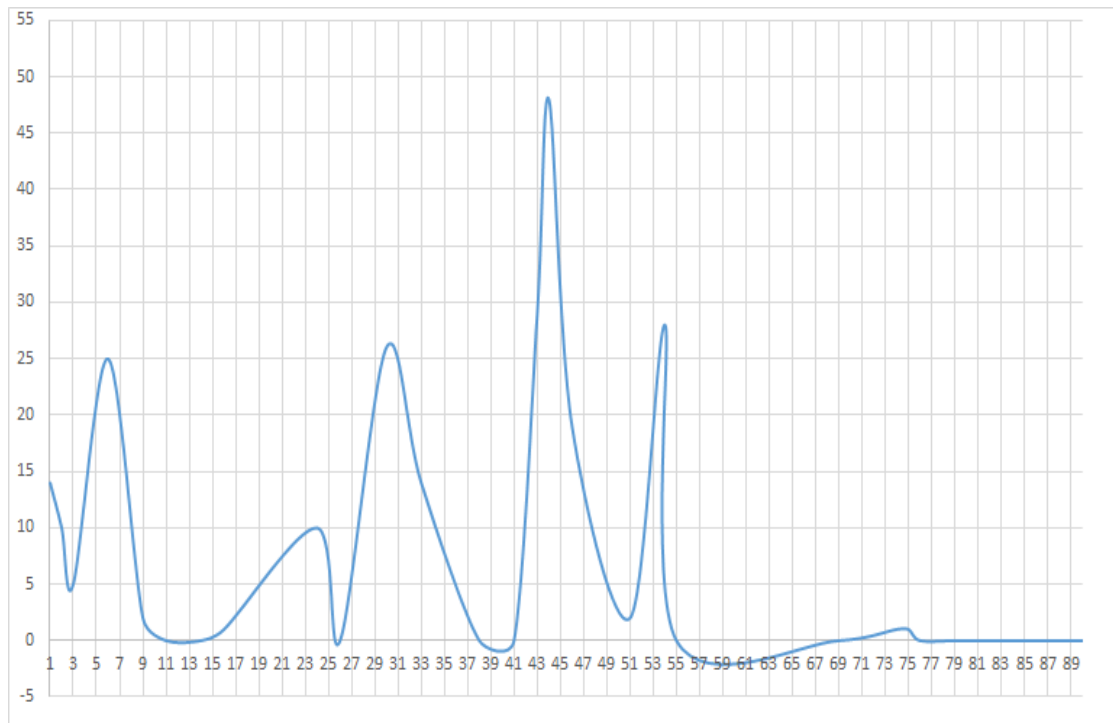
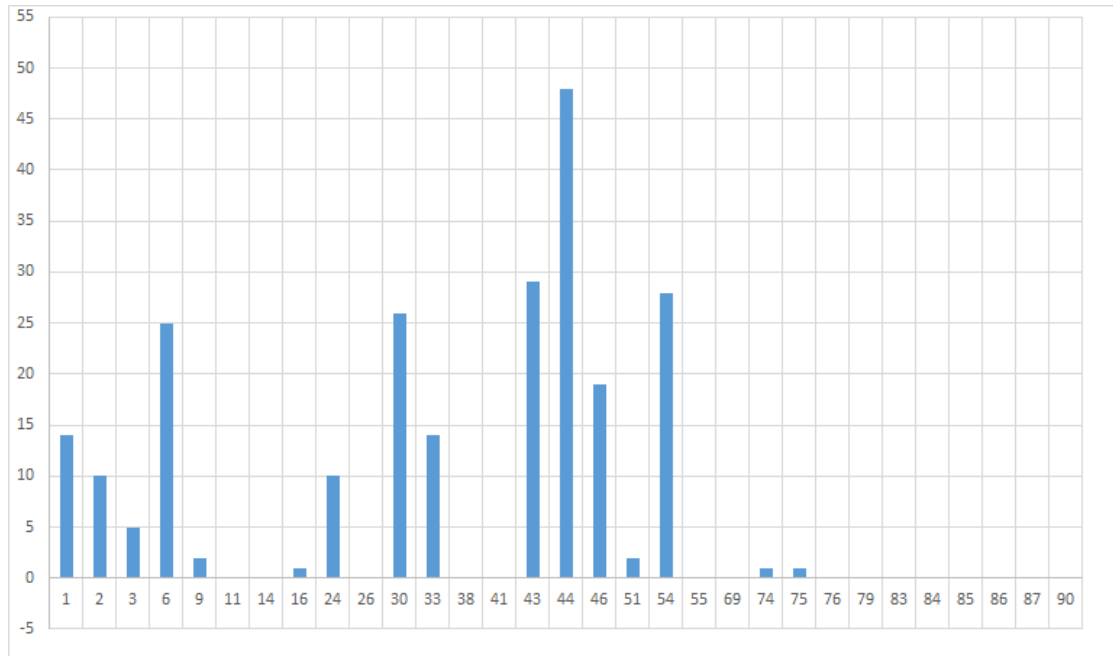
### Limitations of the software

