Boost
Surprising Discoveries for Online Health Information

ITIS/ ITCS - 6162 Knowledge Discovery in Databases (Fall 2017)

Project Group 1

Akash Singh | Ankit Luthra | Bhargavi Narayan Bhat | Sanket Dilip Shete

Table of Contents

What is Surprise?

Throughout this paper we work on differentiating ‘ordinary’ health news from ‘surprising’ news. Here Surprise can be more specifically defined as a divergence from a set baseline / threshold value in terms of quantifiable data, or a low likelihood of an occurrence according to an expected likelihood.

Creating a baseline

Combining all the documents in a corpus to single baseline document.
Setting threshold value for entropy

Approaches to find Surprise

- Combining the power of R + Python (NLTK) for data science
- Using capabilities from both Supervised and Unsupervised learning

Algorithms employed

- Tf-Idf
- LDA
- Clustering
- Representations: Barplot, ggplot, WordCloud

Unsupervised vs Supervised Learning

Our goal of text mining is to differentiate between various documents in the corpus so as to highlight obvious differences between ‘surprising’ and ‘ordinary’ health news presented. Our successful differentiation between the two categories can help address questions like: what kinds of things are mentioned in these documents? And is there any deviation presented in certain subset of all documents? Do the documents contain any ‘positively’ surprising news?

We can leverage unsupervised and/or supervised machine learning algorithms to help us with this differentiation. Supervised algorithms require data to train and test the model that is pre-coded (tagged) with the variable of interest. For example, we went through a finite and feasible number of documents manually and tagged keywords most commonly used to present surprising facts. Then our supervised algorithm uses this information to train a model. We then use our tagged data, to test the model to see how accurate it is.

In text mining problems such as this, where we are not able to tag much of data ourselves, we also employ unsupervised techniques. These techniques try to tell the difference between documents without any prior knowledge. It is to be noted that unsupervised methods are rarely comparable in accuracy to supervised methods.

However it would still be very difficult to conclude that, what is preferable out of the two methods.For our situation, where tagging 10000 documents is very challenging job, A quick unsupervised clustering algorithm can reasonably separate the documents. However in this case, we would need to go through and manually check the cluster results before placing any confidence in them. Not to be discouraged by this manual analysis, the initial split provided by the clustering algorithm would always speed up our workflow with such huge corpus.

In our analysis, we would combine the power of both supervised and unsupervised analysis on the given dataset.

Unsupervised Learning

1. Clustering in R
2. Enhancing clustering capabilities
3. Topic Modeling in Python

Preprocessing text in R

The given documents in “Diabetes” context are feeded into R to create a corpus. This corpus is cleaned and made ready for analysis using various techniques. Techniques involve converting corpus to lower text, removing special symbols like { , : / “ _ - } , stripping off whitespace, removing { punctuation, numbers, english vocab stop words} and stemming.

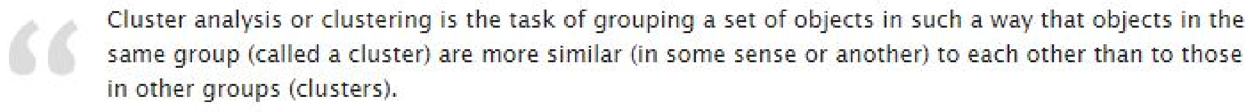
There were some non UTF characters in the corpus which gave us tough time to convert corpus into document term matrix (dtm). We got through it eventually, after trying a plenty of work arounds. Converting non UTF-8 charactaracters into their corresponding byte codes seems to help. Document term matrix will have close to 100% sparsity with relatively less non-sparse entries.

Consider the structure of the DTM. Very briefly, it is a matrix in which the documents are represented as rows and words as columns. In our case, the chosen subset of the corpus has ~100 documents and 5684 words, so the DTM is a 100 x 5684 matrix. It can be imagined as matrix describing a 5684 dimensional space in which each of the words represents a coordinate axis and each document is represented as a point in this space.

Converting dtm into matrix would pose as another challenge because a corpus with somewhat 10,000 files would pose a memory challenge as the matrix would look for close to 5 GB of memory. R provided function gc() for garbage collection and package Big Memory would help suppress somewhat but not in this case with such huge space required. So eventually, decision was made to split documents into chunks and processing them further. The following sections in R will showcase our data analysis with one of such chunk of documents.

