Topic Modelling and Co-occurrences analysis of Twitter Data

Hari Hara Priya Kannan

Contents

1. Abstract

In this project, we will use the IPython notebook to mine data from Twitter with the Tweepy library. Once we have fetched the raw stream for a specific query, we will at first do some basic word frequency analysis on the results using Python's built in dictionaries, and then we will use the excellent NetworkX library developed at Los Alamos National Laboratory to look at the results as a network and understand some of its properties.

Using NetworkX, we aim to answer the following questions: for a given user, which words tend to appear together in tweets, and global pattern of relationships between these words emerges from the entire set of results?

Obviously, the analysis of text corpora of this kind is a complex topic at the intersection of natural language processing, graph theory and statistics, and here we do not pretend to provide an exhaustive coverage of it. Rather, we want to show you how with a small amount of easy to write code, it is possible to do a few non-trivial things based on real-time data from the Twitter stream.

2. Introduction

With a 140-character limit, it’s easy to think of Twitter as a basic platform. But the fact of the matter is numerous marketers see success in their social media marketing strategies by paying closer attention to Twitter data.

Any discussion in social media can be fruitful if the people involved in the discussion are related to a field. In a similar way to advertise an event, it is useful to find users who are interested in the content of the event. In social networks like Twitter, which contain a large number of users, the categorization of users based on their interests will help this cause.

In this project, we will look at the timeline of the popular ‘KDnuggets . KDnuggets is a leading site on **Business Analytics, Big Data, Data Mining, Data Science, and Machine Learning**

https://www.kdnuggets.com/about/index.html

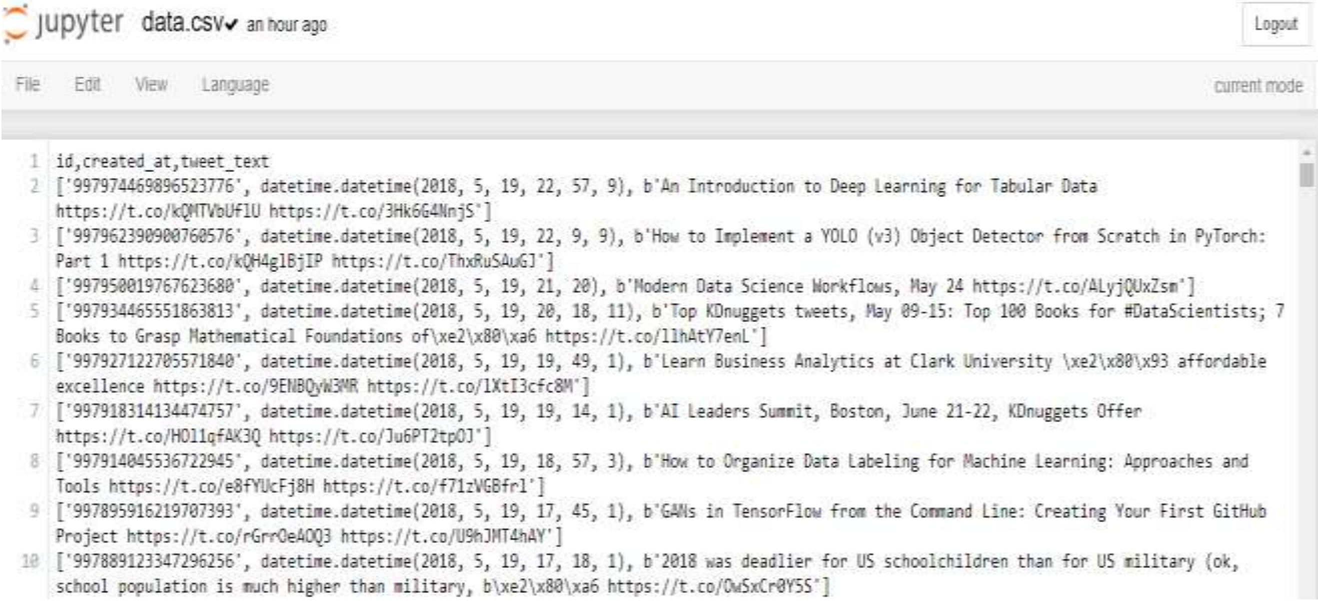
Our aim is to find out what are the most trending topics being discussed and the relationship between them.

