Latent Semantic Analysis:
Applied Linear Algebra in Natural Language Understanding

Amy Reidy – 21/01/2022

Applied Linear Algebra (MATH09003)

1. Introduction

This project aims to describe how linear algebra can be applied to topic modelling in Natural Language Understanding (NLU). NLU is a subfield of Natural Language Processing (NLP), and the goal of NLU is to use computer algorithms to not just process text data, but to make sense of natural language data as it is spoken and written like a human would (Canonico & De Russis, 2018). NLU has many diverse real-world applications, including voice-controlled assistants like Apple's Siri, customer service chatbots, and machine translators. NLU is an important subset of machine learning as it allows us to analyse and gain understanding from large bodies of unstructured data.

Topic modelling is a NLU text-mining technique for extracting topics from text, and it can be used for information retrieval, categorizing documents and exploratory analysis of text datasets. It assumes that documents contain different topics, and these topics are made up of a collection of words. A foundational method of topic modelling is Latent Semantic Analysis (LSA) which is an unsupervised machine learning technique that was introduced by Deerwester et al. (1990) as a way of analysing a text corpus (a large collection of texts) to discover the hidden or latent topics in it. This method differs from traditional NLP as it uses statistical techniques and linear algebra to analyse text, rather than relying on any specification of rules or dictionaries, and it is based on the principle of distributional hypothesis - words that have similar meanings occur in analogous segments of text.

There are two main steps in conducting LSA:

1. Construction of a term-document matrix.
2. Dimensionality reduction of this matrix using Singular Value Decomposition (SVD). This process produces a topic-document matrix and a topic-term matrix.

2. Literature Review

LSA and other topic modelling methods such as Latent Dirichlet Allocation (LDA) are very popular in a wide range of fields as they allow for large quantities of text to be analysed very quickly. Thus, LSA can be a very efficient way to conduct literature reviews as the low-dimensional semantic space that SVD creates makes it easy to identify themes and trends across literature datasets. For example, Aryal et al. (2015) used LSA to conduct a longitudinal analysis of Healthcare Information Technology (HIT) research papers dating from 1990 to 2015 to identify trends in this field over three different time periods. Similarly, Hutchison et al. (2018) used LSA to analyse trends in academic research on Accounting Information Systems (AIS).

Another benefit is that it can produce new, data-driven categories that previously may not have been considered if there are not many well-established theories on the subject (Evangelopoulos et al., 2012). For example, Sehra et al. (2017) used LSA to discover emerging trends related to OpenStreetMap research. They analysed abstracts from 485 academic papers, and used SVD to identify five core research areas, as well as 50 trending topics, and produced recommendations for future research.

An advantage of using LSA, rather than other traditional methods such as manual content analysis, is that it is not only a much quicker process, but it also reduces subjective bias during the analysis. Hence, it can be helpful in fields such as political science. For instance, Valdez et al. (2018) analysed the transcripts of the 2016 U.S. presidential debates, and their results showed that the topics they identified with LSA matched the most frequent policy-related internet searches at the time.

LSA can also be used for tasks such as categorization, clustering, and predictive modelling, as it converts text into numerical data that is more suitable for these methods. For example, Maletic and Valluri (1999) used LSA to cluster software components by applying it to source code and the supporting documentation, whereas Wolfe and Goldman (2003) predicted students’ reasoning about a historical event using similarity indices generated by LSA.

3. Methodology

3.1 Data

To illustrate each of the steps involved in using LSA to uncover themes across documents, this project will analyse titles and abstracts from ten recent research papers (Appendix 1) in a Python Jupyter Notebook. The papers are all related to ‘anticipatory action’ (an emerging approach to disaster risk reduction), and they were chosen by selecting the first ten results from searching for the keywords “anticipatory action” on Google Scholar and filtering the results for only papers published in 2021.

3.2 Pre-processing

The first step in machine learning NLP tasks is to pre-process the text data to transform it into a suitable input format (Appendix 2). Thus, the following steps were taken to pre-process the raw text using the NLTK library (Loper & Bird, 2002):

- Tokenization - this is the process of separating documents into sentences and sentences into words.
- Removal of punctuation, symbols, numbers and ‘stop words’ (words such as ‘the’, ‘of’, ‘a’ that add little value to the sentence).
- Words were converted to all lowercase.
- Lemmatization - using an English dictionary, this function reduces words down to their root word, for example: ‘slept’, ‘sleeping’ and ‘sleeps’ would all return the value ‘sleep’.