Clustering

As we previously discussed, differentiating between the two approaches, To get a quick start on huge corpus such as this, we need to use algorithms that can classify documents automatically based on their structure and content. Here is a more formal definition from wikipedia:

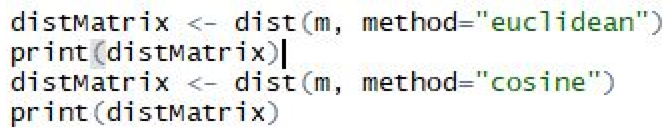

“ Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

Hierarchical clustering

Among the two types of Hierarchical clustering: Agglomerative and Divisive, we used hclust which performs agglomerative hierarchical clustering i.e. each document starts with its own cluster, however iteratively merges documents into cluster closest to each other until the entire corpus forms a single cluster.

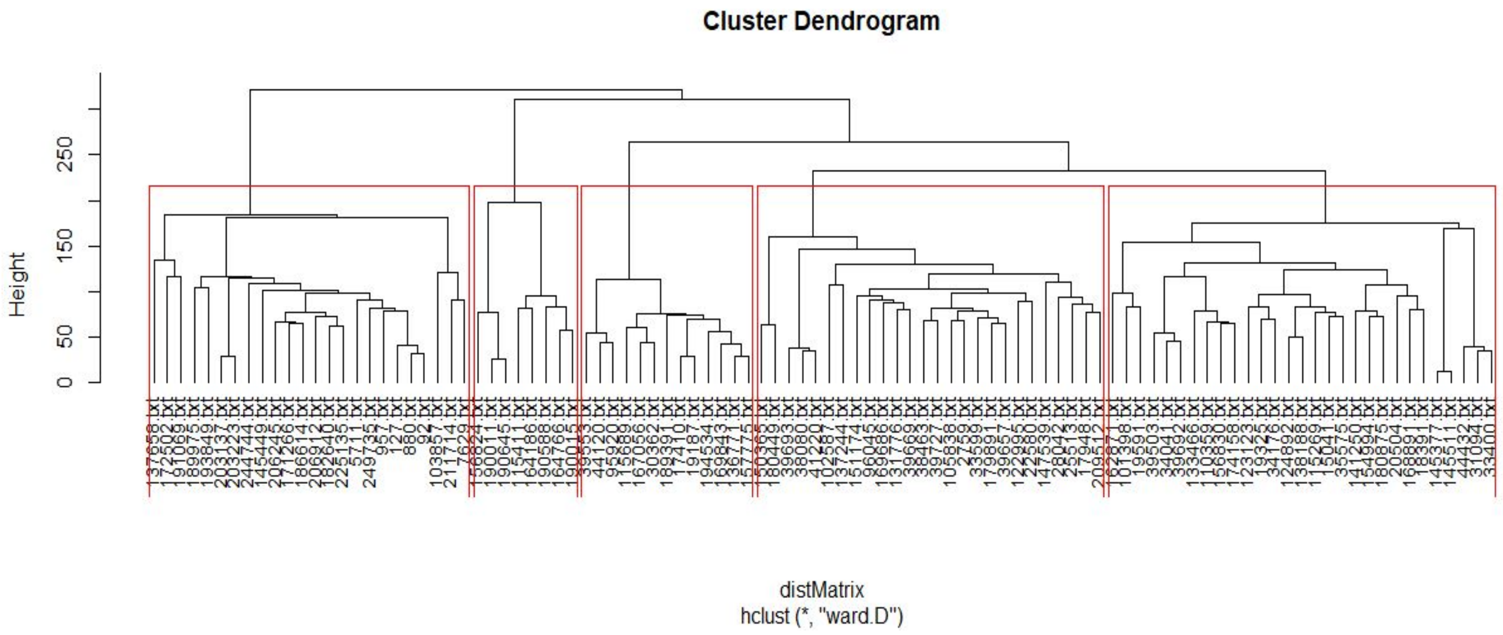
1. Assign each document to its own (single member) cluster
2. Find the pair of clusters that are closest to each other and merge them. So you now have one cluster less than before.
3. Compute distances between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until you have a single cluster containing all documents.