3. Methodology

3.1. Data Scrapping

The data was collected from the timeline of the account ‘Kdnuggets’ using the tweepy API. A total of 3200 tweets were collected in an iterative format. Since we are only using the text from the tweets, the id, created_at and the tweet_text were only attributes that were stored.

The image below shows the sample of the data collected in a csv format:



```
id,created_at,tweet_text
['997974469896523776', datetime.datetime(2018, 5, 19, 22, 57, 9), b'An Introduction to Deep Learning for Tabular Data
https://t.co/kQMTVbUFlU https://t.co/3Hk6G4NnjS']
['997962390900760576', datetime.datetime(2018, 5, 19, 22, 9, 9), b'How to Implement a YOLO (v3) Object Detector from Scratch in PyTorch:
Part 1 https://t.co/kQH4glBjIP https://t.co/ThxRuSAuGJ']
['997950019767623680', datetime.datetime(2018, 5, 19, 21, 20), b'Modern Data Science Workflows, May 24 https://t.co/ALyjQUxZsm']
['997934465551863813', datetime.datetime(2018, 5, 19, 20, 18, 11), b'Top KDnuggets tweets, May 09-15: Top 100 Books for #DataScientists; 7
Books to Grasp Mathematical Foundations of\xe2\x80\xa6 https://t.co/llhAtY7enL']
['997927122705571840', datetime.datetime(2018, 5, 19, 19, 49, 1), b'Learn Business Analytics at Clark University \xe2\x80\x93 affordable
excellence https://t.co/9ENBQyW3MR https://t.co/lXtI3cfc8M']
['997918314134474757', datetime.datetime(2018, 5, 19, 19, 14, 1), b'AI Leaders Summit, Boston, June 21-22, KDnuggets Offer
https://t.co/HOl1qfAK3Q https://t.co/Ju6PT2tpOJ']
['997914045536722945', datetime.datetime(2018, 5, 19, 18, 57, 3), b'How to Organize Data Labeling for Machine Learning: Approaches and
Tools https://t.co/e8fYUcFj8H https://t.co/f71zVGBfrl']
['997895916219707393', datetime.datetime(2018, 5, 19, 17, 45, 1), b'GANs in TensorFlow from the Command Line: Creating Your First GitHub
Project https://t.co/rGrrOeAOQ3 https://t.co/U9hJMT4hAY']
['997889123347296256', datetime.datetime(2018, 5, 19, 17, 18, 1), b'2018 was deadlier for US schoolchildren than for US military (ok,
school population is much higher than military, b\xe2\x80\xa6 https://t.co/OwSxCr0Y5S']
```

3.2 Data Preprocessing

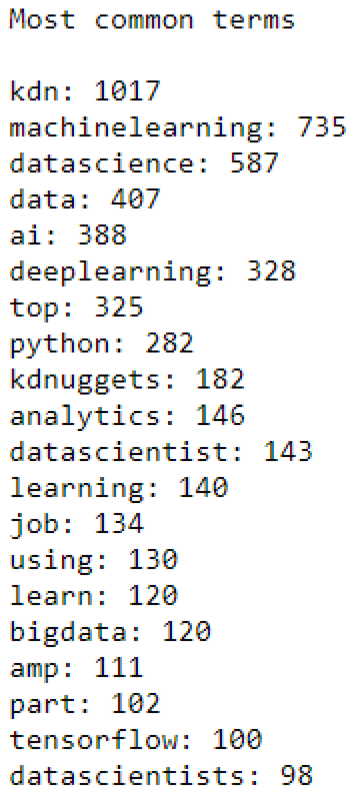
As a first step in the data preprocessing stage, we store all the tweet_text values to a dataframe for further cleaning. From there on the following steps are carried out:

1. Stopwords are removed using the nltk corpus
2. Punctuation marks are removed
3. Numbers were removed
4. Hashtags and hyperlinks are removed
5. Certain more words that were commonly occurring in the hyperlink texts were removed

6. The text was converted to lower case
7. The text was split into words and saved in list called tokens for further processing.