The amount of RAM usage for 300,000 files is about 12GB while running the Word2VecRawTextExample.java, so this might be a problem for computers having lesser RAM.

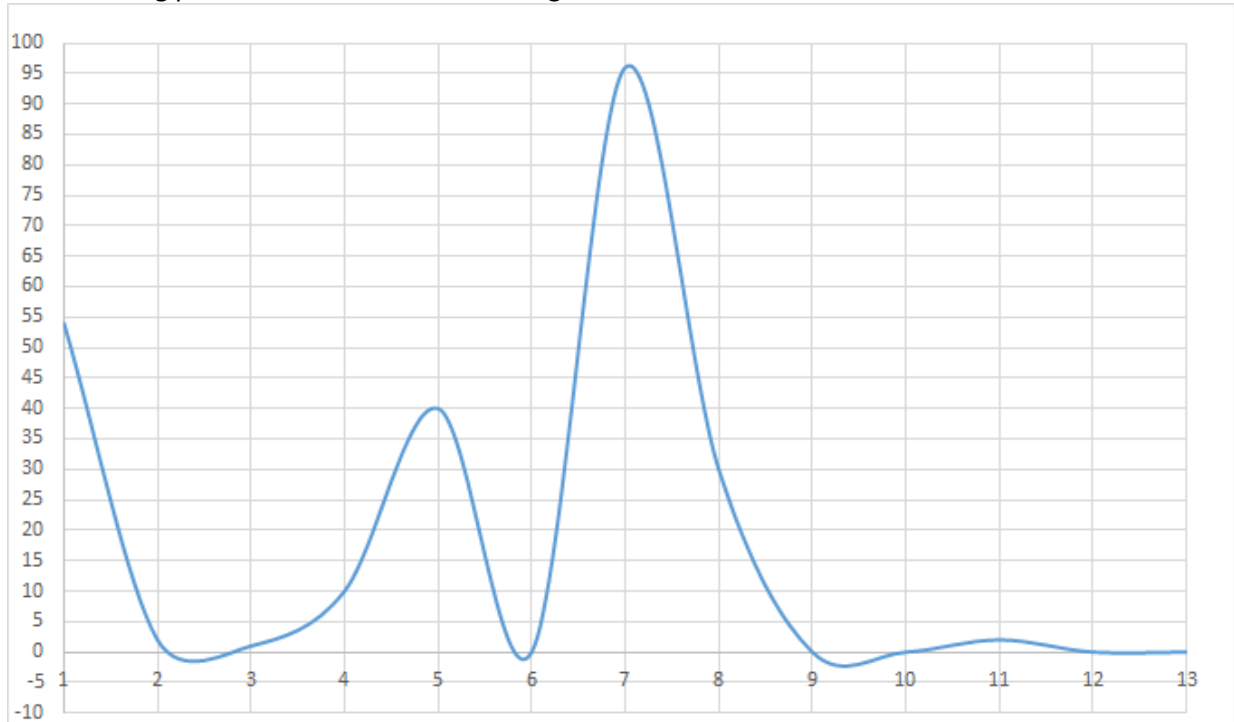
**SOME INTERESTING FUTURE WORK (extra optional reading)**

We additionally include the plots of ground truth generated for 3 months. Some interesting future work involves detecting peaks obtained below from our corpus.

The following plot shows the score of violence against the factor of time. The vertical axis shows the violence value and the horizontal one shows the date which ranges from June-August 2013 (90 days).



The following plot shows the violence score against weeks.



The following plot shows the violence score against month (3 months).

