# SURPRISING DISCOVERIES

*(FOR ONLINE HEALTH INFORMATION)*

Group 4

**Ahmad Al-Doulat, Bhanu, Dilip, Monisha, Naishad**

*University of North Carolina at Charlotte*

## Abstract

Life is full of surprises, from bumping into a friend from home while on holidays, to arriving at a surprise party, to opening an amazing birthday gift, or hitting pay dirt on that 100-1 racehorse. Surprise has been researched since Darwin's time, perhaps because it involves an interesting mixture of emotion and cognition. Imagine that you walk into your house and the walls have changed color from the color they were this morning. If you have no explanation for this turn of events, then you would probably be surprised by this outcome. Many outcomes are surprising, the question is why? Our answer is that outcomes are surprising when they are hard to explain or when they weren't expected but valued. Specifically, that surprise is a meta-cognitive sense of the amount of explanatory, mental work that was carried out to establish coherence between unfolding events in the world. In this project, we programmatically analyzed a diabetes dataset of 10455 articles. Applied LDA and analyzed the outcome using K-means using most significant features. We have conducted a user survey to evaluate our results.

## Introduction

When a person reads the article, he/she has some expectations on what information they would be getting to know in a particular article. This expectation usually differs from person to person based on the domain knowledge they have. When something appears on the article which is deviated from the readers expectation it is considered a surprise. The main objective of this project is to find that element of surprise amongst a set of 10000 articles related to health information mainly diabetes which is obtained using a set of machine learning algorithms. The final outcome is evaluated with the help of a user survey targeting the users with basic domain knowledge.

## Problem Statement

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. 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 person based on this person's background knowledge. Personalized surprise is not necessarily surprising to society.

### Sub Tasks

- Choose one type of surprise you want to work with: general or personalized.
- Construct a working definition of the surprise for your project. That means you need to define expectation, divergence, or expected likelihood.
- Develop one computational approach to find surprising news in the corpus of diabetes news.
- Conduct some evaluations (without users or with users).

## What is a surprise?

- We define the surprise in an article based on the randomness of a topic in the article.
- Articles with high randomness have higher chance to be surprising.
- To measure the surprise in these articles, we need to calculate the entropies.

## Approach


Fig 1. Flow chart of our approach.

### Data preprocessing

Since we have a very huge dataset (10,455 articles) to process, Data Pre-Processing plays a major role in our approach.

Articles that are written in natural language mostly have a lot of redundant words, punctuations, stop words etc., which in no way helps us in finding a surprise. So as a part of data cleaning and pre-processing we removed stop words, punctuations and also lemmatized the words where inflectional endings are removed, and pure dictionary form of the base words are retained. Ex- Lemma of the word “Walking” is “Walk”.

The output of this step is a clean corpus free of punctuations and stop words and lemmatized words.