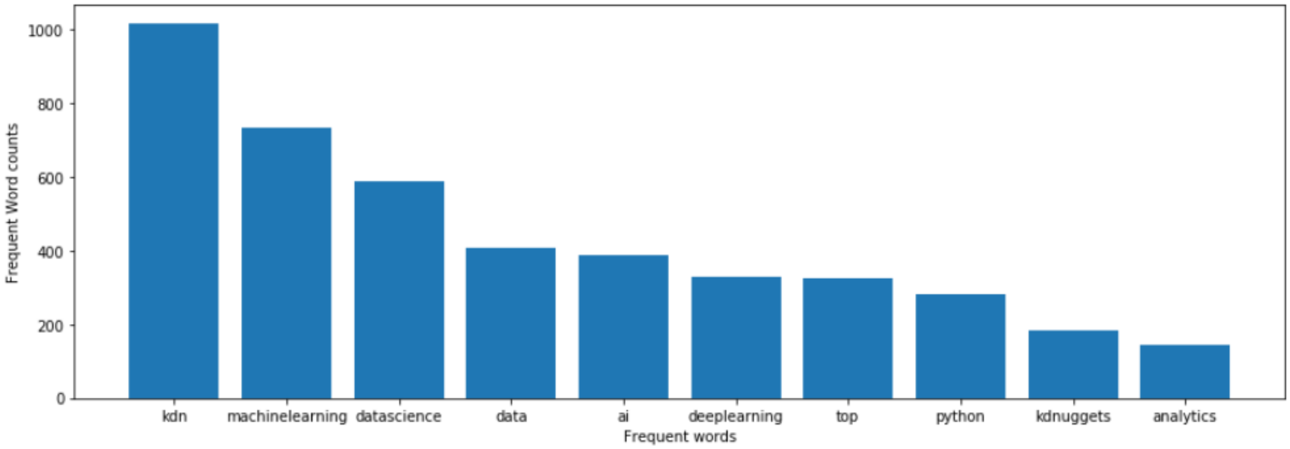
3.3 Finding frequent terms

From the tokens obtained in the previous step, the top 20 frequent terms were determined along with their count. The results are shown in the image below.



```
Most common terms

kdn: 1017
machinelearning: 735
datascience: 587
data: 407
ai: 388
deeplearning: 328
top: 325
python: 282
kdnuggets: 182
analytics: 146
datascientist: 143
learning: 140
job: 134
using: 130
learn: 120
bigdata: 120
amp: 111
part: 102
tensorflow: 100
datascientists: 98
```

The top ten words were visualized with a histogram as shown below:



3.4 Topic Modelling

Topic models provide a simple way to analyze large volumes of unlabeled text. A "topic" consists of a cluster of words that frequently occur together. Using contextual clues, topic models can connect words with similar meanings and distinguish between uses of words with multiple meanings.