Before running the algorithm, we had to convert the DTM into a standard matrix which can be used by dist, the distance computation function in R (the DTM is not stored as a standard matrix). The distance matrices were computed and tried with different methods namely, “euclidean” and “cosine”.



```r
distMatrix <- dist(m, method="euclidean")
print(distMatrix)
distMatrix <- dist(m, method="cosine")
print(distMatrix)
```

Next we run hclust. The algorithm offers several options. We use a popular option called Ward’s method – Ward's minimum variance method aims at finding compact, spherical clusters. Finally, we visualise the results in a dendogram



Hierarchical clustering results

rect.hclust () gives us various cluster zones to extract out dissimilar documents in the chosen subset of the corpus. Fore example, the second cluster zone yields us with fewest similar documents in the subset. Clearly, the most well-defined clusters are those that have the largest separation; many closely spaced branch points indicate a lack of dissimilarity (i.e. distance, in this case) between clusters. Based on this, the figure reveals that there are 5 (k=5) well-defined cluster groups. Using trial method, we try different k values to recognize different cluster groups. K value 5 yields us with the most robust cluster grouping as it happens at large distance, and is cleanly separated (distance-wise).

K-means clustering

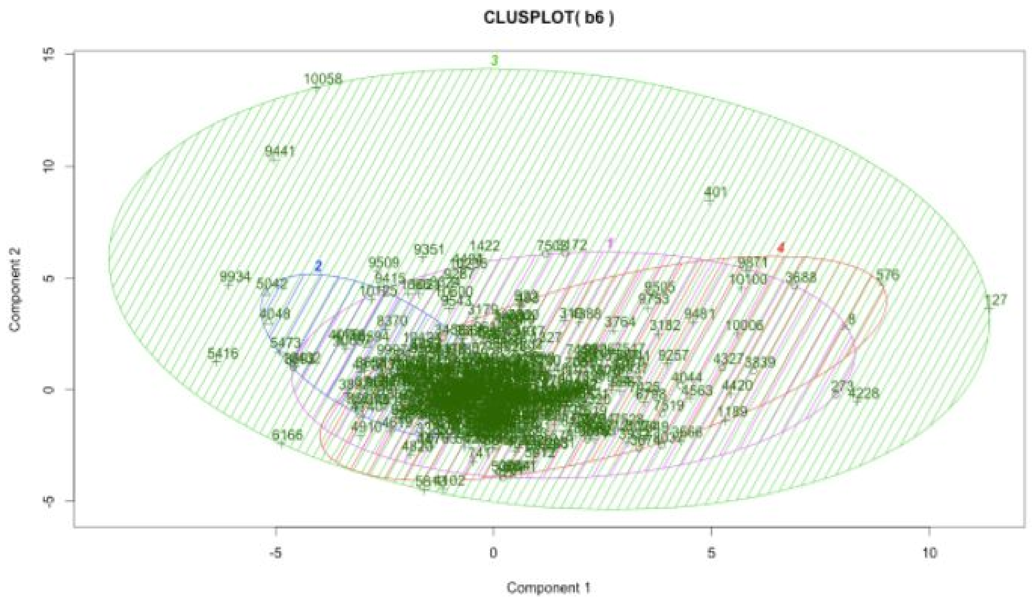
In contrast to hierarchical clustering, our next algorithm – K means – requires us to define the number of clusters upfront (this number being the “k” in the name). The algorithm then generates k document clusters in a way that ensures the within-cluster distances from each cluster member to the centroid (or geometric mean) of the cluster is minimised.

Steps Involved:

1. Assign the documents randomly to k bins
2. Compute the location of the centroid of each bin.
3. Compute the distance between each document and each centroid

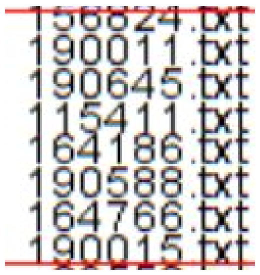
4. Assign each document to the bin corresponding to the centroid closest to it.
5. Stop if no document is moved to a new bin, else go to step 2.

Since the k means method has a limitation of generating local minimum rather than global minimum we ran the k means with different initial configuration values, a number of times. Consequently harmonizing results were found which were consistent with hierarchical clustering results.



Finding surprise through Clustering results

From the above clustering results, cluster group 2 was found further analysed to find surprising facts.


