SURPRISING DISCOVERIES FOR ONLINE HEALTH INFORMATION

Monica Kuppuswamy
Nikhil M Hegde
Venkatesh Umamaheswaran

Abstract of the Project:

The project aims at developing computational approaches (association mining, text modeling, and machine learning techniques) to identify “surprising” news from a news corpus. “Surprise” is defined as a divergence from an expectation or a low likelihood of an occurrence according to an expected likelihood. The dataset here is about the health information data specifically related to the diabetes disease.

Surprise might be general to society or personalized. General surprise is something that violates the common knowledge of the entire society whereas personalized surprise is just for a particular person based on this person’s background knowledge. Personalized surprise is not necessarily surprising to society. Here in this project, personalized surprise was decided to be found from the corpus. There were three main approaches followed to find out the surprising core from the corpus.

1. Cosine Similarity using Pre-trained Word Vectors
2. Semantic Similarity API
3. LDA Topic Modeling
4. Word to Vec

Cosine Similarity with Pre Trained word vectors:

Calculating surprise score

In order to find the surprise score number of sentences talking out of context in a particular document was tried to be found. For that to happen, the cosine similarity between the sentences were found. But the problem was that a simple cosine similarity between the two sentences would not yield you the correct similarity score of the sentences. Then it was decided to add more context to the sentences by adding more similar words to a word in a sentence. The Dataset given was of medical domain specifically Diabetes. A similar word list was needed if given a particular word. A pre trained Word Vector which was trained on medical documents was obtained. The size of the word vector was 4.42 GB. For every word in the sentence, the related word vectors was obtained and then the cosine similarity between these sentence vectors was calculated.

The cosine similarity between consecutive sentences for a particular document was calculated and then the 25 percentile of the maximum of the cosine similarity of two sentences for a particular document was calculated. After that, the threshold was obtained, and then two consecutive pairs of scores which are lesser than the threshold was found. More explanation is given below.

So let's say that for a particular document, the below listed scores for each consecutive sentence pairs.

Sentence 1 Index	Sentence 2 Index	Cosine Similarity Score
1	2	0.839877
2	3	0.530932
3	4	0.688539
4	5	0.246744
5	6	0.221123
6	7	0.440131

From the above table it is visible the maximum score is 0.739877.
Threshold = 0.266744
There was also a condition where if the threshold chosen was greater than 0.4 then the 0.4 was chosen as the threshold.
So now when comparing the scores to the threshold for consecutive pairs, we get that sentence pairs (4,5) and (5,6) scores are consecutive and lower than the threshold. The only common sentence in this pair is sentence 5.
For the whole document there is one sentence which is talking out of context. So the surprise score for this particular document is ⅙ or 0.16.
The above steps were repeated for all the documents in the corpus and below are the results:

Results and Validation

Total documents in the corpus:	10455
Maximum number of surprising sentences in a document:	17
Maximum surprise score of a document:	0.222

Total documents having surprising content:	3088
Average surprise score of 3088 docs:	0.0942
Percentage of information talking out of context:	2.87%

Validation was done manually by involving team members and friends.
Around 20 sentences which were talking out of context was chosen and then went to that particular document and read the document and tried to find out why it was classified as out of context.
Below are the results from the validation:

Total Number of sentences validated = 20
Correctly classified as out of context = 14
Incorrectly classified as out of context = 6

Using the semantic similarity API

There is semantic similarity API service which gives you similarity between two sentences. This API involves making a web request to API endpoint and getting the similarity of the two sentences. We used the same logic for calculating the surprise score as in the Cosine similarity using pre trained word vectors. Each sentence was sent to the API service and the score was noted.

This API was based on a model on distributional similarity and Latent Semantic Analysis (LSA). Semantic relations extracted from WordNet.

For two consecutive sentences in the document the similarity was obtained and recorded in this format.

Sentence 1 Index	Sentence 2 Index	Cosine Similarity Score
1	2	0.339877
2	3	0.530932
3	4	0.688539
4	5	0.246744

The similarity score between consecutive sentences for a particular document was calculated and then the 25 percentile of the maximum of the cosine similarity of two sentences for a particular document was calculated. After the threshold was obtained, and then two consecutive pairs of scores which are lesser than the threshold was found.