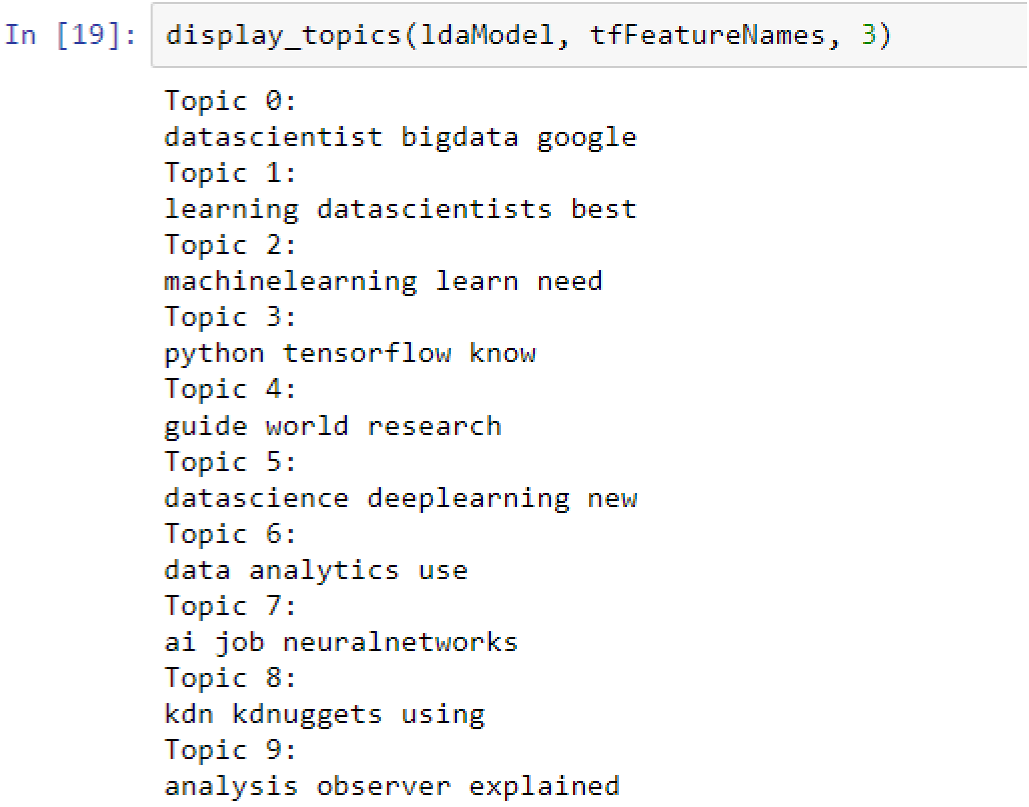
Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (“curse of dimensionality”) and also reduce computational costs.

Python’s Scikit Learn provides a convenient interface for topic modeling using algorithms like Latent Dirichlet allocation(LDA), LSI and Non-Negative Matrix Factorization.

To build the model from the tokens we have obtained, we perform the following steps:

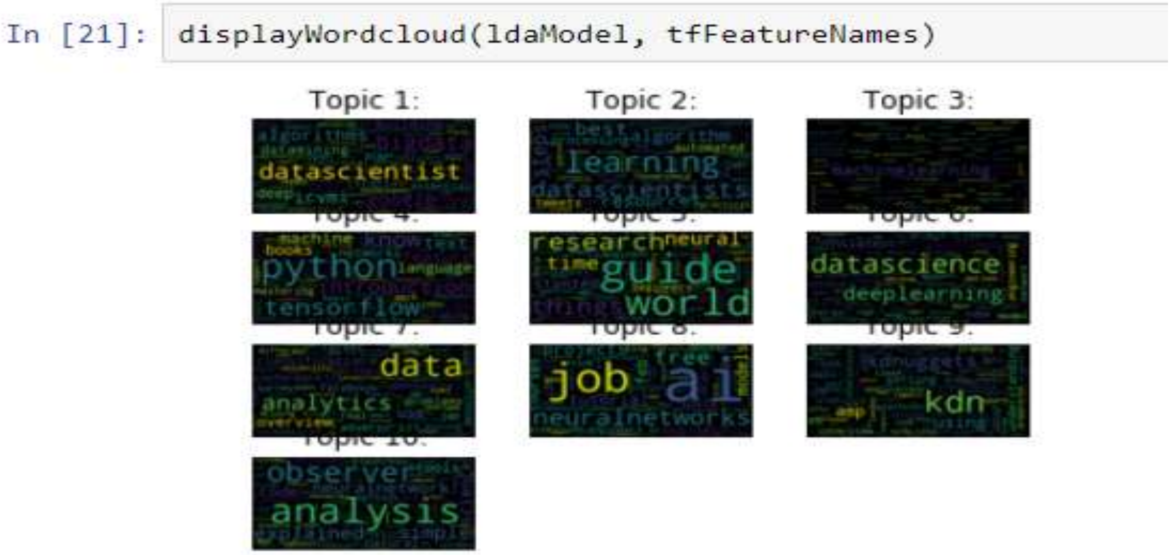
1. Create document word matrix
2. Build LDA model with sklearn