156824.txt
190011.txt
190645.txt
115411.txt
164186.txt
190588.txt
164766.txt
190015.txt

Surprise from clustered docs

115411.txt The study findings also showed that patients receiving dapagliflozin experienced greater

reductions in body weight compared to patients on placebo.

156824.txt
-- Patients who received linagliptin monotherapy were significantly more likely to achieve a reduction in HbA1c greater than or equal to 0.5 percent at 24 weeks than placebo (47.1 percent versus 19.0 percent).

164186.txt
Double blind Phase 3 clinical study, which demonstrated that the addition of the investigational drug dapagliflozin to existing glimepiride (sulphonylurea) therapy produced significant reductions in glycosylated hemoglobin levels (HbA1c) in adult patients with type 2 diabetes compared to glimepiride alone.

164766.txt
The number of individuals with any hypoglycemic event was significantly lower for patients treated with dapagliflozin plus metformin as compared to those treated with glipizide plus metformin (3.5% vs. 40.8% respectively;

Clustering + Bag of words

Our Bag of Words approach is detailed in Supervised analysis section. The resulting clustered group was scanned with bag of words approach to extract surprising facts. However this does not resulted into satisfactory results because of very few occurence of words from our chosen bag of words. To further deep dive into analysis we combined tf idf, word cloud strategies with clustering.

Enhancing Clustering

Initial DTM was enhanced by filtering out maximum term frequency as upper bound. This upper bound was created by ignoring overly common words. A product of length of corpus * 0.5 yields terms that appear in more than 50% of the documents.

DTM was further subjected to weight tf-idf to downweight the terms that occur frequently across the documents. This was done by computing the tf-idf statistics, as:
```
dtm <- weightTfIdf(dtm, normalize = TRUE)
```

Clustering as preprocessing for Supervised learning

Clustering proved to be substantial help in selecting important features to find ‘surprise’ during supervised analysis. From the k-means unsupervised learning model, generated clusters of related documents, were found as a convenient way to visually examine relations between variables.Most important based on the clustering results new feature can be generated which could increase the significance of an independent variable, and the overall interpretability of the model.

Term Frequency

Bag of words list with both least occurring as well as most frequent words were created to find ‘surprise’ from the subset. The subset resulted in 2063 terms which only occurred once in the corpus. We also tried constructing new DTM in contrast to our original DTM where overly common words were filtered out as upper bound.

Surprise using minimum frequency

206912.txt
Human insulin suppresses the mosquito immune system, according to a paper in the June Infection and Immunity.

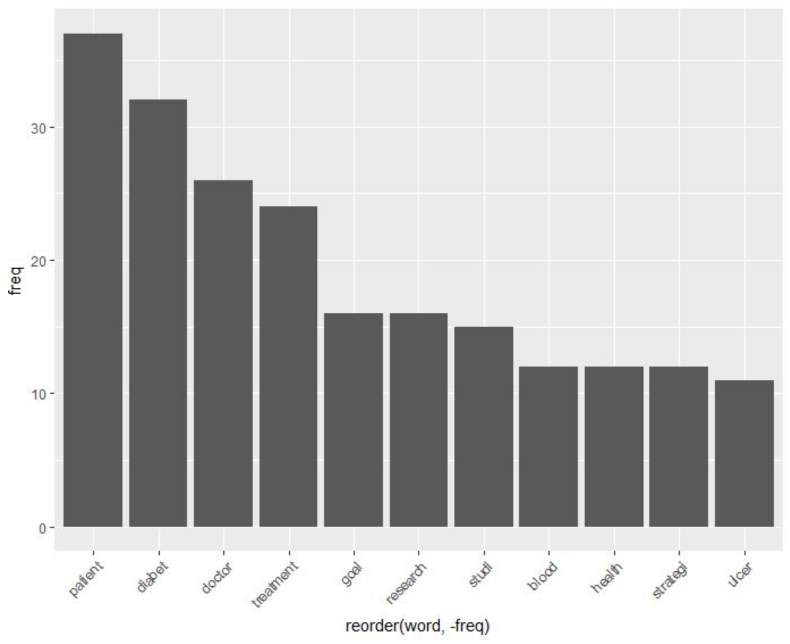
And while mosquitoes and malaria might seem to go together like baseball and hotdogs, mosquitoes' immunological resistance to the malaria parasite actually slows its spread among H. sapiens.