The comparison of results was recorded in a csv and this is the result.

doc	**total count**	**surprise count**	**lines**	**surprise score**	**threshold**
221230.txt	12	1	7,8|	0.0833	0.1653
95765.txt	22	2	0,1|19,20|	0.0909	0.2470
147119.txt	23	1	0,1|	0.0435	0.2479
133868.txt	20	1	0,1|	0.0500	0.2521
221218.txt	11	1	4,5|	0.0909	0.3016
186779.txt	9	1	2,3|	0.1111	0.1703
208107.txt	14	2	1,2|8,9|	0.1429	0.2954
156202.txt	18	2	0,1|1,2|	0.1111	0.2070
228016.txt	16	1	2,3|	0.0625	0.3257
112454.txt	19	2	9,10|10,11|	0.1053	0.2420

The lines column signifies the index of the sentence in the document.

For a sample of 1349 docs, 620 documents were classified as that they have surprising words.
Surprising information in 1349 docs: 4.96%

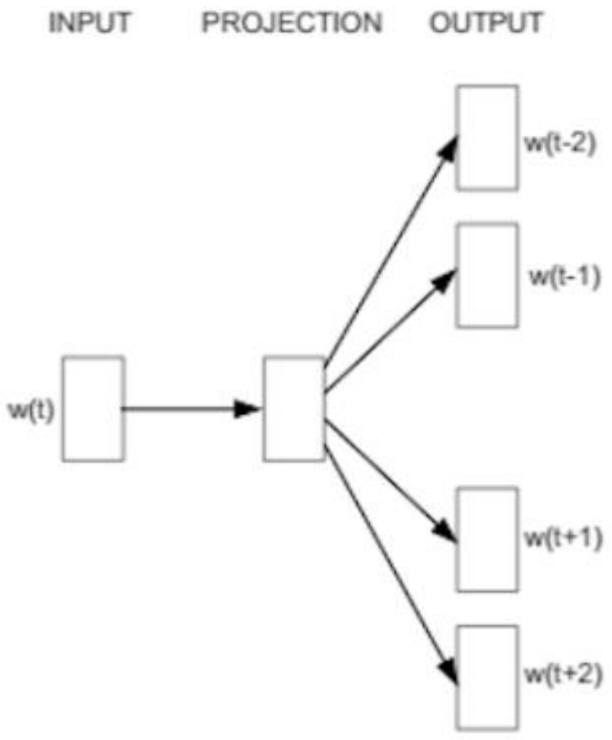
Word to Vector representation

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand.

In order to see if there would find any surprising information, Word2Vec was been run on our corpus using a skip-gram model.

Word2vec is similar to an autoencoder, encoding each word in a vector, but rather than training against the input words through reconstruction, as a restricted Boltzmann machine does, word2vec trains words against other words that neighbor them in the input corpus.

It does so in one of two ways, either using context to predict a target word (a method known as continuous bag of words, or CBOW), or using a word to predict a target context, which is called skip-gram. The latter method was used because it produces more accurate results on large datasets.



Skip-gram

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 the numbers in the vector.