For example, this is a sample sentence from one of the papers used in the project before pre-processing (MacLeod et al., 2021):

> *“The implication: there is potential value in seasonal forecasts, but to exploit it one must be prepared to play the long game.”*

And this is the sentence after it has been pre-processed:

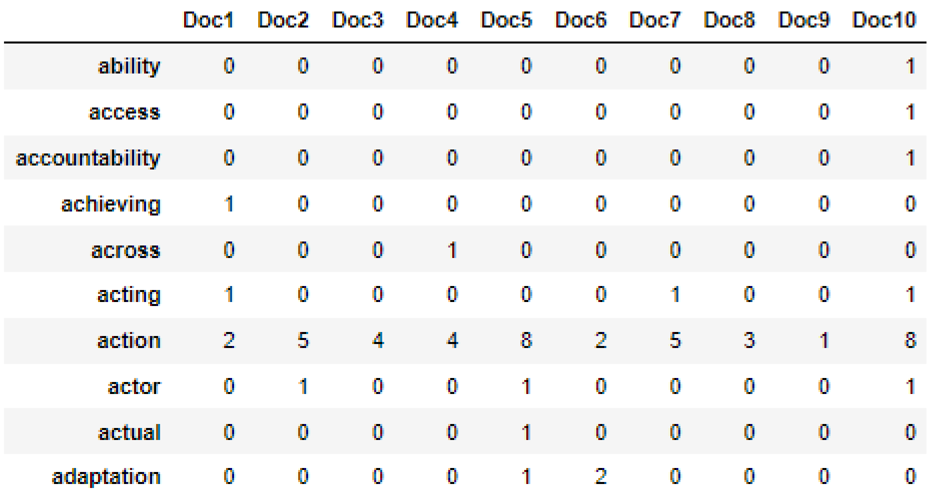
> *“implication potential value seasonal forecast exploit must prepared play long game”*

3.3 Document-Term Matrix

The next step is to create a document-term matrix by encoding the words as integers or floating-points in a process called vectorization. There are two common ways to do this using Scikit-Learn: CountVectorizer and TFIDF (Ramos, 2003).

CountVectorizer is a function that uses a Bag of Words (BoW) method to produce a sparse matrix containing word counts per sentence or document; it is called a bag-of-words as it disregards grammar, word order, etc.

Figure 1 shows a snapshot of the matrix produced by applying CountVectorizer to the text dataset. We can see that the word ‘action’ occurs in every document, which is not surprising as it was one of the search keywords.

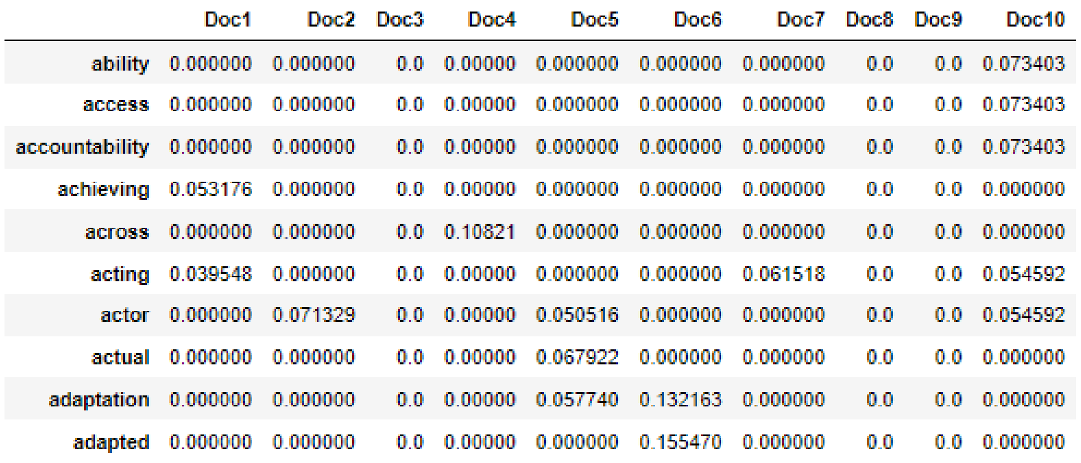


	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
ability	0	0	0	0	0	0	0	0	0	1
access	0	0	0	0	0	0	0	0	0	1
accountability	0	0	0	0	0	0	0	0	0	1
achieving	1	0	0	0	0	0	0	0	0	0
across	0	0	0	1	0	0	0	0	0	0
acting	1	0	0	0	0	0	1	0	0	1
action	2	5	4	4	8	2	5	3	1	8
actor	0	1	0	0	1	0	0	0	0	1
actual	0	0	0	0	1	0	0	0	0	0
adaptation	0	0	0	0	1	2	0	0	0	0

Fig. 1 Portion of CountVectorizer Matrix

One disadvantage of this method is that it is biased towards the most frequent words, so this project will use the TFIDF method to create a weighted term-document matrix (Appendix 3). TFIDF stands for Term Frequency - Inverse Document Frequency. While this method is also based on the frequency of a word in the corpus, it differs from CountVectorizer as it also provides a numerical representation of how important a word is for statistical analysis. TFIDF assumes that words that feature very frequently in a corpus and words that feature very rarely are both not statistically important for finding a pattern, and so it penalizes words that are too abundant or too infrequent. For this project, the parameter ‘tf_max’ was set to 0.9, which meant only the terms that featured in a maximum of 90% of the documents were included in the matrix; this resulted in the original search keywords “anticipatory” and “action” being filtered out.