122995.txt
In angioplasty, blocked arteries are opened by inflating a small balloon inside a narrow vessel and often followed by the placement of a stent.

145449.txt
Working with the Marine Mammal Center in Sausalito, Calif., and other partners, scientists initially suspected a marine environmental cause of epilepsy by studying marine mammals and other wildlife with seizures that washed up on California beaches over the past decade.

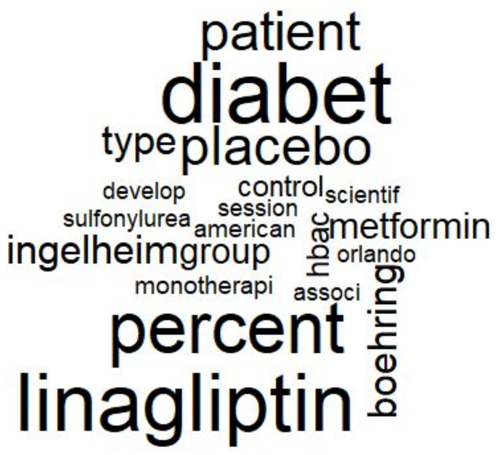
Surprise using max. Frequency terms

Bar Plot



Word cloud

representation for max occurrence



Topic Modeling in Python

Apart from clustering, we also performed another type of unsupervised analysis: topic modeling. We used the lda() function (Latent Dirichlet Allocation).

Unlike k-means which is a discriminative model (it tries to tell documents apart by conditioning on the contents of the document), LDA is a generative model (it creates a probabilistic model of how the words in each document were generated/written). LDA will determine which words are likely generated from a specific topic, then determine the topic of a document by examining these probabilities. LDA will also give a guess at the name of a topic. Like k-means, we need to supply the number of topics.

LDA Approach

Assume that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. LDA is an example of a topic model. And for each topic t, compute two things: 1) p(topic t | document d) = the proportion of words in document d that are currently assigned to topic t, and 2) p(word w | topic t) = the proportion of assignments to topic t over all documents that come from this word w.

After repeating the previous step a large number of times, we will eventually reach a roughly steady state where your assignments are pretty good. So use these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

Algorithm

1. Read the content of each file from corpus and store it in List
2. Remove stopword from list
3. Convert the list into dictionary.
4. Convert the dictionary into bag of words.
5. Pass these Bag of words as a input to LDA model to create LDA model of corpus-LDA_1.
6. Take the text file as a input and perform same operation as above to create LDA model of the test file-LDA_2
7. Pass LDA_1 and LDA_2 to function to calculate distance between LDA.

Applying Distance formula

Hellinger Diversion

Hellinger distance is a metric to measure the difference between two probability distributions. It is the probabilistic analog of Euclidean distance.

Given two probability distributions, P and Q, Hellinger distance is defined as a useful parameter when quantifying the difference between two probability distributions. For example, if you estimate a distribution for users and non-users of a service. If the Hellinger distance is small between those groups for some features, then those features are not statistically useful for segmentation

$$h(P,Q) = \frac{1}{\sqrt{2}} \cdot \|\sqrt{P} - \sqrt{Q}\|_2.$$

Kullback-Leibler Distance

The Kullback–Leibler divergence (also called relative entropy) is a measure of how one probability distribution diverges from a second, expected probability distribution. Applications include characterizing the relative (Shanon) entropy in information systems, randomness in continuous time series, and information gain when comparing statistical models of inference.

In contrast to variation of information, it is a distribution-wise asymmetric measure and thus does not qualify as a statistical metric of spread. In the simple case, a Kullback–Leibler divergence of 0 indicates that we can expect similar, if not the same, behavior of two different distributions, while a Kullback–Leibler divergence of 1 indicates that the two distributions behave in such a different manner that the expectation given the first distribution approaches zero. In simplified terms, it is a measure-of-surprise, with diverse applications such as applied statistics machine learning.

Jaccard Distance

The Jaccard similarity index (sometimes called the Jaccard similarity *coefficient*) compares members for two sets to see which members are shared and which are distinct. It’s a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations.

Although it’s easy to interpret, it is extremely sensitive to small samples sizes and may give erroneous results, especially with very small samples or datasets with missing observations.