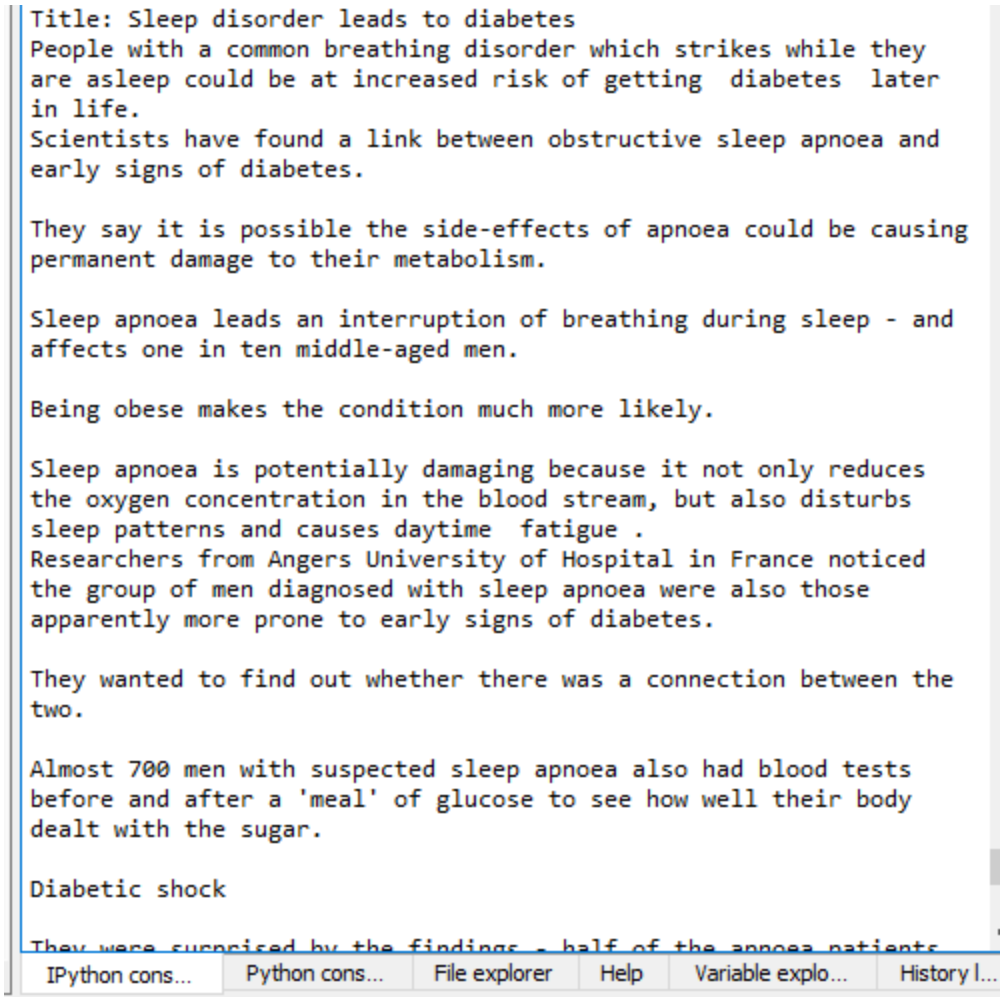

Fig 2. Snapshot of an article before Data Pre-Processing

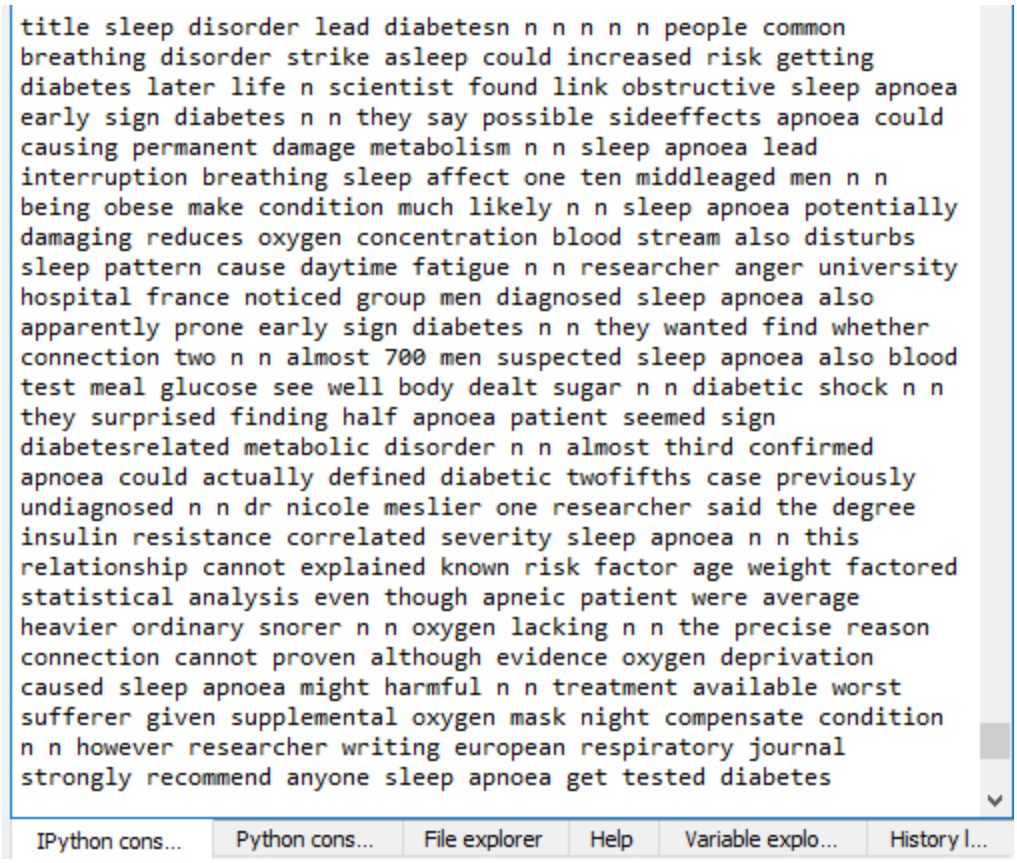

Fig 3. Snapshot of the same article after Data Pre-Processing

## Data representation

- The best way of representing documents is using a bag of topics which can be obtained using the topic modelling technique called Latent Dirichlet Allocation (LDA).
- 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 basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. In LDA the topic distribution is assumed to have a sparse Dirichlet prior.
- We performed LDA topic modelling for 30 topics on the pre-processed data.
- A Sample of five topics and five words in each topic:

| Words | Topic 0 | Topic 1 | Topic 2 | Topic 3 | Topic 4 |
|---|---|---|---|---|---|
| 1. | american | surgery | drug | glucose | medicine |
| 2. | association | loss | patient | blood | research |
| 3. | million | patient | treatment | test | institute |
| 4. | uk | obesity | metformin | technology | medical |
| 5. | people | bariatric | therapy | monitoring | center |

Table 1. Sample of 5 topics and the words in each topic.

| | Topic 0 |
|---|---|
| Article | (0, '0.107*"diabetes" + 0.080*"n" + 0.025*"people" + 0.020*"type" + 0.013*"uk" + 0.011*"2" + 0.010*"american" + 0.010*"association" + 0.009*"1" + 0.008*"million"') |
