Text analysis assignment report

Gloria Ainebyona
Makerere University
School of Computing and Informatics Technology
gloria.ainebyona@students.mak.ac.ug

Abstract

Text clustering is the task of grouping a set of unlabelled texts in such a way that texts in the same cluster are more similar to each other than those in the other clusters.
In this report, I present a methodology that used to come up with clusters from the comments that were given by supervisors to interns.I trained an Named Entity Recognition to recognise the different entities in the comments.

1 Text clustering using kmeans algorithm

1.1 Methodology

1.1.1 Data exploration

The dataset used was of size 4968 rows and 2 columns from https://www.fams-cit.com/fscomments. This was majorly about comments given by Supervisors in the different organisations to their respective interns for a given period of time. The features found in the dataset were comment-id and Comment columns as shown in figure 1 below. My main focus was on the comment column as it was the most important feature for this assignment on text clustering.

```
dataset.shape
```

```
(4969, 2)
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4969 entries, 0 to 4968
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   comment_id  4969 non-null   int64 
 1   Comment     4968 non-null   object
dtypes: int64(1), object(1)
memory usage: 77.8+ KB
```

Figure 1: Data exploration

1.1.2 Data Pre-processing

Data pre-processing was done to normalise the text for suitability to create a corpus for use by the algorithm. This was performed through removal of irrelevant symbols, stop words that did not carry significant meaning for the clustering purpose, conversion of the text to lower case for all words in order to avoid having some words being misclassified due to their different meaning irrespective of the same spelling.
I also carried out conversion of number definition in order to convert any numbers in the text into their respective word text. The irrelevant white spaces between text statements were also removed. A method like stemming was not used because this tends to make some words lose meaning while it truncates some characters from these words.
I used a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer on the corpus in order to assign a vector value to each word in comments. TF-IDF is a numerical statistic that demonstrates how important a word is. Term Frequency is just ratio number of current word to the number of all words in document/string/etc. TF-IDF vectorizer also helps in weighing the word counts by a measure of how they appear in the text document.

1.1.3 Algorithm

I used kmeans algorithm for clustering which is unsupervised machine learning algorithm. I used this because it handles unlabelled data and helps to summarize information from large text data by creating the different clusters or groups depending on the distance measure from the cluster's centroid.

1.1.4 Model training

I first trained the model on 10 clusters and used the elbow method to return the optimum value of k. The optimum k was determined at the value 5 as shown in figure 2 below. From this I extracted the 5 clusters. K stands for the number of clusters.

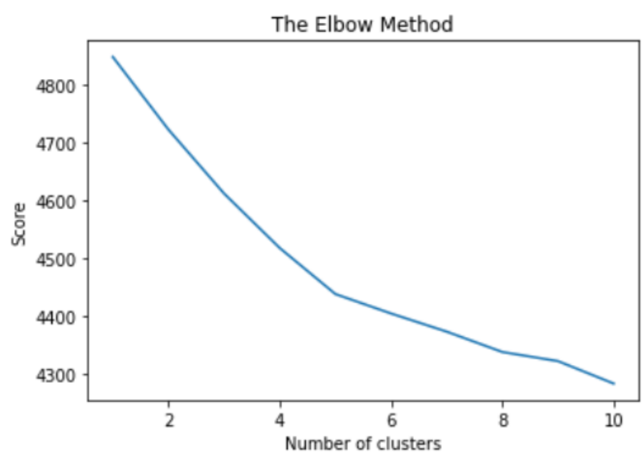

Figure 2: Elbow method to determine the optimum k value

1.2 Result

It is noticeable that the algorithm used similarity measure during clustering by categorizing sentences with similar words together in the one cluster. An example is the progressive comments that were categorised under cluster 4 and comments that contained the word good were categorised under cluster 3 as shown in figures 3 and 4 below.

```
clusters.head()
```

	comment_id	Comment	cluster
0	5	djfjkdfjkjkffdk edited	0
1	41	faith exhibited enthusiasm taking project hand...	0
2	49	understood structure grails different componen...	0
3	50	intern oriented ict setup infrastructure sorot...	0
4	52	student oriented organization structure develo...	0
5	53	activities well completed	2
14	68	activity took time completed completed satisfa...	2
19	96	noted tasks completed	2
28	144	good attitude resilience good start	3
30	148	good progress expect student work together period	3
101	385	completed satisfaction	2
105	389	tasks well done good work	3
111	399	completed satisfaction	2

