Natural Language Processing

Final Assignment

How well do the topics generated by early NLP methods align with the Sustainable Development Goals (SDGs) defined by the United Nations?

July 11, 2023

Nazlıcan Eroğlu
Lavin Pallas

1 Introduction

The aim of this report is to analyse the OSDG dataset consisting of abstracts stemming from various documents and 17 sustainability goals which are used as labels. The dataset contains manually labeled text documents related to policy, research, and news reports which in total is derived from over 3000 documents. The 17 SDG goals consist of: poverty, zero hunger, good health and well-being, quality education, gender equality, clean water, clean energy, economic growth, industry innovation and infrastructure, reduced inequalities, sustainability, responsible consumption and production, climate action, sea life, life on land, peace justice, partnerships for the goals. As the topic of sustainability is becoming more important as many of the 17 goals have still not been fulfilled, this topic caught our attention and we decided to model sustainability goals with natural language processing techniques in order to figure out to what extend each of those 17 topics are acknowledged in a big dataset. We performed descriptive analysis, topic modeling using Latent Dirichlet Allocation and applied BERT embeddings and fine-tuning. We also employed k-means clustering with and without Principal Component Analysis on the BERT embeddings with label prediction as the objective . In subsequent sections the results are evaluated and discussed whereby each of the methods will be analysed in more detail further in this report

Figure 1: The 17 SDG goals

2 Data Collection

For our analysis, we will be using OSDG community dataset (OSDG et al., 2023) The objective of OSDG, an open-source initiative, is to combine different approaches to categorize any text document based on the Sustainable Development Goals (SDG) of UN. Following that, they aim to pin

down the objectives of various documents more clearly and hence address global issues such as poverty, climate change, and inequality in a more effective way. The dataset is publicly available, and it includes thousands of text documents that belongs to policy documents, research articles and news reports. In the most updated version, there are 41,689 text files in total. The interesting thing about the dataset is that it is manually labelled into most related SDGs by over 1,400 OSDG Community Platform participants from over 130 countries.

The source data for this dataset consists of paragraph-length text excerpts drawn from publicly available documents, including reports, policy documents, and publication abstracts. More than 3,000 of these documents are sourced from UN-related databases such as SDG-Pathfinder and SDG Library, many of which already have SDG labels associated with them. Each text excerpt typically comprises between three to six sentences and is approximately 90 words in length on average.

Since the aim of this report is to compare various fundamental natural language models, we believe that working with an already annotated data would be helpful in facilitating this comparison. On top of that, we think that the variety in the types of documents also enable us to apply different methods without being very biased towards one of them. Lastly, the size of the dataset is another reason behind the choice of this dataset.

3 Pre-processing

Before any techniques such as GloVe embeddings or Latent Dirichlet Allocation (LDA) can be applied to the data, first pre-processing is applied in order to remove stop words, commas, periods and unnecessary numbers. The data has also been tokenized which involves the splitting of the sentences into words whereby the order of words is disregarded. Subsequently, lemmatization is applied which involves the converting of words to their base form for simplicity and generalization purposes. In addition, the numbers which represent percentages have been converted into text as they usually represent useful information. The data pre-processing is essential as the noisy information of the data is not relevant to the analysis and will lead to disruption of the results of the LDA algorithm. With pre-processing, the results become more interpretable and accurate which leads to more effective capturing of the underlying topics.

One should note that we choose to not to pre-processing both for the data that we will use to fine-tune BERT and the data we use to get the embeddings. Unlike the LDA, BERT does not necessitate intricate and time-consuming preprocessing steps such as stemming, lemmatization, or stop-word removal, which are customary for models based on bag-of-words or TF-IDF. This is be-

cause BERT's architecture leverages the transformer mechanism, learning contextualized embeddings through self-attention. By reading text in both directions, BERT can gain a comprehensive understanding of the language's structure and semantics, grasping the full context of each word within a sentence.

The frequency of the most occuring words has been measured which is presented in Table 1. Furthermore, a bigram and a trigram have been generated in order to get an overview of the most common two and three word sentences. An example of the set of bigrams from the dataset is plotted in Table 2. Next, we will apply LDA as our first method in our dataset.

word	frequency
countries	11898
development	8636
social	8014
public	7508
policy	7367
health	6617
international	6562
water	6309

Table 1: Word Frequency

sentence
gender perspective
labour markets
market fishing
highly gendersegregated
male existence
gendersegregated terms
terms existence
from gender

Table 2: Bigram

4 Topic Modelling with LDA

Latent Dirichlet Allocation is a model for topic modeling with the aim of estimating the distribution of topics in each document and the distribution of words within each topic. The topic distribution for each document is computed by the frequency of word assignments. The model identifies topics from a dataset by randomly assigning a topic to each word in the dataset and by iteratively updating the topic according to the probability of the topic distribution. After the number of desired topics is set, the algorithm performs a number of iterations until convergence is reached.