3. Review topics distributions. The following image shows the topic distributions obtained.



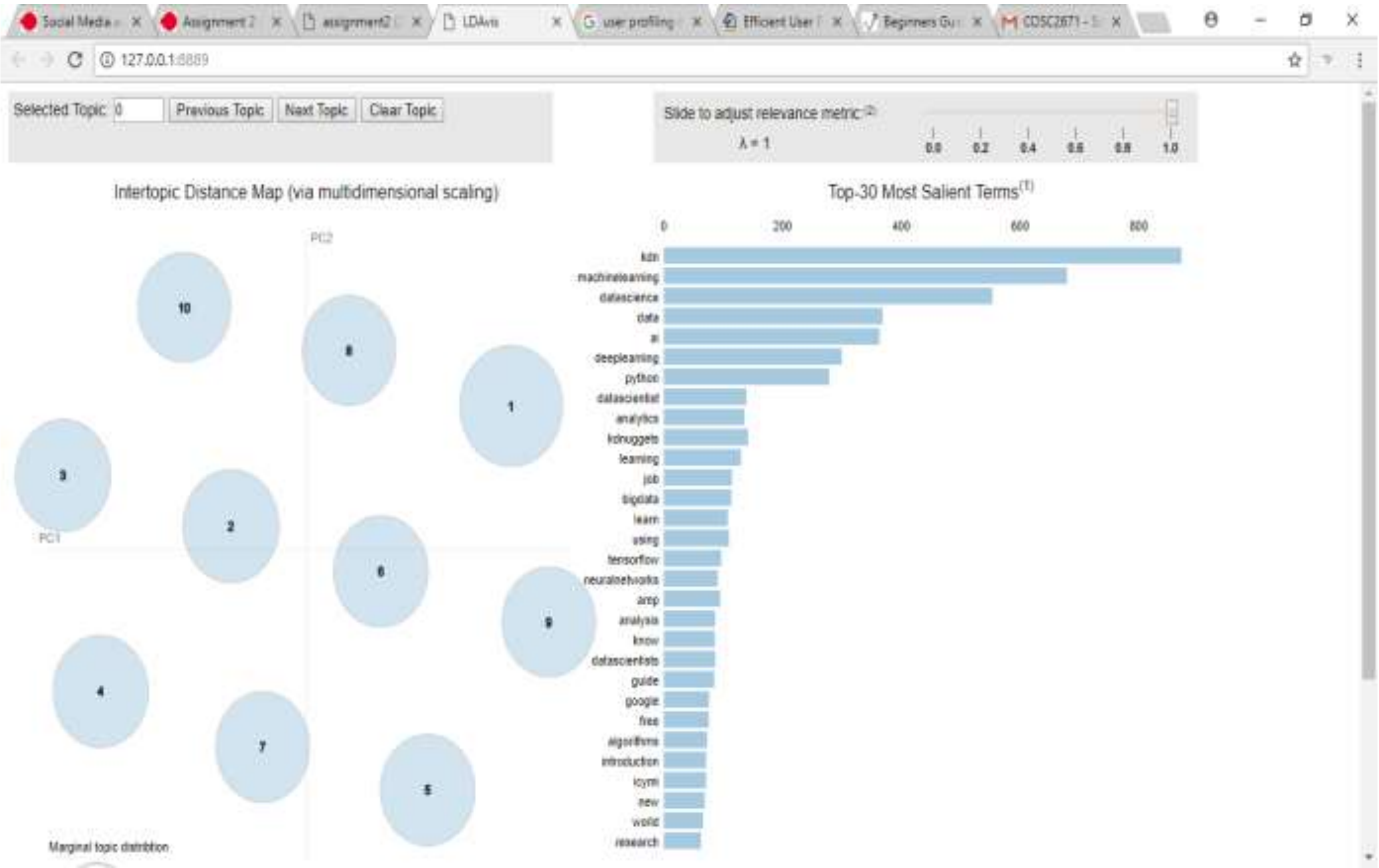
```
In [19]: display_topics(ldaModel, tfFeatureNames, 3)

         Topic 0:
         datascientist bigdata google
         Topic 1:
         learning datascientists best
         Topic 2:
         machinelearning learn need
         Topic 3:
         python tensorflow know
         Topic 4:
         guide world research
         Topic 5:
         datascience deeplearning new
         Topic 6:
         data analytics use
         Topic 7:
         ai job neuralnetworks
         Topic 8:
         kdn kdnuggets using
         Topic 9:
         analysis observer explained
```