Figure 2 shows a portion of the 638*10 document-term matrix produced by TFIDF that will be used in the next step. Like in the CountVectorizer matrix, the rows correspond to terms and there is a column for each document.



	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
ability	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.073403
access	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.073403
accountability	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.073403
achieving	0.053176	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000
across	0.000000	0.000000	0.0	0.10821	0.000000	0.000000	0.000000	0.0	0.0	0.000000
acting	0.039548	0.000000	0.0	0.00000	0.000000	0.000000	0.061518	0.0	0.0	0.054592
actor	0.000000	0.071329	0.0	0.00000	0.050516	0.000000	0.000000	0.0	0.0	0.054592
actual	0.000000	0.000000	0.0	0.00000	0.067922	0.000000	0.000000	0.0	0.0	0.000000
adaptation	0.000000	0.000000	0.0	0.00000	0.057740	0.132163	0.000000	0.0	0.0	0.000000
adapted	0.000000	0.000000	0.0	0.00000	0.000000	0.155470	0.000000	0.0	0.0	0.000000

Fig 2. Portion of Weighted Term-Document Matrix

3.4 Singular Value Decomposition

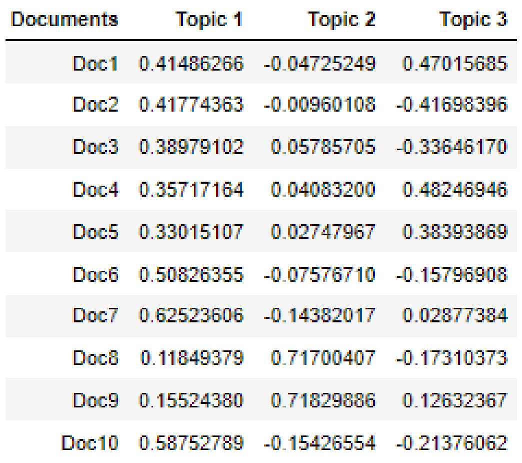
As the term-document matrix is usually extremely high-dimensional, it is very difficult to identify any patterns using this type of matrix. And so, the next step is to lower the rank of the matrix using one of the most common methods of matrix factorization in linear algebra, Singular Value Decomposition (SVD).

LSA uses a variant of SVD called Truncated SVD (Appendix 4) to reduce the dimensionality of the term-document matrix to a “semantic” space of low dimensionality by only computing a specific number of topics (this number is chosen by the analyst). It can be difficult to identify the optimal number of topics or themes for a certain corpus of text, but ideally the number of topics is large enough to give a good sense of the recurring themes in the data, but not too large that it ends up overfitting the data and the topics become too specific to individual documents (Deerwester et al., 1990). Accordingly, only 3 topics were chosen for this experiment as the dataset is quite small.

After the number of topics was chosen, Truncated SVD decomposed the term-by-document matrix into a product of three simpler matrices:

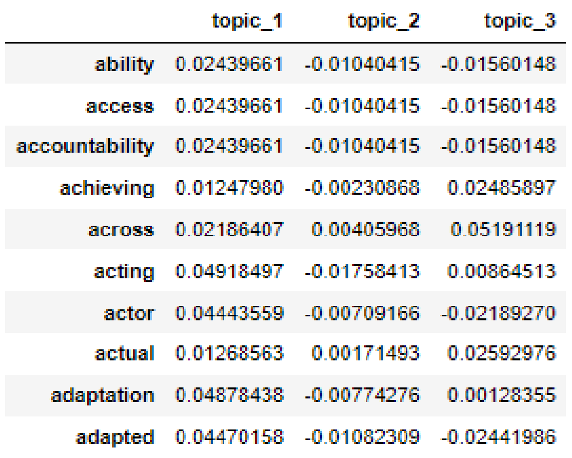
$$A = U * S * V^T$$

where A is the original m*n term-document matrix (Figure 2), U is an m*k term-topic matrix (Figure 5, Appendix 5), V is an n*k document-topic matrix (Figure 4, Appendix 6), and S is a diagonal k*k matrix of the singular values of A. In this case, m = 638 (# of terms), n = 10 (# of documents), and k = 3 (# of topics).