The general formula is given by

$$P(W,Z,\theta,\phi,\alpha,\beta) = \prod_{j=1}^{M} P(\theta_j;\alpha) \prod_{i=1}^{K} P(\phi_i;\beta) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\phi Z_{j,t}) \tag{1}$$

whereby N refers to the number of words in a document, β refers to the document topic distribution whereas α refers to the topic word distribution, θ refers to the topic distribution for

document i, ϕ refers to the word distribution for topic k, $Z_{i,j}$ refers to the topic for the jth word in document i and finally $W_{i,j}$ refers to a specific word. The P represents the probability of generating the original document whereby the 2 refer to the Dirichlet distribution and the remaining 2 refer to the multinomial distribution. The 1st and 3rd terms refer to the distribution of the topics and the 2nd and fourth 4th the word distribution (Blei D, 2003).

To apply our preprocessed dataset in an LDA model, we started with a building a dictionary. Following that, we have created a corpus and applied the LDA model. Since this was the first time we are applying our dataset to a model, we decided to use whole dataset instead of separating the dataset into training, validation and test sets. At first, we decided to train the model for 20 topics. We run the model and saved the most related keywords for each of these topics. You can see the results in the Figure 2. We think that the separation among topics is promising when we compare with the SDG goals. For instance, topic 1 can easily be attributed to SDG 5: Gender and equality. Similarly, we see that topic 3 can be related to SDG 4: Quality education.

We have also created histogram of highest probability topics which is reported in Figure 3. We see that topic 17 is the topic with the highest probability whereas topic 1 has the lowest chance in figure 2. Next, we have plotted the probability distribution over documents of the highest probability topic in figure 4. It reaches its maximum of 0.4 at a little bit over 5000 documents. Lastly, we created the histogram for distribution of the active topics in a document in figure 5.

Furthermore, we have applied the pyLDAvis package to create an interactive tools where we can analyze the results of LDA better. Here, you can see the analysis regarding the topic 10 in figure 5. We can see the top-30 most relevant terms for each topic and the share of each topic in our token set.

	Keyword #1	Keyword #2	Keyword #3	Keyword #4	Keyword #5	Keyword #6	Keyword #7	Keyword #8	Keyword #9	Keyword #10	Keyword #11	Keyword #12	Keyword #13	Keyword #14	Keyword #15
Topic 1	'woman',	'income',	'gender',	'country',	'men',	'household',	'high',	'oecd',	'female',	'average',	'labour',	'inequality',	'rate',	'gap',	'employment',
Topic 2	'health',	'care',	'service',	'public',	'system',	'government',	'quality',	'patient',	'hospital',	'mental',	'primary',	'provide',	'medical',	'provider',	'private',
Topic 3	'child',	'education',	'country',	'oecd',	'high',	'age',	'rate',	'level',	'low',	'student',	'average',	'poverty',	'year',	'among',	'parent',
Topic 4	'water',	'management	'resource',	'plan',	'government',	'use',	'level',	'also',	'include',	'environmenta	'river',	'supply',	'cost',	'state',	'quality',
Topic 5	'policy',	'development	'national',	'research',	'regional',	'international	'include',	'management	'institution',	'government',	'framework',	'provide',	'chapter',	'system',	'university',
Topic 6	'food',	'``',	'use',	'may',	'indicator',	'country',	'household',	'indigenous',	'access',	'people',	'also',	'consumption	'number',	'mobile',	'migration',
Topic 7	'risk',	'et',	'al',	'impact',	'use',	'effect',	'.',	'change',	'scenario',	'result',	'disaster',	'level',	'estimate',	'model',	'flood',
Topic 8	'energy',	'climate',	'investment',	'development	'sector',	'finance',	'project',	'change',	'country',	'infrastructure	'private',	'emission',	'efficiency',	'also',	'support',
Topic 9	'country',	'million',	'per',	'total',	'increase',	'usd',	'waste',	'year',	'billion',	'region',	'world',	'export',	'production',	'africa',	'fish',
Topic 10	'data',	'information',	'provide',	'use',	'policy',	'process',	'report',	'develop',	'need',	'public',	'support',	'research',	'national',	'analysis',	'assessment',
Topic 11	'school',	'education',	'teacher',	'student',	'programme',	'system',	'teach',	'need',	'leader',	'learn',	'professional',	'group',	'also',	'training',	'performance'
Topic 12	'per',	'cent',	'cost',	'price',	'increase',	'high',	'electricity',	'rate',	'power',	'fuel',	'low',	'transport',	'country',	'tax',	'plant',
Topic 13	'de',	'poverty',	'school',	'line',	'la',	'et',	'index',	'poor',	'le',	'household',	'en',	'use',	'income',	'child',	'extreme',
Topic 14	'policy',	'economic',	'country',	'social',	'growth',	'political',	'inequality',	'poverty',	'economy',	'change',	'effect',	'corruption',	'show',	'result',	'crisis',
Topic 15	'international'	'right',	'law',	'human',	'security',	'criminal',		'global',	'united',	'crime',	'political',	'justice',	'violence',	'legal',	'treaty',