| 100032.txt | 0 |
| 100065.txt | 0 |
| 100079.txt | 0.192324326 |
| 100082.txt | 0.02197372184 |
| 100108.txt | |
| 100111.txt | 0 |
| 100121.txt | 0.03464738092 |
| 100147.txt | 0 |
| 100168.txt | 0 |
| 100203.txt | 0.03307876011 |
| 100205.txt | 0.0212297325 |
| 100219.txt | 0.03259146106 |
| 100273.txt | 0.7106316339 |
| 100276.txt | 0.2097591926 |
| 100336.txt | 0 |
| 100367.txt | 0.03064118677 |
| 100467.txt | 0.1058845255 |
| 100571.txt | 0.09303430155 |
| 10059.txt | 0.1470310787 |
| 100591.txt | 0.1570747686 |
| 100652.txt | 0.06667506399 |
| 100683.txt | 0.3101448968 |
| 100690.txt | 0.437171194 |
| 100729.txt | 0.2905191238 |

Fig 4. Snapshot of one topic after the application of LDA Topic Modelling

The above snapshot shows the contribution of the topic in each of the document.

## Randomness identification

According to our definition a document which has highly random topics has a good chance of holding surprising content.
In order to calculate the randomness, we used “Entropy”.
**Entropy** is a measure of the randomness in the information being processed.
Entropy is calculated for all the 30 topics which are distributed over 10000 documents.