Figure 3: Clusters from comments

105	389	tasks well done good work	3
111	399	completed satisfaction	2
115	403	good work student	3
161	569	good work	3
326	1239	progressive	4
329	1244	progressive	4
330	1245	progressive	4
334	1249	progressive	4
335	1251	progressive	4
1478	2679	moses successfully coded presented end interns...	1
1528	2743	despite challenges power network rebecca manag...	1
1610	2859	stom successfully coded presented end internsh...	1
1612	2862	sarah done internship supposed write end inter...	1
1618	2868	despite distance managed finish internship hes...	1

Figure 4: Clusters from comments

1.2.1 Formed clusters

To find out how the algorithm clustered the different comments, for each cluster, I printed out 10 key words that it considered as shown in Figure 5 below.

```
Cluster 2:
 completed
 weeks
 tasks
 successfully
 challenges
 equip
 handson
 competence
 going
 main
------------
Cluster 3:
 good
 progress
 work
 ui
 student
 job
 start
 administration
 tasks
 systems
------------
```

```
cluster centroids:

Cluster 0:
 good
 work
 encourage
 time
 week
 internship
 tasks
 student
 team
 able
------------
Cluster 1:
 internship
 explaining
 supposed
 acquired
 report
 end
 skills
 new
 finished
```

```
Cluster 4:
 progressive
 tasks
 hand
 week
 technically
 recordings
 installations
 networking
 systems
 development
```

Figure 5: Key words per cluster

1.2.2 Testing the algorithm on unknown data

After training the model, I used text that was not part of the training in order to make some predictions. And the results were returned as shown in figure 6.

```
Y = vectorizer.transform(["satisfactorily completed"])
prediction = model.predict(Y)
print(prediction)

[2]
```

```
Y = vectorizer.transform(["good person."])
prediction = model.predict(Y)
print(prediction)

[3]
```

Figure 6: Figure 5: Key words per cluster

From the keywords presented per cluster, I grouped the clusters into categories of Excellent, Good, Neutral, Poor and Very Poor. This categorization was based on the word similarity in the statements clustered together as shown in figure 7 below.

	comment_id	Comment	cluster	clustered_category
4919	8068	aweebwa successfully implemented file upload d...	0	Excellent
4920	8069	samuel started internship going equip handson ...	0	Excellent
4921	8070	ahmed managed complete internship supposed pre...	1	Neutral
4922	8071	habibah started internship going equip handson...	2	Good
4923	8072	successfully completed weeks tasks	2	Good
4924	8073	habibah almost done internship encourage use r...	0	Excellent
4925	8075	managed complete tasks introduced opensource m...	0	Excellent
4926	8076	anjellinah fully closed tasks assigned interns...	0	Excellent
4927	8077	student gradually improving abiding instructio...	0	Excellent
4928	8078	student gradually improving abiding instructio...	0	Excellent
4929	8079	student gradually improving abiding instructio...	0	Excellent
4930	8080	student gradually improving abiding instructio...	0	Excellent
4931	8081	student gradually improving abiding instructio...	0	Excellent
4932	8082	student gradually improving abiding instructio...	0	Excellent

Figure 7: Categories of clusters

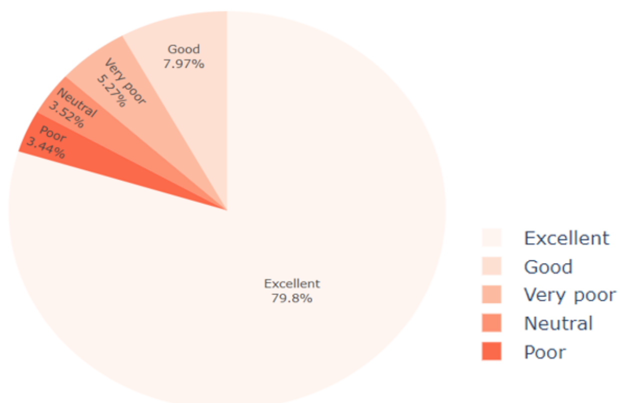

Figure 8: Percentage per cluster

1.3 Model evaluation

I used a silhouette score to evaluate the performance of the algorithm. With the increasing number of the k value in training, led to; The increase in silhouette score. For example; with k=10, silhouette score was at 0.062. with K=20, silhouette score was at 0.8, with k=200, silhouette score was at 0.115.

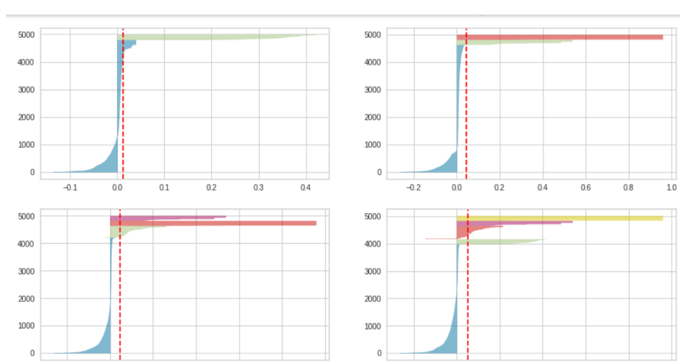

Figure 9: Model performance from the silhouette graphs

2 Named Entity Recognition (NER)

2.1 Methodology

2.1.1 Data annotation

I performed data annotation of text data using the NER Annotator by creating tags and labelling all potential entities in the data.