Figure 2: Table for LDA

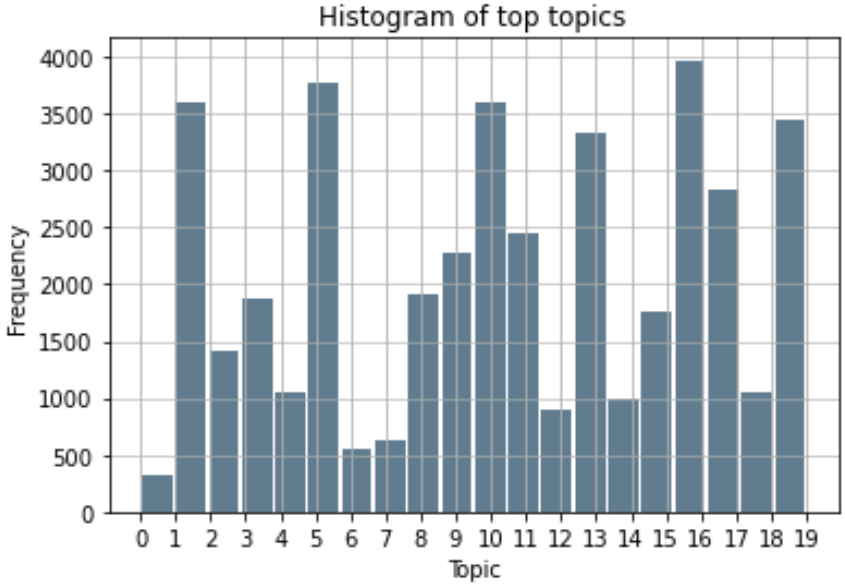

Figure 3: Histogram of Highest Probability Topics

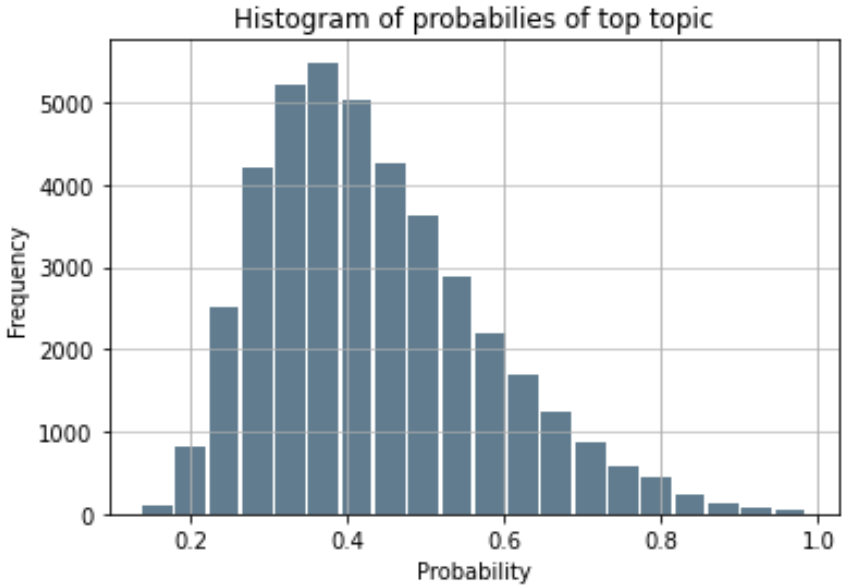

Figure 4: Histogram of Probabilities of Top Topic

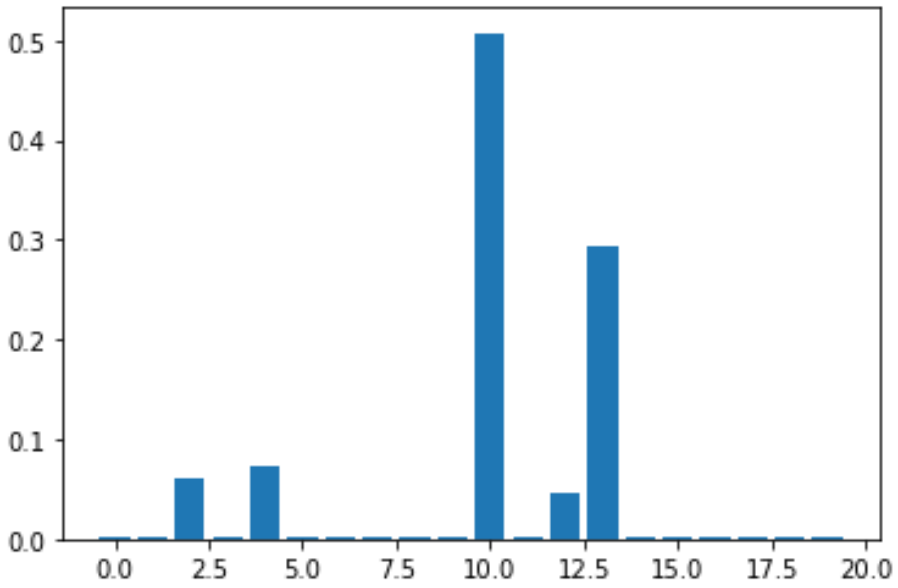

Figure 5: Topic Distribution

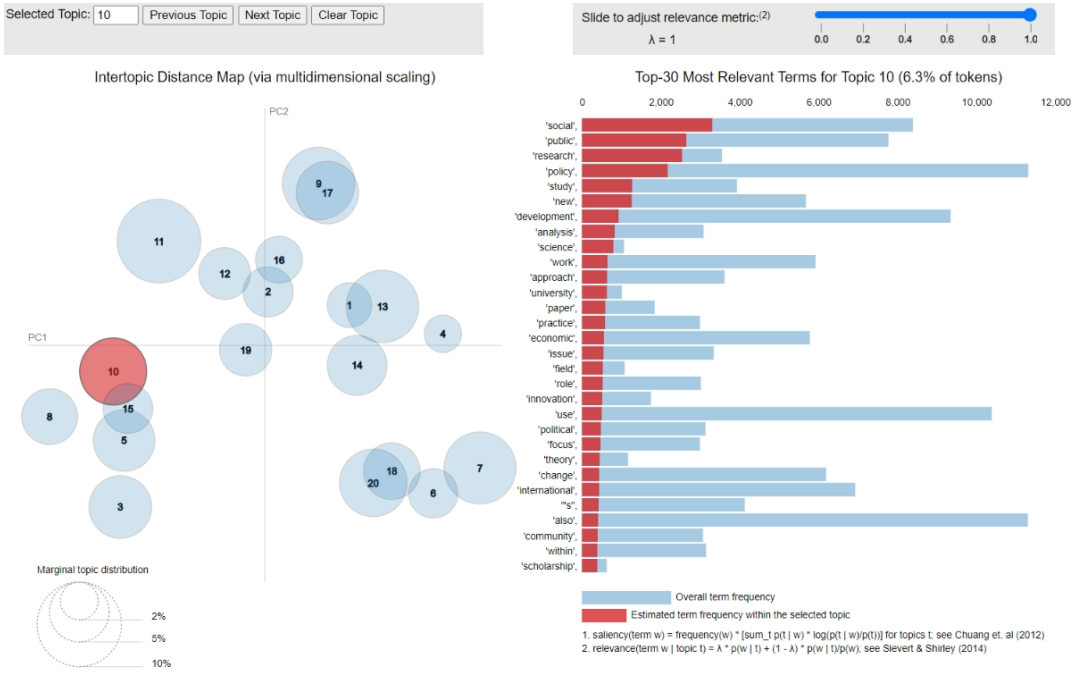

Figure 6: Description of topic 10 by using pyLDAvis package

4.1 Further analysis with LDA

When we check the figure 2 closely, we believe that only topic 13 and topic 7 seems to have an unclear context to some extend. That's why, we decided to do further cleaning. This time, we have also compared the topic coherence of different number of topic decisions to get the optimal number of topics.

In order to obtain clearer topics, further foreign stop words such as 'de', 'la', 'le', 'un' and 'les' which are french and spanish, have been removed. We have also plotted the topic coherence scores for different number of topics. We could not find any significant differences across number of topics. That's why, we present the results with 15 topics for comparison purposes. This time, we see that each topic has a set of keywords which are not stopwords/articles. It can be seen that topic 10 is related to the quality education goal, topic 5 is related to economic growth and topic 14 is related to good health and well-being. Although topic 15 seems to be related to quality education, it also contains keywords such as employment, unemployment and firm which seem to be related to economic growth, therefore it is not clear which SDG it belongs to.