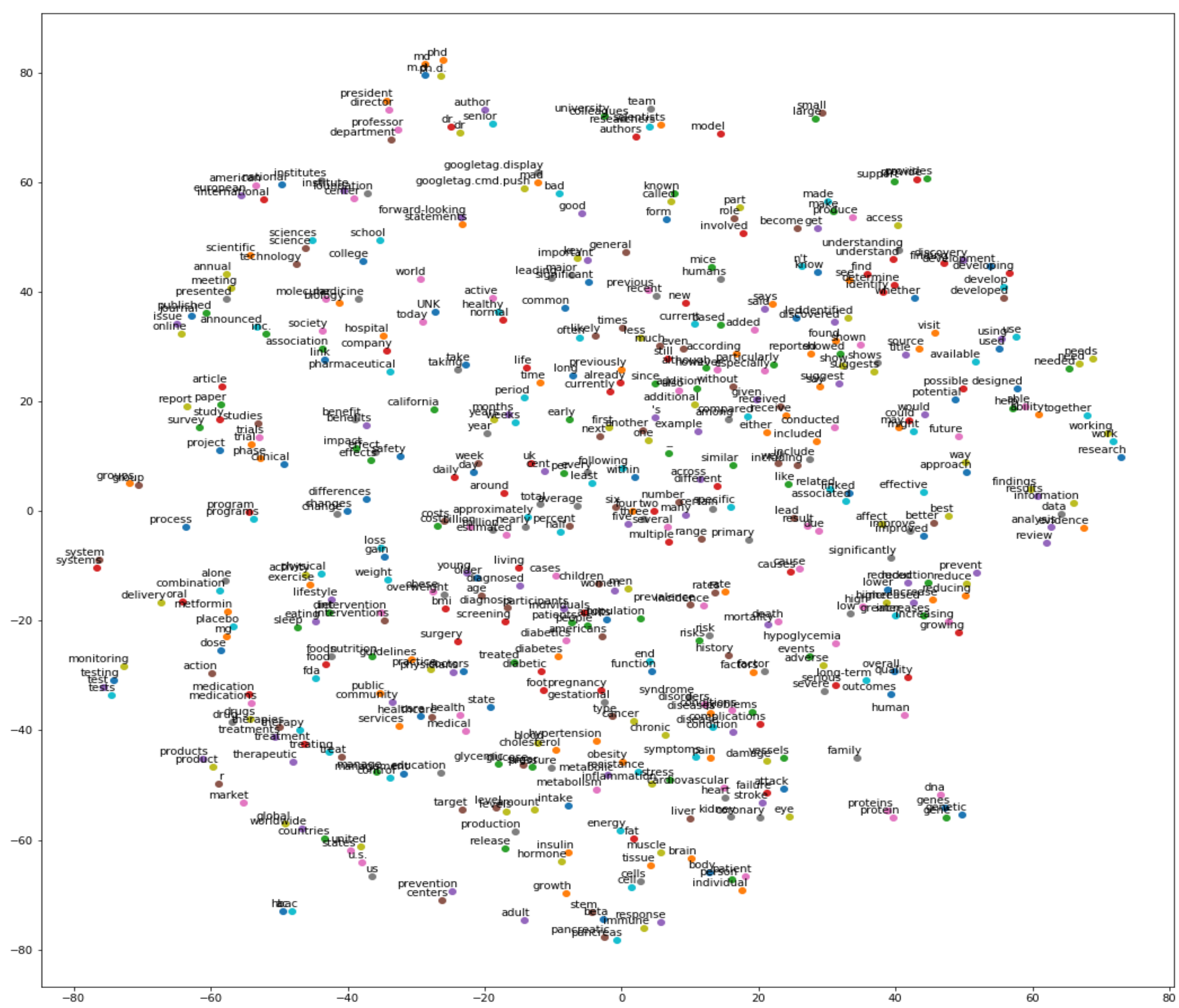
So a well trained vector would place similar words close to each other.

The word2vec training was done using the help of tensorflow in Python. We used the dataset to come up with the embeddings.

Some of the constants used for this training is listed below:

Number of steps: 500000
Number of skips: 2
Maximum number of skips: 2

The final embeddings that is obtained is of very high dimension. So we need to use TSNE. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. Using sklearn library which has an implementation of tsne, the word embedding was visualized. Below is the result of the visualization.



LDA Topic Modeling:

Topic Modeling Using LDA and finding document similarity:

In LDA, each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently. In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics. This is performed for every document in the entire corpus to retrieve a list of topics from the corpus.

Bag of Words:

Also known as vector space model or document-term matrix. In this model, a text (such as a sentence or a document) is represented as the bag(multiset) of its words. By calculating the frequency of each term with the document while assigning a unique id to each token and collecting corresponding word counts, we generate a list of vectors which is equal to the total number of documents where each vector contains the unique token id and the corresponding occurrences in the document.