Documents	Topic 1	Topic 2	Topic 3
Doc1	0.41486266	-0.04725249	0.47015685
Doc2	0.41774363	-0.00960108	-0.41698396
Doc3	0.38979102	0.05785705	-0.33646170
Doc4	0.35717164	0.04083200	0.48246946
Doc5	0.33015107	0.02747967	0.38393869
Doc6	0.50826355	-0.07576710	-0.15796908
Doc7	0.62523606	-0.14382017	0.02877384
Doc8	0.11849379	0.71700407	-0.17310373
Doc9	0.15524380	0.71829886	0.12632367
Doc10	0.58752789	-0.15426554	-0.21376062

Fig 4. Document-Topic Matrix



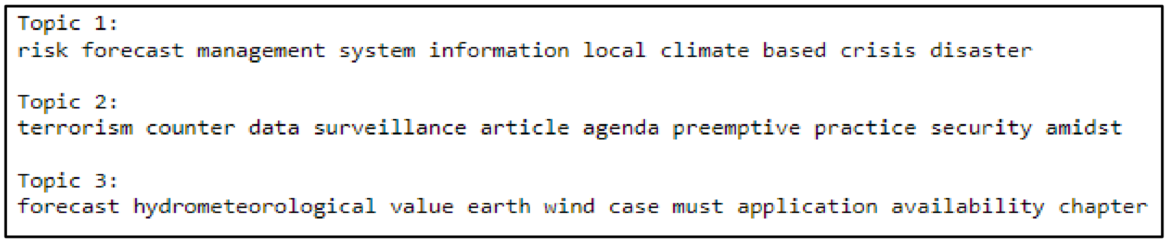
	topic_1	topic_2	topic_3
ability	0.02439661	-0.01040415	-0.01560148
access	0.02439661	-0.01040415	-0.01560148
accountability	0.02439661	-0.01040415	-0.01560148
achieving	0.01247980	-0.00230868	0.02485897
across	0.02186407	0.00405968	0.05191119
acting	0.04918497	-0.01758413	0.00864513
actor	0.04443559	-0.00709166	-0.02189270
actual	0.01268563	0.00171493	0.02592976
adaptation	0.04878438	-0.00774276	0.00128355
adapted	0.04470158	-0.01082309	-0.02441986

Fig 5. Truncated Term-Topic Matrix

From the document-topic matrix in Figure 4, it seems that Topic 1 is generally relevant to all documents, although documents 8 and 9 are the least related to it. It is also evident that these two documents are the most related to Topic 2, whereas this topic is not very relevant to the other documents. Furthermore, it appears that documents 8 and 9 are quite like each other, but very different from the rest of the dataset, and further analysis of the dataset could be undertaken by applying cosine similarity to the document vectors (the rows of the document-topic matrix) to evaluate how similar documents are to each other.

4. Results

Truncated SVD derived three topics from the term-document matrix, and Figure 6 shows the top ten words for each topic (Appendix 7), while Figures 7-9 show word clouds generated by the top thirty words in each topic.

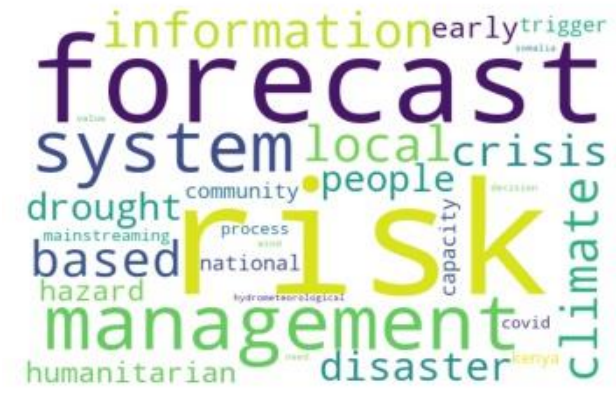


```
Topic 1:
risk forecast management system information local climate based crisis disaster

Topic 2:
terrorism counter data surveillance article agenda preemptive practice security amidst

Topic 3:
forecast hydrometeorological value earth wind case must application availability chapter
```