Formula:

$J(X,Y) = |X \cap Y| / |X \cup Y|$

Significance of Percentage

1. This percentage tells you how similar the two sets are.
2. Two sets that share all members would be 100% similar. the closer to 100%, the more similarity (e.g. 90% is more similar than 89%).
3. If they share no members, they are 0% similar.
4. The midway point — 50% — means that the two sets share half of the members.

Python functions used

1. **doc2bow(text)**: Convert document (a list of words) into the bag-of-words format = list of (token_id, token_count) 2-tuples. Each word is assumed to be a tokenized and normalized string (either unicode or utf8-encoded). No further preprocessing is done on the words in document; apply tokenization, stemming etc. before calling this method.

2. **gensim.models.ldamodel.LdaModel**: This module allows both LDA model estimation from a training corpus and inference of topic distribution on new, unseen documents. The model can also be updated with new documents for online training.

Threshold Value

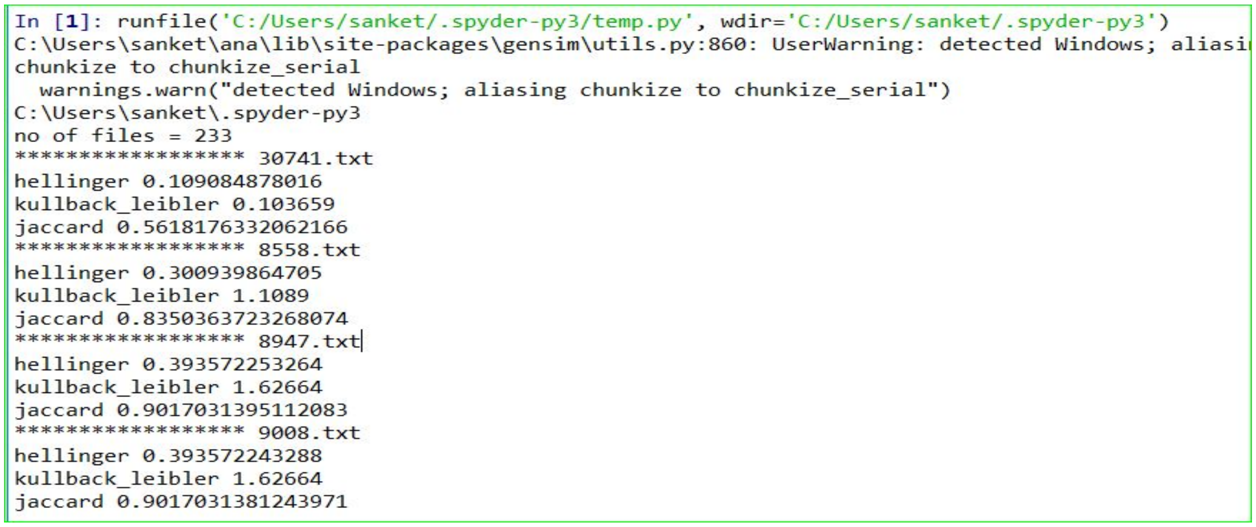
In this case we are calculating the diversion of a document from a base corpus.That's why If the distance is greater than 0.6 then we call that document as a surprise document.

For example in the output 8947.txt is a surprising document with surprised fact that

“ The results of immunohistochemical staining suggest that these new beta cells are not derived from duct cells. Rather, the beta-cell growth in insulin-resistant states occurs by "epithelial-to-mesenchymal transition," a mechanism in which cells take on a more primitive form and begin replicating. ”

Output

As we can see below in the output we get values of three different formulae as mentioned above. Furthermore we came to conclusion that in our case jaccard distance formula give better result than other distance formulae.



```
In [1]: runfile('C:/Users/sanket/.spyder-py3/temp.py', wdir='C:/Users/sanket/.spyder-py3')
C:\Users\sanket\ana\lib\site-packages\gensim\utils.py:860: UserWarning: detected Windows; aliasi
chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
C:\Users\sanket\.spyder-py3
no of files = 233
****************** 30741.txt
hellinger 0.109084878016
kullback_leibler 0.103659
jaccard 0.5618176332062166
****************** 8558.txt
hellinger 0.300939864705
kullback_leibler 1.1089
jaccard 0.8350363723268074
****************** 8947.txt
hellinger 0.393572253264
kullback_leibler 1.62664
jaccard 0.9017031395112083
****************** 9008.txt
hellinger 0.393572243288
kullback_leibler 1.62664
jaccard 0.9017031381243971
```

TF-IDF Approach

Finding similar documents

How do I find documents similar to a particular document?