$$H(t = i) = -\sum_{k=0}^{Size(CC)} p(k_i) \log_2 p(k_i)$$

| Document | (0, '0.107*"diabetes" + 0.080*"n" + 0.025*"people" + 0.020*"type" + 0.013*"uk" + 0.011*"2" + 0.010*"american" + 0.010*"association" + 0.009*"1" + 0.008*"million"') | (1, '0.037*"n" + 0.034*"diabetic" + 0.026*"kidney" + 0.021*"disease" + 0.020*"blood" + 0.019*"eye" + 0.017*"patient" + 0.016*"vessel" + 0.014*"joslin" + 0.012*"retinopathy"') | (2, '0.043*"weight" + 0.039*"surgery" + 0.023*"loss" + 0.019*"patient" + 0.019*"obesity" + 0.016*"obese" + 0.015*"2" + 0.015*"diabetes" + 0.012*"type" + 0.011*"bariatric"') | (3, '0.033*"drug" + 0.029*"patient" + 0.028*"2" + 0.026*"type" + 0.022*"treatment" + 0.021*"diabetes" + 0.018*"metformin" + 0.013*"study" + 0.011*"therapy" + 0.011*"effect"') |
|---|---|---|---|---|
| 100032.txt | 0 | 0 | 0 | 0 |
| 100065.txt | 0 | 0 | 0 | 0 |
| 100079.txt | 0.192324 | 0 | 0 | 0 |
| 100082.txt | 0 | 0 | 0 | 0 |
| 100108.txt | 0.021974 | 0 | 0.011825 | 0.092666 |
| 100111.txt | 0 | 0 | 0.01072 | 0.038117 |
| 100121.txt | 0.034647 | 0.015306 | 0 | 0.074676 |
| 100147.txt | 0 | 0 | 0 | 0 |
| 100168.txt | 0 | 0 | 0 | 0 |
| 100203.txt | 0.033079 | 0.149609 | 0 | 0 |
| 100205.txt | 0.02123 | 0 | 0 | 0.053218 |
| 100219.txt | 0.032591 | 0 | 0 | 0 |
| 100273.txt | 0.710632 | 0 | 0.014403 | 0 |
| 100276.txt | 0.209759 | 0 | 0 | 0 |

Fig 5. Snapshot of the entropies for the LDA topics

## Dataset Reduction

After calculating the topic entropies, we took the topics with the top three entropies (TTT)(Highly Random) just to make sure that we remove all the articles which have most frequently occurring/general topics.

| | A | B | C |
|---|---|---|---|
| 1 | Topic_18 | Topic_28 | Topic_2 |
| 2 | 0.9 | 0 | 0 |
| 3 | 0.7 | 0 | 0 |
| 4 | 0.3 | 0.4 | 0 |
| 5 | 0 | 0.4 | 0.3 |
| 6 | 0 | 0.5 | 0.3 |
| 7 | 0.7 | 0 | 0 |
| 8 | 0 | 0.3 | 0.4 |
| 9 | 0 | 0 | 0.7 |
| 10 | 0 | 0.7 | 0 |
| 11 | 0 | 0.7 | 0 |
| 12 | 0 | 0.4 | 0.4 |
| 13 | 0 | 0 | 0.7 |
| 14 | 0.7 | 0 | 0 |
| 15 | 0.2 | 0.2 | 0.3 |
| 16 | 0 | 0 | 0.7 |
| 17 | 0.7 | 0 | 0 |
| 18 | 0 | 0.1 | 0.6 |
| 19 | 0 | 0.3 | 0.6 |
| 20 | 0 | 0.3 | 0.4 |
| 21 | 0.8 | 0 | 0 |

Fig 6. Document probability of three topics with highest entropy

As these TTT represent the topics with highest randomness, we select the set of articles (S) that satisfy the following:
The summation of three topic probabilities is more than a threshold t (we chose t=0.7); which means that this article has a high chance to be more random.

$$\sum_{e=1}^{len(TTT)} p(e)$$

By applying this, we reduce the dataset size based on the randomness for each article.

## Outlier Detection

The output from the previous step gives us a set of articles that hold the most random content. Since there is always a scope for outliers we performed another method called K-means clustering to detect and eliminate the outliers.

To perform K-means we should know the number of clusters we are dealing with. Instead of randomly selecting a number we used a very well-known Elbow method.

- The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k (say, k from 1 to 30 in our case), and for each value of k calculate the sum of squared errors (SSE).
- Then, plot a line chart of the SSE for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k (the SSE is 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster). So, our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k.