Fig. 6 Top Ten Words by Topic



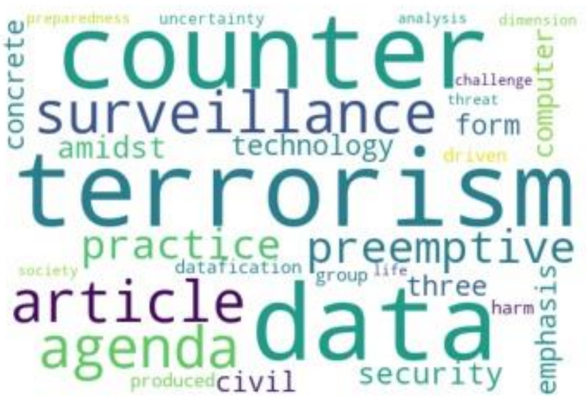


Fig. 7 Word Cloud of Topic 1

Fig. 8 Word Cloud of Topic 2

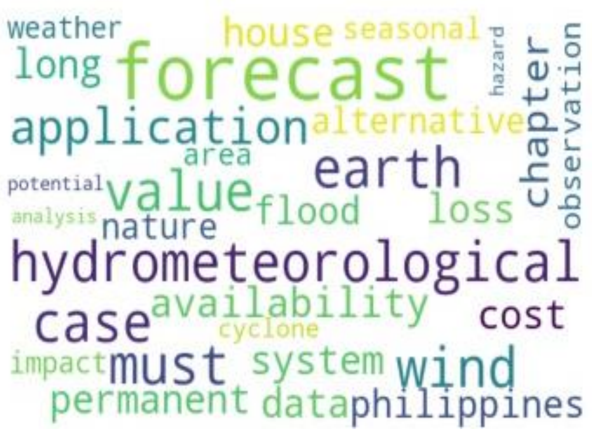


Fig. 9 Word Cloud of Topic 3

We can see that Topic 1 contains lots of general and technical terms relating to anticipatory action (including "disaster risk management system" which is a definition of anticipatory action), while Topic 2 focuses on the theme of security and anticipating terrorism, and Topic 3 is more concerned with nature and weather forecasting.

While there are already some identifiable themes, LSA works best with very large datasets, so more accurate results could be obtained by increasing the number of documents that are analysed.

5. Conclusion

This project shows how linear algebra can be applied to topic modelling through the LSA technique. By using SVD to reduce the noise and dimensionality of a term-document matrix, LSA allows analysts to quickly identify themes across large bodies of text. This was demonstrated by applying LSA to documents comprised of titles and abstracts from ten academic papers, and while this experiment still produced meaningful topics, these results could be further improved by augmenting the text dataset by including more research papers.

6. References

Aryal, A., Gallivan, M., & Tao, Y. Y. (2015). Using latent semantic analysis to identify themes in IS healthcare research.

Canonico, M., & De Russis, L. (2018). A comparison and critique of natural language understanding tools. *Cloud Computing*, *2018*, 120.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391-407.

Evangelopoulos, N., Zhang, X., & Prybutok, V. R. (2012). Latent semantic analysis: five methodological recommendations. *European Journal of Information Systems, 21*(1), 70-86.

Hutchison, P. D., Daigle, R. J., & George, B. (2018). Application of latent semantic analysis in AIS academic research. *International Journal of Accounting Information Systems*, *31*, 83-96.

Kherwa, P., & Bansal, P. (2017, September). Latent Semantic Analysis: An approach to understand semantic of text. In *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)* (pp. 870-874). IEEE.

Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

MacLeod, D., Kniveton, D. R., & Todd, M. C. (2021). Playing the long game: anticipatory action based on seasonal forecasts. *Climate Risk Management*, 100375.

Maletic, J. I., & Valluri, N. (1999, October). Automatic software clustering via latent semantic analysis. In *14th IEEE International Conference on Automated Software Engineering* (pp. 251-254). IEEE.

Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).

Sehra, S. S., Singh, J., & Rai, H. S. (2017). Using latent semantic analysis to identify research trends in OpenStreetMap. *ISPRS International Journal of Geo-Information*, *6*(7), 195.

Tonta, Y., & Darvish, H. R. (2010). Diffusion of latent semantic analysis as a research tool: A social network analysis approach. *Journal of Informetrics*, *4*(2), 166-174.

Valdez, D., Pickett, A. C., & Goodson, P. (2018). Topic modeling: latent semantic analysis for the social sciences. *Social Science Quarterly*, *99*(5), 1665-1679.

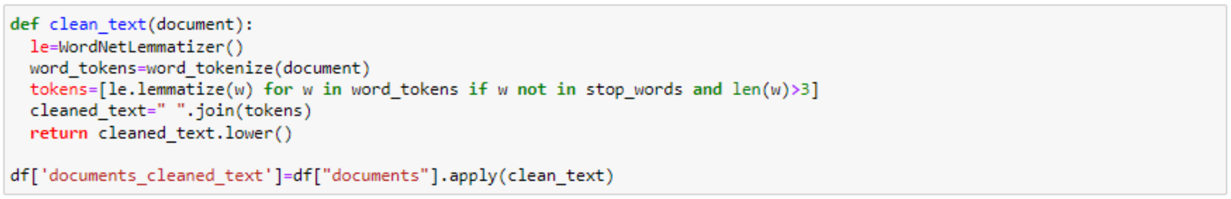
Wolfe, M. B., & Goldman, S. R. (2003). Use of latent semantic analysis for predicting psychological phenomena: Two issues and proposed solutions. *Behavior Research Methods, Instruments, & Computers*, *35*(1), 22-31.