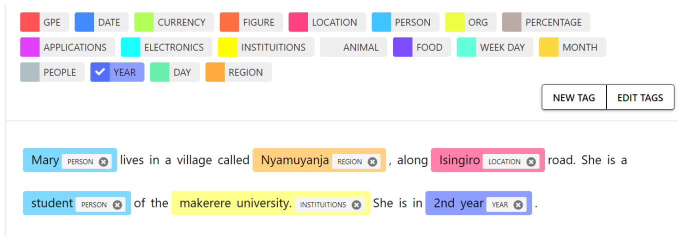

Figure 10: Tags and data labelling

I later exported the annotated data as a json file as shown in figure 11 below, which I used as training data.


```
[214,216,"PERSON"],[261,271,"APPLICATIONS"]]}],["With more practice, he will be able to operate with minimal supervision.\\r",
{"entities":[[18,22,"PERSON"]]}],["She needs more practice otherwise very interested\\r",{"entities":[[0,3,"PERSON"]]}],["She
knows about her hardware and can identify and troubleshoot related problems\\r",{"entities":[[0,3,"PERSON"],
[16,19,"PERSON"]]}],["The student is well organized, willing to learn and innovative\\r",{"entities":[[4,11,"PERSON"]]}],["She
has already learned the basics of data analysis in pandas, which is key to the main task which is developing data
visualizations. \\r",{"entities":[[0,3,"PERSON"]]}],["She has also presented a conceptual model for the API design for the data
that will be required to implement the dashboards.\\r",{"entities":[[0,3,"PERSON"]]}],["This part was challenging to him
especially with the configurations and terminations otherwise he makes effort to see that things work at the end of it all\\r",
{"entities":[[29,32,"PERSON"],[95,97,"PERSON"]]}],["I gave him a task to work (improve) on the front end of a system that is
already in production at this company, and he has been progressing well, I realized he knows some programming concepts, so I
need to introduce him to real-world problems so he can find solutions to them. He is generally doing well, he will ask where he
does not understand.\\r",{"entities":[[0,1,"PERSON"],[7,10,"PERSON"],[116,118,"PERSON"],[146,147,"PERSON"],[157,159,"PERSON"],
[196,197,"PERSON"],[216,219,"PERSON"],[246,248,"PERSON"],[271,276,"PEOPLE"],[277,279,"PERSON"],[303,307,"PERSON"]]}],["He has
some knowledge about IP addresses and now is able to put them in practice\\r",{"entities":[[0,2,"PERSON"]]}],["This week had
good development momentum. No issues from me.\\r",{"entities":[[56,58,"PERSON"]]}],["He knows about IP addresses which is very
good and now I believe he knows how to enter them and interconnect and configure different devices\\r",{"entities":
[[0,2,"PERSON"],[37,46,"COMMENT"],[55,56,"PERSON"],[65,67,"PERSON"]]}],["This week had slower momentum due to some
miscalculations on our part.\\r",{"entities":[[0,9,"TIME"]]}],["This week\'s work was mostly about reducing tech debt and
improving pull request quality. There is a lot of progress being made on this front by all team members.\\r",{"entities":
[[0,11,"TIME"],[148,160,"PEOPLE"]]}],["No notable issues came up to my attention in regard to tasks execution. I am also happy
that Benjamin gets to help out on pull request reviews.\\r",{"entities":[[72,76,"PERSON"],[93,101,"PERSON"]]}],["This weeks
progress was derailed due to Health issues by Benjamin. We hope next week will be better.\\r",{"entities":[[0,10,"TIME"],
[57,66,"PERSON"]]}],["Tasks execution went well. No notable issues from me per their execution.\\r",{"entities":
[[50,52,"PERSON"]]}],["Good attempt and great enthusiasm\\r",{"entities":[[0,12,"COMMENT"]]}],["This was new to him but now is
```