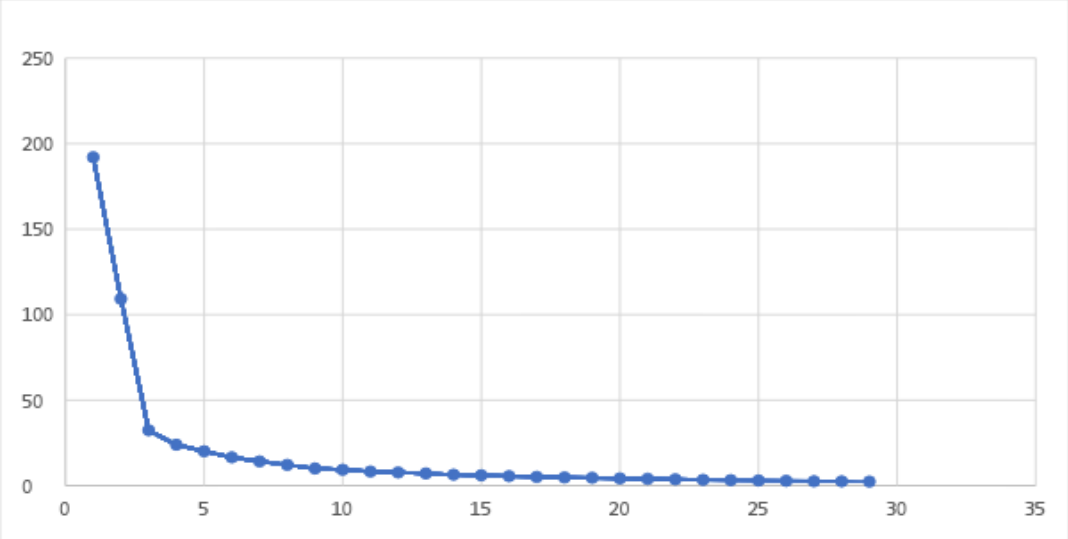


Fig 7. Line chart of the SSE for k (1-30)

From the above plot, it's clear that elbow point is 3 which gives us the optimum number of clusters for performing K-means Clustering on our data set.

Now that we have 3 clusters, we identify the outlier by measuring the distances between each pair of centroid points of those clusters.

The centroid with the highest distance is the outlier, which in our case is cluster 1.

| Clusters | Cluster 0 | Cluster 1 | Cluster 2 |
|---|---|---|---|
| **Cluster 0** | 0 | 0.893883 | 0.68764 |
| **Cluster 1** | 0.893883 | 0 | 0.909207 |
| **Cluster 2** | 0.68764 | 0.909207 | 0 |

Table 2. Distance between the clusters.