7. Appendices

Appendix 1

	List of Documents Used in Experiment Dataset
1.	MacLeod, D., Kniveton, D. R., & Todd, M. C. (2021). Playing the long game: Anticipatory action based on seasonal forecasts. *Climate Risk Management*, *34*, 100375.
2.	Humphrey, A., Sheikh, M. A., Weingartner, L., & Levine, S. (2021). Understanding the role of anticipatory action in Somalia.
3.	Gettliffe, E. (2021). UN OCHA anticipatory action. Lessons from the 2020 Somalia pilot.
4.	Kruczkiewicz, A., McClain, S., Bell, V., Warrick, O., Bazo, J., Mason, J., ... & Horna, N. (2021). Earth Observations for Anticipatory Action: Case Studies in Hydrometeorological Hazards. In *Earth Observation for Flood Applications* (pp. 237-253). Elsevier.
5.	Vonk, D., van den Homberg, M., Kingma, N., Alkema, D., Teklesadik, A., Riquet, D., & van Aalst, M. (2021, April). Balancing permanent and forecast-based action to lessen wind-induced building damage in the Philippines. In *EGU General Assembly Conference Abstracts* (pp. EGU21-15290).
6.	Tozier de la Poterie, A., Clatworthy, Y., Easton-Calabria, E., Coughlan de Perez, E., Lux, S., & van Aalst, M. (2021). Managing multiple hazards: lessons from anticipatory humanitarian action for climate disasters during COVID-19. *Climate and Development*, 1-15.
7.	Mwangi, E., Taylor, O., Todd, M. C., Visman, E., Kniveton, D., Kilavi, M., ... & Colman, A. (2021). Mainstreaming forecast based action into national disaster risk management systems: experience from drought risk management in Kenya. *Climate and Development*, 1-16.
8.	Baker-Beall, C., & Mott, G. (2021). The new EU counter-terrorism Agenda: preemptive security through the anticipation of terrorist events. *Global Affairs*, 1-22.
9.	Kazansky, B. (2021). 'It depends on your threat model': the anticipatory dimensions of resistance to data-driven surveillance. *Big Data & Society*, *8*(1), 2053951720985557.
10.	Klassen, S., & Oxley, M. (2021). Information in Power-Connecting local responders to the risk information that they need.

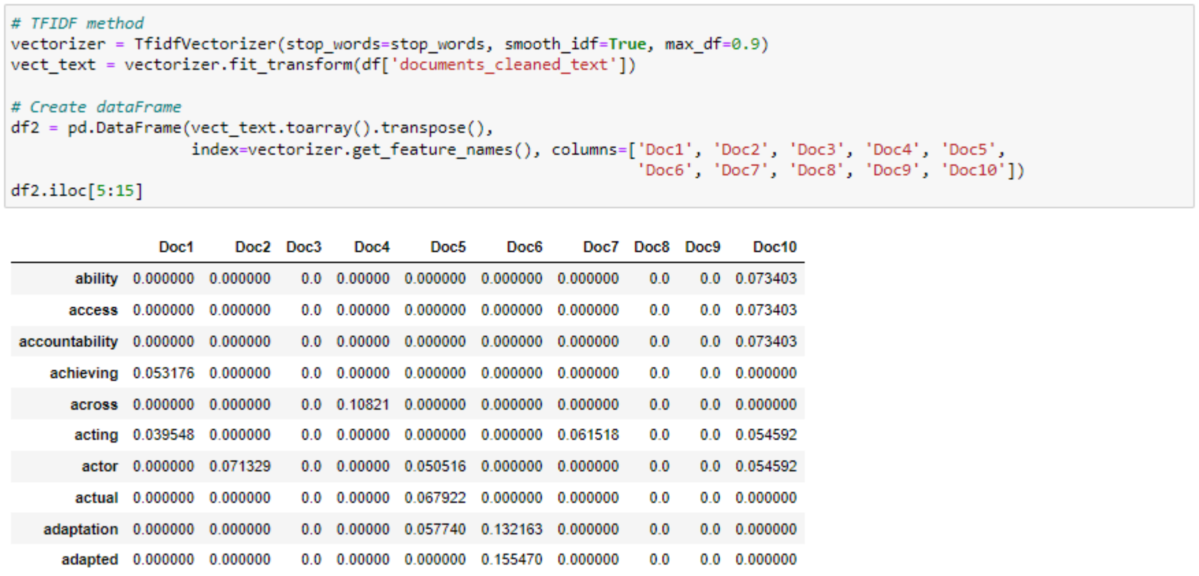
Appendix 2 – Data pre-processing code:



```python
def clean_text(document):
  le=WordNetLemmatizer()
  word_tokens=word_tokenize(document)
  tokens=[le.lemmatize(w) for w in word_tokens if w not in stop_words and len(w)>3]
  cleaned_text=" ".join(tokens)
  return cleaned_text.lower()

df['documents_cleaned_text']=df["documents"].apply(clean_text)
```