Figure 11: Extracted Json file

I converted the json file into the spacy format as shown in figure 12 below by using the docBin function, which helps in binary serialization of the data. The conversion of the json data helps to normalise all the data into a binary format that is easy for processing and training.

```
[36] import json
     Data = open('annotations-data.json')
     TRAINING_DATA = json.load(Data)

[37] for text, annot in tqdm(TRAINING_DATA['annotations']):
         doc = nlp.make_doc(text)
         ents = []
         for start, end, label in annot["entities"]:
             span = doc.char_span(start, end, label=label, alignment_mode="contract")
             if span is None:
                 print("Skipping entity")
             else:
                 ents.append(span)
         doc.ents = ents
         db.add(doc)

     db.to_disk("./annotations-data.spacy") # save the docbin object

     100%|██████████| 171/171 [00:00<00:00, 2559.10it/s]
```

Figure 12: Conversion from Json file to spacy

spaCy 3.4.1 was then used for data training. This has inbuilt configurations to automatically do the training on given data as shown in figure 13 below. The average accuracy was around 94

```
============================ Training pipeline ============================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00     58.95    5.87    4.38    8.87    0.06
  6     200       1172.18   3956.38   81.65   81.85   81.46    0.82
 14     400        375.65   1669.36   89.70   89.70   89.70    0.90
 25     600        244.73   1202.16   92.98   92.61   93.34    0.93
 37     800        301.29   1237.82   93.08   92.36   93.82    0.93
 53    1000        314.82   1292.21   93.35   93.21   93.50    0.93
 72    1200        403.72   1540.44   93.05   92.76   93.34    0.93
 95    1400        468.36   1861.18   92.41   95.29   89.70    0.92
124    1600        333.49   2013.47   93.21   92.91   93.50    0.93
159    1800        365.75   2523.58   93.59   93.51   93.66    0.94
201    2000        340.49   2833.43   93.44   94.35   92.55    0.93
251    2200        337.91   3382.93   93.80   94.10   93.50    0.94
316    2400        353.17   4140.53   93.80   94.10   93.50    0.94
382    2600        347.01   4314.23   93.92   93.55   94.29    0.94
449    2800        286.45   4228.42   93.72   94.09   93.34    0.94
516    3000        258.55   4363.13   93.67   93.52   93.82    0.94
582    3200        224.76   4222.08   93.32   93.62   93.03    0.93
649    3400        257.28   4217.50   93.93   93.42   94.45    0.94
716    3600        277.44   4250.75   93.48   93.78   93.19    0.93
782    3800        252.77   4234.63   93.58   93.65   93.50    0.94
849    4000        200.09   4220.01   93.36   92.00   94.77    0.93
916    4200        208.35   4242.82   93.51   93.36   93.66    0.94
982    4400        202.76   4221.20   93.63   94.08   93.19    0.94
```

Figure 13: Data training with spacy

2.2 Results from the NER model

I later tested the trained model with a group of texts that were not part of the training data. I used displacy function to display the results of entities recognized by the model as shown in figure 14 below.

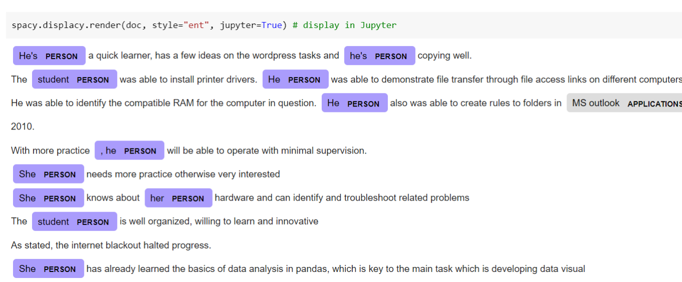

Figure 14: Entities displayed from model prediction

Conclusion

With enough training data, NER has the potential to perform even up to 0.1 accuracy.

References

[1].“Clustering text documents using k-means,” scikit-learn. https://scikit-learn.org/stable/auto$_{e}xamples/text/plot_{d}ocument_{c}lustering.html.$
[2]$.Li, H.(2009).TextClustering.In : LIU, L., ZSU, M.T.(eds)EncyclopediaofDatabaseSystems.Springer, Boston,$
$//doi.org/10.1007/978-0-387-39940-9_{4}15.$
[3]$.V.Kalyanarangan, \imath TextClustering : GetquickinsightsfromUnstructuredData, \jmath KDnuggets, Jun.27, 2017.http$
$//www.kdnuggets.com/2017/06/text-clustering-unstructured-data.html.$
[4]$.S.Li, \imath NamedEntityRecognitionwithNLTKandSpaCy, \jmath Medium, Aug.17, 2018.https$:
$//towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-$
$8c4a7d88e7da.$
[5]$.C.Marshall, \imath Whatisnamedentityrecognition(NER)andhowcanIuseit?, \jmath Medium, Jun.02, 2020.https$:
$//medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-$
$can-i-use-it-2b68cf6f545d$