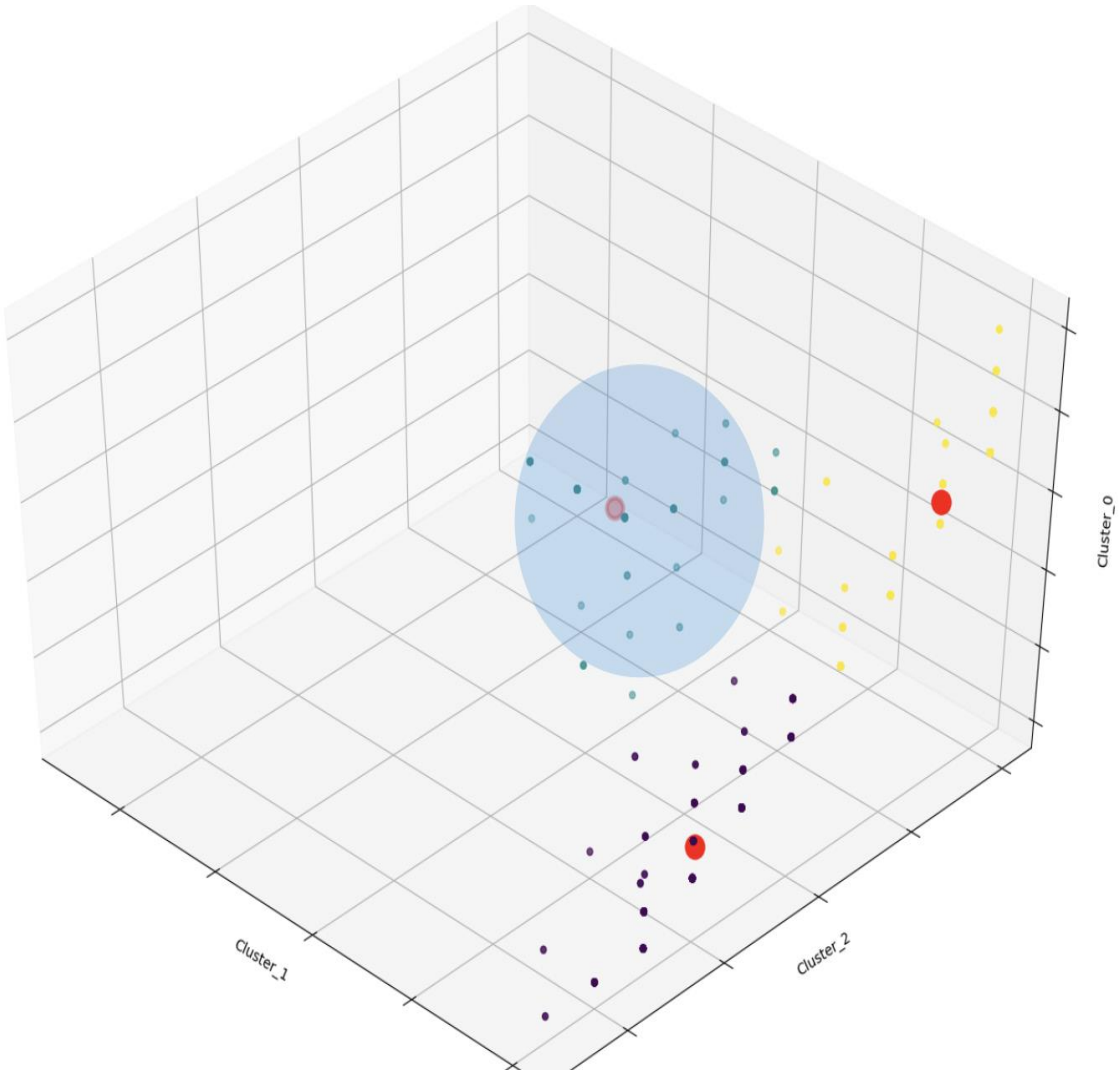


Fig 8. Scatter plot of the three clusters.

From the above figure, we can observe that the cluster circled with blue line is the outlier.

This is considered as an outlier based on the total distance between clusters (the cluster which is farther from the other two is ignored)

Usually the outlier has noisy/unwanted data and is omitted from the result. Based on the same assumption we proposed that the other two clusters have surprising articles.

## Evaluation of the outcome

From the scatter plot in Fig 7, we got to know about the outliers. Cluster 1 is an outlier and thus is omitted from the result. To evaluate our result, we have conducted a user survey that targets the people with very basic knowledge in this domain.

This survey has 10 articles that were picked randomly from the above three clusters. Each question has the article title and a link to the article.

User must go through the article and select one of the answer options (Surprising/ Not Surprising) mandatorily to finish the survey.

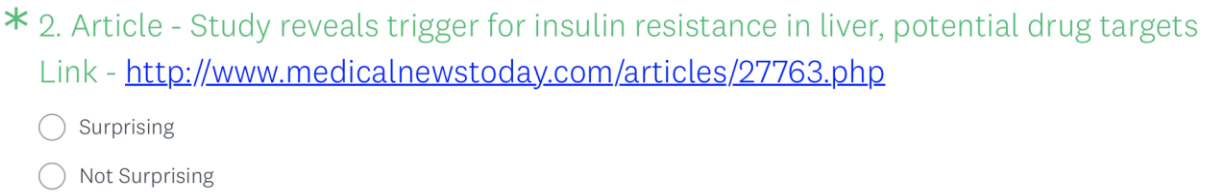


Fig 9. Snapshot of a question asked in user survey.

We managed to get around 85 responses from various users and after going through the analytics provided by survey monkey we concluded that the assumption made in the previous step is correct.