Constructing the model:

With the help of the calculated document-term matrix, a LDA model was constructed using the python genism and the top 20 topics from a set of 10k+ documents were extracted. The topics retrieved from this model is represented as probability distribution of the most frequent words with the topic.
The topics extracted from the health data corpus include:

LIST OF TOPICS EXTRACTED FROM HEALTH DATA
(0, 0.009*"peopl" + 0.009*"physic" + 0.007*"uk" + 0.007*"per" + 0.006*"year" + 0.005*"manag"')
(2, '0.024*"diabet" + 0.022*"health" + 0.011*"bt" + 0.010*"liraglutid" + 0.009*"pharmacist" + 0.009*"patient" + 0.008*"work" + 0.008*"associ" + 0.007*"peopl" + 0.007*"new"')
(3, '0.027*"diabet" + 0.017*"insulin" + 0.011*"eye" + 0.010*"studi" + 0.009*"type" + 0.009*"r" + 0.008*"vision" + 0.008*"patient" + 0.008*"year" + 0.007*"blood"')
(4, '0.018*"diabet" + 0.011*"diseas" + 0.011*"patient" + 0.008*"clinic" + 0.008*"trial" + 0.008*"studi" + 0.007*"result" + 0.007*"product" + 0.006*"type" + 0.006*"insulin"')

(5, '0.011*"research" + 0.010*"cell" + 0.010*"diabet" + 0.009*"studi" + 0.009*"insulin" + 0.008*"drug" + 0.007*"type" + 0.007*"use" + 0.007*"medic" + 0.007*"univers"')
(6, '0.017*"diabet" + 0.014*"research" + 0.014*"islet" + 0.012*"cell" + 0.011*"studi" + 0.010*"transplant" + 0.009*"insulin" + 0.008*"glucos" + 0.007*"univers" + 0.005*"type"')
(7, '0.060*"diabet" + 0.010*"american" + 0.009*"associ" + 0.009*"peopl" + 0.009*"patient" + 0.008*"diseas" + 0.008*"drug" + 0.008*"new" + 0.008*"research" + 0.008*"treatment"')
(8, '0.034*"diabet" + 0.011*"children" + 0.011*"medic" + 0.011*"patient" + 0.010*"studi" + 0.009*"care" + 0.008*"health" + 0.008*"risk" + 0.007*"diseas" + 0.007*"year"')]

Results:

Feeding in each test document in to the constructed model trained with a set of documents, and thereby obtaining the similarity value between documents. Intuitively setting the expected likelihood value to be 0.7, the results obtained were as follows:

Document 1 Index	Document 2 Index	Similarity
1	2	0.22319322384513257
2	3	0.9984972510569807
3	4	0.35282021298690813
4	5	0.998796931418585
5	6	0.00024112211664730363

Inferences:

Comparing the similarity value obtained between documents with the expected likelihood value of 0.7, the following inference about each of the document is made:

As seen from the table, the similarity value between Document 2 and 3 is very high compared to those between Document 1 and 2. This implies that the document 1 seems to contain surprising news about health data.
Similarly, Document 4 seems to contain surprising news about health data as the similarity between 3 and 4 is below the expected likelihood value.

Challenges Faced:

- Very difficult to clean this type of data
- Not enough domain knowledge, owing to this evaluation wasn't possible.
- Sentence tokenize performance a little low.
- No single model to understand the context of a sentence

APPENDIX:

The source code for this project can be found in the src directory.

API request.py - File for making the web request to API service and get the similarity score of sentences
Cosine similarity.py - File for getting the cosine similarity of two sentences using pretrained word vectors and write output into csv file.
Phrase.py - separate file for calculating cosine similarity
Tsne.png - Word embeddings visualization
wordtoVec.ipynb - The python notebook for generating wordtovec model.
Topicmodeling.py - Python file for doing topic modeling.

References:

https://deeplearning4j.org/word2vec
http://swoogle.umbc.edu/SimService/index.html
https://radimrehurek.com/gensim/models/ldamodel.html