4. A word cloud is created to visualize the topics as follows:



```
In [21]: displayWordcloud(ldaModel, tfFeatureNames)
```

5. Visualize the model with pyLDAvis. The following image shows the interactive visualization for the LDA model.



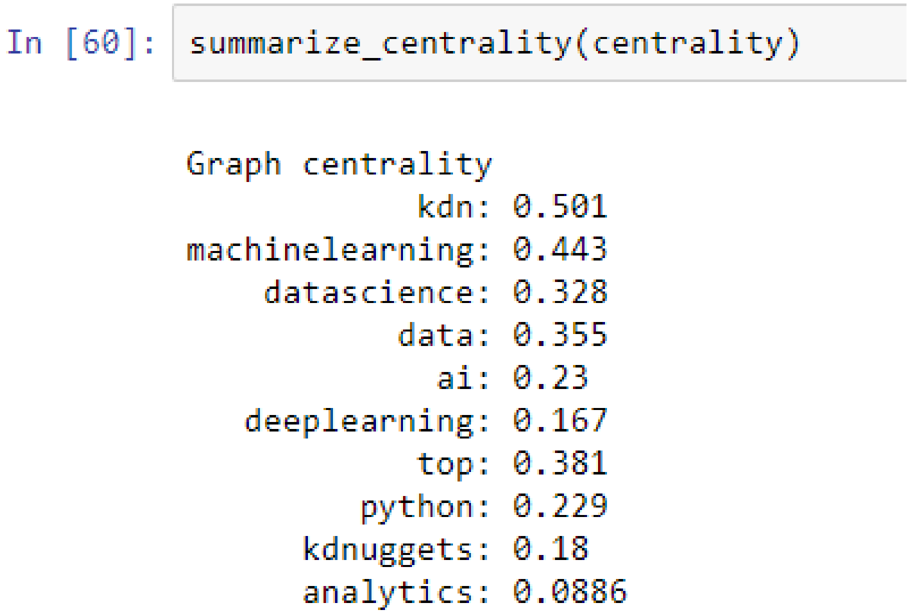
3.5 Co-occurrence graph

An interesting question to ask is: which pairs of words co-occur in the same tweets? We can find these relations and use them to construct a graph, which we can then analyze with NetworkX and plot with Matplotlib.

We limit the graph to have at most 10 (for the most frequent words) just to keep the visualization easier to read.

An interesting summary of the graph structure can be obtained by ranking nodes based on a centrality measure. NetworkX offers several centrality measures, in this case we look at the Eigenvector Centrality.