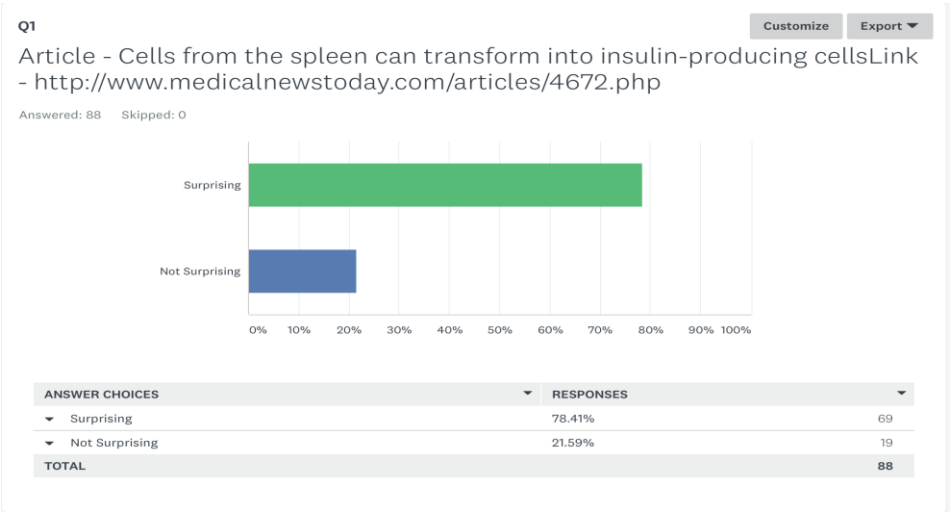


Fig 10. Snapshot of the survey analysis for one question.

| Clusters | User's evaluation | Our approach |
|---|---|---|
| Cluster 0 | Surprising (81%) | Surprising |
| Cluster 1 | Surprising (17%) | Non-Surprising |
| Cluster 2 | Surprising (76%) | Surprising |

Table 3. User Survey Analysis

## Conclusions

- The more the probability of random topics in an article, the more surprising the article will be.
- In simple terms, we can say that random topic combinations are more likely to contain surprising content.
- The user study shows that the articles belonging to cluster 0 and cluster 2 are more surprising compared to articles in cluster 1.

## Challenges

- The major challenge with this project was to define **Surprise**. We explored different approaches to define this surprise.
- Selecting the appropriate approach among them based on the few test runs and the one which would fit better for the given problem**.**
- We had problems with implementing LDA topic modeling in gensim.
- When we went through the documents returned after randomness identification, we found that they are not that surprising and contained medical terms. So, we wanted to further refine our result.
- Once we got the set of surprising documents, we were not sure how to evaluate them. Then, we conducted a user survey to evaluate the result.

## References


- https://pypi.python.org/pypi/gensim
- https://radimrehurek.com/gensim/models/ldamodel.html
- https://www.datascience.com/blog/k-means-clustering
- Xi Niu, Mary Lou Meher and Fakhri Abbas 2017. "*Surprise Me If You Can: Serendipity in Health Information"*.

# Appendix A

## Python Codes

```
from os import listdir
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import string
import gensim
from gensim import corpora
from collections import defaultdict
import csv


def clean(doc):
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized


if __name__ == '__main__':
    doc_complete = []
    input_path = "C:/Users/doula/Desktop/KDD Final Project Workspace/diabetes/"
    output_path = "C:/Users/doula/Desktop/KDD Final Project Workspace/"
    stop = set(stopwords.words('english'))
    exclude = set(string.punctuation)
    lemma = WordNetLemmatizer()
    doc_names_list = []
    for current_file in listdir(input_path):
        doc_text = ''
        input_file = open(input_path + current_file, 'r', encoding="utf8")
        current_doc = input_file.readlines()
        doc_text = str(current_doc)
        doc_complete.append(doc_text)
        input_file.close()

    doc_clean = [clean(doc).split() for doc in doc_complete]
    dictionary = corpora.Dictionary(doc_clean)
    doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
    Lda = gensim.models.ldamodel.LdaModel
    lda_model = Lda(doc_term_matrix, num_topics=30, id2word=dictionary, passes=50)
    topic_list = lda_model.print_topics(num_topics=30, num_words=10)
    doc_topic_distribution = [lda_model[dictionary.doc2bow(doc)] for doc in doc_clean]

    doc_topic_distribution = defaultdict(dict)
    for current_file in listdir(input_path):
        for index in range(30):
            doc_topic_distribution[current_file][index] = 0

    for current_file in listdir(input_path):
        doc_text = ''
        input_file = open(input_path + current_file, 'r', encoding="utf8")
        current_doc = input_file.readlines()
        doc_bow = dictionary.doc2bow(clean(str(current_doc)).split())
        doc_topic_distr = lda_model[doc_bow]
        for d in doc_topic_distr:
            doc_topic_distribution[current_file][d[0]] = round(d[1], 1)

    topic_list.insert(0, '')
    with open(output_path + 'topics_distribution_rounded.csv', 'w', newline='') as
output_file:
        writer = csv.writer(output_file, delimiter=',')
        writer.writerow(topic_list)
        for k, v in doc_topic_distribution.items():
            writer.writerow([k, v[0], v[1], v[2], v[3], v[4], v[5], v[6], v[7], v[8],
```