Another approach is to look at a term's *inverse document frequency* (idf), which decreases the weight for commonly used words and increases the weight for words that are not used very much in a collection of documents. This can be combined with term frequency to calculate a term's *tf-idf* (the two quantities multiplied together), the frequency of a term adjusted for how rarely it is used.In information retrieval, tf-idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

Algorithm

1. Create some test documents.
2. Use NLTK to tokenize.
3. A document will now be a list of tokens.
4. Create a dictionary from a list of documents. A dictionary maps every word to a number.
5. Create a corpus. A corpus is a list of bags of words. A bag-of-words representation for a document just lists the number of times each word occurs in the document.
6. Create a tf-idf model from the corpus.
7. Create a similarity measure object in tf-idf space.

Function used from gensim

Gensim.similarities.similarity: The main class is Similarity, which builds an index for a given set of documents. Once the index is built, you can perform efficient queries like “Tell me how similar is this query document to each document in the index?”. The result is a vector of numbers as large as the size of the initial set of documents, that is, one float for each index document.

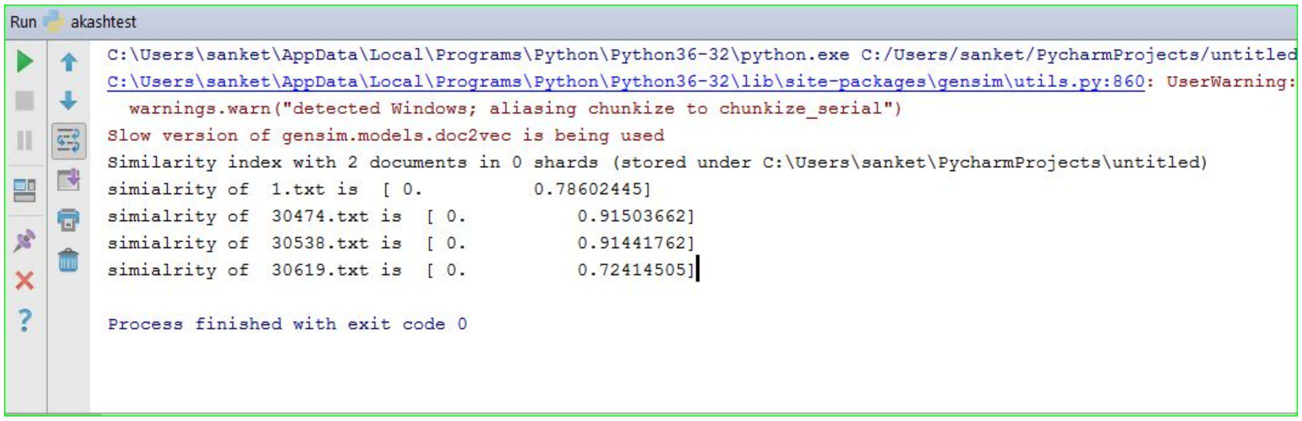
Threshold Value

After testing on many document we come up with the threshold score of 75%.If the similarity value of document is less than 0.75, then we call that document as a surprised document.

Output

In the output as shown below we can see that tf-idf of document 30619.txt is less than other file which shows the below surprising fact,

“Peak incidence occurs during puberty , around 10 to 12 years of age in girls, and around 12 to 14 years of age in boys.”



```
C:\Users\sanket\AppData\Local\Programs\Python\Python36-32\python.exe C:/Users/sanket/PycharmProjects/untitled
C:\Users\sanket\AppData\Local\Programs\Python\Python36-32\lib\site-packages\gensim\utils.py:860: UserWarning:
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
Slow version of gensim.models.doc2vec is being used
Similarity index with 2 documents in 0 shards (stored under C:\Users\sanket\PycharmProjects\untitled)
simialrity of  1.txt is  [ 0.          0.78602445]
simialrity of  30474.txt is  [ 0.          0.91503662]
simialrity of  30538.txt is  [ 0.          0.91441762]
simialrity of  30619.txt is  [ 0.          0.72414505]

Process finished with exit code 0
```

Supervised Learning

1. Bag of Words
2. Naive Bayes Model

In order to conclude to more prominent findings of surprise we further try to propose supervised analysis. As we know, These methods are only applicable if we have tagged data and In practice, tagging ~10,000 documents is a challenging task also counting the factor accuracy is very important.