Appendix 3 – TFIDF matrix code:

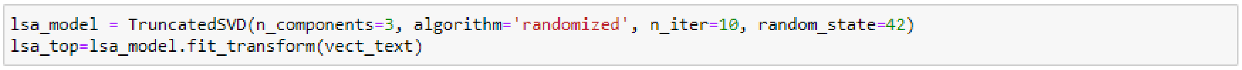


```python
# TFIDF method
vectorizer = TfidfVectorizer(stop_words=stop_words, smooth_idf=True, max_df=0.9)
vect_text = vectorizer.fit_transform(df['documents_cleaned_text'])

# Create dataFrame
df2 = pd.DataFrame(vect_text.toarray().transpose(),
                   index=vectorizer.get_feature_names(), columns=['Doc1', 'Doc2', 'Doc3', 'Doc4', 'Doc5',
                                                               'Doc6', 'Doc7', 'Doc8', 'Doc9', 'Doc10'])
df2.iloc[5:15]
```

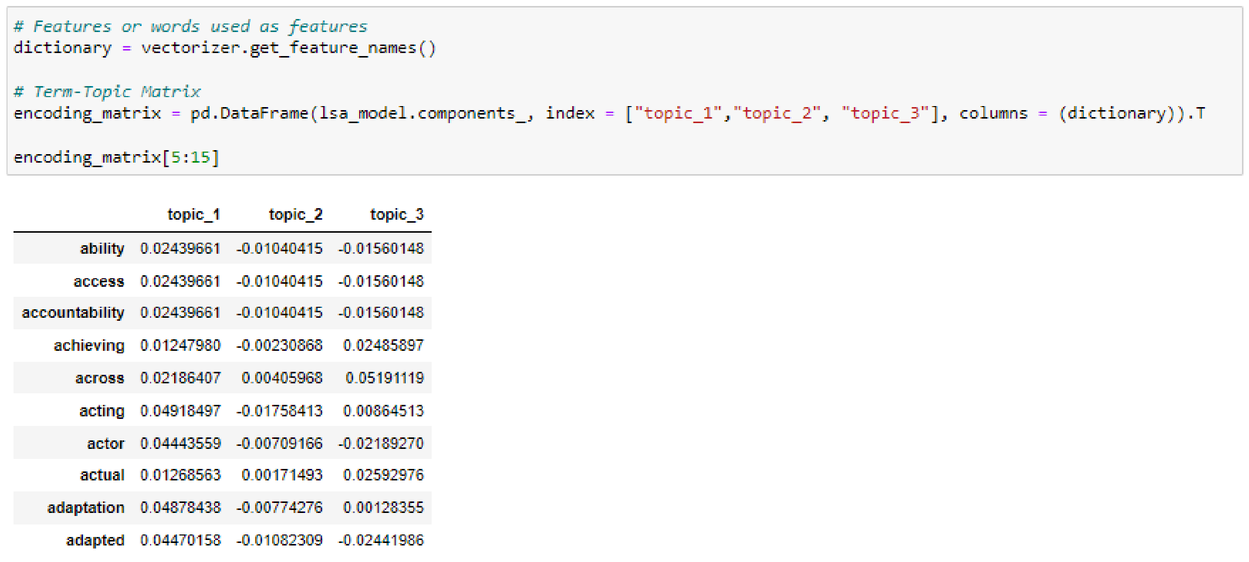
	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
ability	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.073403
access	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.073403
accountability	0.000000	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.073403
achieving	0.053176	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000
across	0.000000	0.000000	0.0	0.10821	0.000000	0.000000	0.000000	0.0	0.0	0.000000
acting	0.039548	0.000000	0.0	0.00000	0.000000	0.000000	0.061518	0.0	0.0	0.054592
actor	0.000000	0.071329	0.0	0.00000	0.050516	0.000000	0.000000	0.0	0.0	0.054592
actual	0.000000	0.000000	0.0	0.00000	0.067922	0.000000	0.000000	0.0	0.0	0.000000
adaptation	0.000000	0.000000	0.0	0.00000	0.057740	0.132163	0.000000	0.0	0.0	0.000000
adapted	0.000000	0.000000	0.0	0.00000	0.000000	0.155470	0.000000	0.0	0.0	0.000000

Appendix 4 – Truncated SVD code:



```python
lsa_model = TruncatedSVD(n_components=3, algorithm='randomized', n_iter=10, random_state=42)
lsa_top=lsa_model.fit_transform(vect_text)
```

Appendix 5 – Term-topic matrix code:



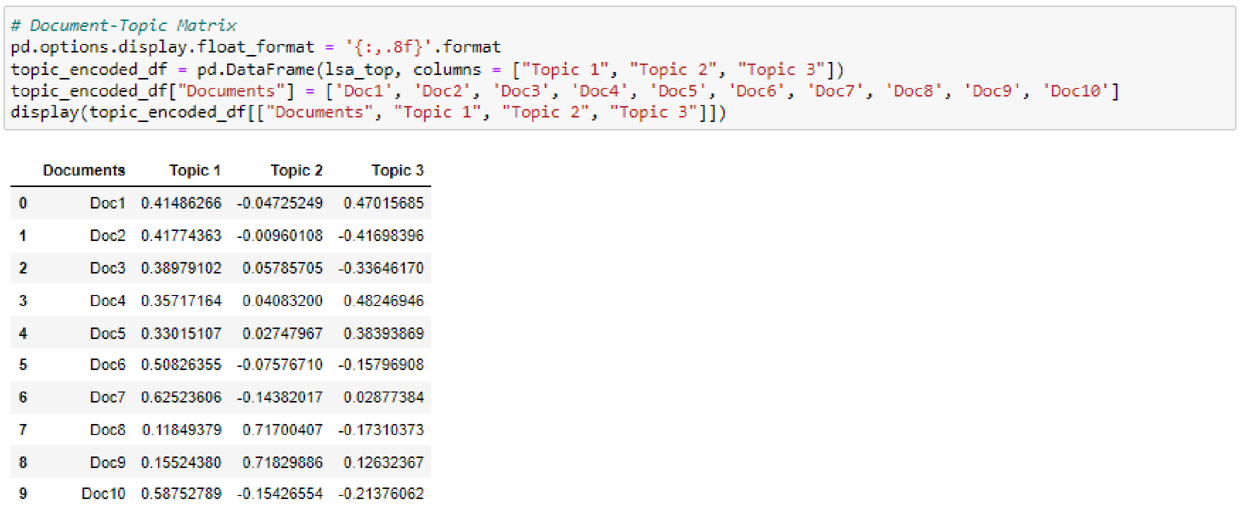
```
# Features or words used as features
dictionary = vectorizer.get_feature_names()

# Term-Topic Matrix
encoding_matrix = pd.DataFrame(lsa_model.components_, index = ["topic_1","topic_2", "topic_3"], columns = (dictionary)).T

encoding_matrix[5:15]
```

	topic_1	topic_2	topic_3
ability	0.02439661	-0.01040415	-0.01560148
access	0.02439661	-0.01040415	-0.01560148
accountability	0.02439661	-0.01040415	-0.01560148
achieving	0.01247980	-0.00230868	0.02485897
across	0.02186407	0.00405968	0.05191119
acting	0.04918497	-0.01758413	0.00864513
actor	0.04443559	-0.00709166	-0.02189270
actual	0.01268563	0.00171493	0.02592976
adaptation	0.04878438	-0.00774276	0.00128355
adapted	0.04470158	-0.01082309	-0.02441986

Appendix 6 - Document-topic matrix code:



```
# Document-Topic Matrix
pd.options.display.float_format = '{:,.8f}'.format
topic_encoded_df = pd.DataFrame(lsa_top, columns = ["Topic 1", "Topic 2", "Topic 3"])
topic_encoded_df["Documents"] = ['Doc1', 'Doc2', 'Doc3', 'Doc4', 'Doc5', 'Doc6', 'Doc7', 'Doc8', 'Doc9', 'Doc10']
display(topic_encoded_df[["Documents", "Topic 1", "Topic 2", "Topic 3"]])
```

	Documents	Topic 1	Topic 2	Topic 3
0	Doc1	0.41486266	-0.04725249	0.47015685
1	Doc2	0.41774363	-0.00960108	-0.41698396
2	Doc3	0.38979102	0.05785705	-0.33646170
3	Doc4	0.35717164	0.04083200	0.48246946
4	Doc5	0.33015107	0.02747967	0.38393869
5	Doc6	0.50826355	-0.07576710	-0.15796908
6	Doc7	0.62523606	-0.14382017	0.02877384
7	Doc8	0.11849379	0.71700407	-0.17310373
8	Doc9	0.15524380	0.71829886	0.12632367
9	Doc10	0.58752789	-0.15426554	-0.21376062

Appendix 7 – Top 10 words in each topic code

```
# most important words for each topic
vocab = vectorizer.get_feature_names()

for i, comp in enumerate(lsa_model.components_):
    vocab_comp = zip(vocab, comp)
    sorted_words = sorted(vocab_comp, key= lambda x:x[1], reverse=True)[:10]
    print("Topic "+str(i+1)+": ")
    for t in sorted_words:
        print(t[0],end=" ")
    print("\n")
```

```
Topic 1:
risk forecast management system information local climate based crisis disaster

Topic 2:
terrorism counter data surveillance article agenda preemptive practice security amidst

Topic 3:
forecast hydrometeorological value earth wind case must application availability chapter
```