Program 1: LDA Topic Modeling

```
import csv
import math
from collections import defaultdict
from collections import Counter
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from operator import itemgetter


def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)

def ClusterIndicesNumpy(clustNum, labels_array): #numpy
    return np.where(labels_array == clustNum)[0]


if __name__ == '__main__':
    input_p = 'C:/Users/doula/Desktop/KDD Final Project
Workspace/topics_distribution_rounded.csv'
    columns = defaultdict(list)
    with open(input_p) as f:
        reader = csv.DictReader(f)
        for row in reader:
            for (k, v) in row.items():
                columns[k].append(v)
    if '' in columns: del columns['']

    dict_with_floats = defaultdict(list)
    dict_with_floats = {k: list(map(float, columns[k])) for k in columns}
    dict_no_zeros = defaultdict(list)
    for k, v in dict_with_floats.items():
        temp_v = [item for item in v if item != 0.0]
        dict_no_zeros[k] = temp_v

    output_p = 'C:/Users/doula/Desktop/KDD Final Project Workspace/out.txt'
    f = open(output_p, 'w')

    topic_entro = defaultdict(float)
    for k, v in dict_no_zeros.items():
        setted_list = set(v)
        for f in setted_list:
            p = v.count(f) / len(v)
            topic_entro[k] = topic_entro[k] - p * math.log2(p)


    top_3 = Counter(topic_entro)
    for m in top_3:
        print(m)
    top_3_topic_entropies = top_3.most_common(3)
    doc_top_3_topics = defaultdict(list)
    with open(input_p) as input_file:
        reader = csv.DictReader(input_file)
```

Program 2: Topic Entropy Calculation

```
import csv
import math
from collections import defaultdict
from collections import Counter
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from operator import itemgetter

output_p = 'C:/Users/doula/Desktop/KDD Final Project
Workspace/out.txt'
f = open(output_p, 'w')

topic_entro = defaultdict(float)
for k, v in dict_no_zeros.items():
    setted_list = set(v)
    for f in setted_list:
        p = v.count(f) / len(v)
        topic_entro[k] = topic_entro[k] - p * math.log2(p)

top_3 = Counter(topic_entro)
for m in top_3:
    print(m)
top_3_topic_entropies = top_3.most_common(3)

doc_top_3_topics = defaultdict(list)
with open(input_p) as input_file:
    reader = csv.DictReader(input_file)
    for row in reader:
        # print(row[''])
        for item in top_3_topic_entropies:
            doc_top_3_topics[row['']].append(row[item[0]])
doc_vec = {k: list(map(float, doc_top_3_topics[k])) for k in
doc_top_3_topics}
print(len(doc_vec))


output_p = 'C:/Users/doula/Desktop/KDD Final Project
Workspace/k_mean.csv'
list_of_topics = []
list_of_docs = []
aggregate_topic_docs = []
with open(output_p, 'w', newline='') as out:
    writer = csv.writer(out)
    writer.writerow(('Topic_18', 'Topic_28', 'Topic_2'))
    for k, v in doc_vec.items():
        if sum(v) >= 0.7:
```

Program 3: K-means Clustering