To demonstrate it we took a subset of 10 randomly selected documents and split our data into training and test sets in supervised analysis - both of which need to be tagged. We first train a model on the training data, then apply it to the test data and see how well it did (otherwise, we could just be overfitting to the training data). If our data is sufficiently large and representative, our accuracy numbers should give us a good idea of how well the model will perform on data we don't have.

A good rule of thumb is to use 80% of the data to train and 20% to test. In our case, that is 8 for training, 2 for test. We will need to reprocess the data and randomize it to make sure we get a good split. The 10 documents were read through during brainstorming session within our group to classify them as 'surprising' and 'ordinary'.

Bag of Words approach in R

We used the bag of words model to simply represent natural language processing and information retrieval (IR). In general context, this model utilizes a text (such as a sentence or a document) is represented as the bag of words, disregarding grammar, punctuation and even multiple occurence. During a brainstorming session we came out with list of words commonly used to describe human surprising situation. This can further used for document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

List:

surprisingly	bombshell	revelation	amazing	eye-opener
stupefy	awakening	flabbergast	Bowl 0ver	Shake up
stun	astound	open-mouthed	startle	Breath away

Stemming was performed on bag of words to create a vector list of stemmed "surprising" words.
This looked like {"surpris", "amaz", "revel", "startl", "astoun", …………}
A basic scan was initially performed using Grep and Substring functions to extract meaningful sentences, which gave us beneficial results.

Output from R console:

```
 90387.txt14
"In the new study, Masahiko Inouye and colleagues point out that scientists have tried for
years to develop artificial versions of DNA in order to extend its amazing information
storage capabilities."

95379.txt19
 "That is why we were amazed to find that in cells lacking either glutathione or
glutathione peroxidase 4, a distinctive signaling pathway is engaged, which causes cell
death."

98863.txt22
 "In conclusion, Ms Addington said: \"None of this amazing research could happen without
the generosity of our supporters."
```

Naive Bayes Model

After pre-processing and DTM creation in R, we tried to implement Naive Bayes Model to extract classifier from the 8 documents [1:8] and predict the rest 2 [9:10].We calculated the confusion matrix and the recall accuracy.

```
recall_accuracy(as.character(x.rand$V1[9: 10]), as.character(predicted))
```

```
## [1] 0.4507
```

In a confusion matrix, the rows represent the actual group of the data, while the columns represent the predicted group of the data. We observed that it classified document 9 as surprising however it was marked as non.

To quantify the results another way, the recall accuracy tells us: of the documents that were truly in a given class how many were correctly labeled in that class by the algorithm. Calculating precision is the next challenge in this proposal. A third measure, F1, combines precision and recall. A major benefit of supervised analysis is that there are many different machine learning algorithms that can be applied.

Future Scope: Tree Next using RTExtTools

Conclusion

The methods explored during this analysis and described in the report (Classification, clustering, and topic modeling) should be enough to get us moving in the right direction in text mining given dataset for surprise finding. If you were previously unfamiliar with any of the topics mentioned here (especially those related to machine learning), it may be worth your time to do some additional research into the specific assumptions and limitations of each method/algorithm.

Challenges

1. Results obtained from different approach are very different, and difficult to harmonize.
2. Technical challenges like memory issue hinder mining a huge data set such as the one given here.
3. Running and testing the model on large datasets has high time complexity
4. Certain words like “increase” and “decrease” occur in same context and hence are similar, but have opposite meaning.

Future Scope

Extensive supervised analysis and Tree Next using RTExtTools.

References

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

https://www.oreilly.com/learning/how-do-i-compare-document-similarity-using-python

http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/

http://data-analytics.net/cep/Schedule_files/Textmining%20%20Clustering,%20Topic%20Modeling,%20and%20Classification.htm

https://de.dariah.eu/tatom/feature_selection.html

https://rpubs.com/saqib/DocumentClustering

https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781783551811/5/ch05lvl1sec31/document-clustering https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/lda.html

https://gerardnico.com/wiki/natural_language/tf-idf

http://tidytextmining.com/