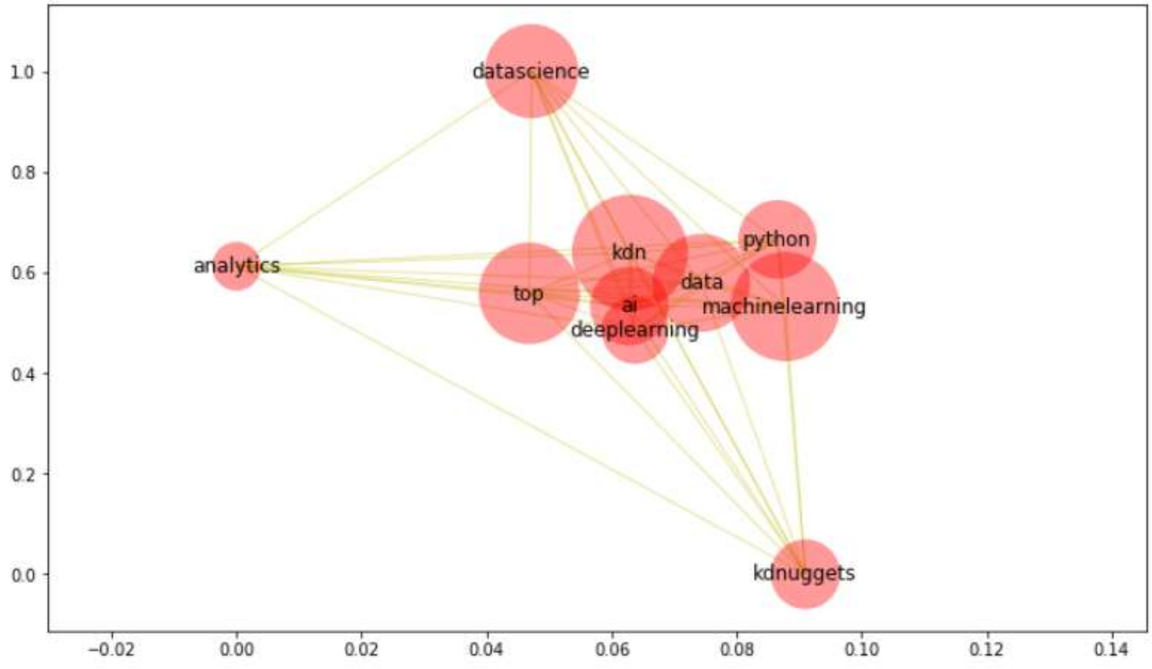
In graph theory, eigenvector centrality (also called eigen centrality) is a measure of the influence of a node in a network. Relative scores are assigned to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. A high eigenvector score means that a node is connected to many nodes who themselves have high scores.



```
In [60]: summarize_centrality(centrality)
```

```
Graph centrality
             kdn: 0.501
 machinelearning: 0.443
     datascience: 0.328
            data: 0.355
              ai: 0.23
    deeplearning: 0.167
             top: 0.381
          python: 0.229
       kdnuggets: 0.18
       analytics: 0.0886
```

The above table shows the centrality scores of each of the nodes.Now we will visualize the centrality results in a graph format.



4.Results

From the above analysis we can conclude the following:

1. Top frequent words were kdn, machinelearning, datascience, data, ai, machine learning. This gives us an idea of what is being discussed dominantly in the tweets.
2. The major topics includes data scientist bigdata google, learning data scientists better, machinelearning learn need and python TensorFlow know.
3. The co-occurrence graph shows us that top, ai, deeplearning, machinelearning, python is strongly connected.

In the present, hiring on social media is becoming increasingly common. Through this analysis, we can identify individuals who are pioneers in their field.

5.Conclusion

In this analysis, we have tried to get an overview about the user's profile for any given user and found out the frequent topics and their co-occurrences. The analysis has helped us get a basic understanding of topic modelling and creating graphs. With the above results, we can safely conclude that in future work, we can extend to find communities that are interested in the above topics using the follower's data. This can be useful in many ways such as targeting people of similar interests for a new campaign in the topic.

6.References

- The idea was inspired from https://gist.github.com/ellisonbg/3837783.
- Readings about LDA https://www.machinelearningplus.com/nlp/topic-modeling-python-sklearn-examples/
- Readings about Centrality https://networkx.github.io/documentation/stable/reference/algorithms/centrality.html