Again, the pyLDAvis package has been applied to the further pre-processed dataset. The results for topic 10 is reported in Figure 8. In comparison to the initially pre-processed dataset which only excluded the English stopwords, the percentage of relevant tokens representing the topic is lower. One way to interpret this is that, topic 10 can be represented with fewer tokens which makes it a more focused topic. When we check the keywords, we can say that it is associated with SDG 15 the most. Topic 9 has also been depicted in Figure 9 where it seems that the percentage of relevant tokens representing the topic is higher. It appears that topic 9 is related to the SDG goal good health and well-being.

Finally, from the histogram of topics with the highest probability in Figure 10, it appears that the distribution of topics is less diverse compared to before. We think that this is a natural result as some of the topics which used to contain stopwords/articles do not exist anymore. Instead, LDA creates new topics with different keywords. If these topics are not prominent in the text, then the texts that LDA allocates among topics would gather around some already existing topics. In our case, we can see that topic 7 and 6 are such topics.

Ideally, we would like to compare these histograms with the correct distribution of the topics. However, since not all the texts are labelled correctly in our dataset, we cannot provide that. Next, we touch upon this problem for our further applications.

	Keyword #1	Keyword #2	Keyword #3	Keyword #4	Keyword #5	Keyword #6	Keyword #7	Keyword #8	Keyword #9	Keyword #10	Keyword #11	Keyword #12	Keyword #13	Keyword #14	Keyword #15
Topic 1	'land',	'price',	'forest',	'use',	'production',	'area',	'increase',	'agricultural'	'farm',	'fish',	'crop',	'farmer',	'change',	'may',	'also',
Topic 2	'social',	'public',	'research',	'study',	'political',	'analysis',	'theory',		'cultural',	'community',	'paper',	'practice',	'conflict',	'use',	'article',
Topic 3	'state',	'government	'internationa	'local',	'institution',	'authority',	'public',	'policy',	'national',	'role',	'agency',	'organisatior	'provide',	'council',	'federal',
Topic 4	'law',	'right',	'internationa	'human',	'legal',	'court',	'article',	'state',	'convention',	'rights',	'criminal',	'justice',	'european',		'treaty',
Topic 5	'tax',	'benefit',	'transfer',	'country',	'united',	'oecd',	'income',	'germany',	'revenue',	'high',	'kingdom',	'system',	'cash',	'denmark',	'states',
Topic 6	'food',	'trade',	'waste',	'product',	'market',	'export',	'sector',	'agricultural'	'production',	'economy',	'also',	'agriculture',	'economic',	'growth',	'investment',
Topic 7	'country',	'climate',	'developmen	'finance',	'support',	'develop',	'change',	'adaptation',	'also',	'project',	'national',	'need',	'include',	'provide',	'target',
Topic 8	'energy',	'cost',	'investment'	'technology',	'electricity',	'efficiency',	'sector',	'power',	'increase',	'use',	'emission',	'market',	'reduce',	'also',	'renewable',
Topic 9	'country',	'poverty',	'child',	'high',	'increase',	'level',	'population',	'region',	'people',	'rate',	'age',	'data',	'year',	'growth',	'low',
Topic 10	'school',	'education',	'student',	'teacher',	'child',	'learn',	'skill',	'data',	'provide',	'use',	'indicator',	'need',	'also',	'information	'programme',
Topic 11	'water',	'area',	'managemer	'river',	'include',	'use',	'service',	'quality',	'health',	'basin',	'fund',	'hospital',	'project',	'resource',	'also',
Topic 12	'policy',	'‴,	'review',	'chapter',	'approach',	'process',	'issue',	'reform',	'rule',	'economic',	'case',	'institutional	'political',	'change',	'new',
Topic 13	'public',	'government	'service',	'local',	'developmen	'system',	'plan',	'national',	'policy',	'managemer	'city',	'area',	'also',	'provide',	'urban',
Topic 14	'health',	'care',	'developmen	'policy',	'social',	'sustainable'	'mental',	'service',	'need',	'community'	'include',	'people',	'access',	'primary',	'change',
Topic 15	'oecd',	'education',	'high',	'country',	'low',	'rate',	'average',	'employmen	'student',	'level',	'secondary',	'among',	'university',	'unemploym	'firm',

Figure 7: Topic Distribution for further pre-processed data

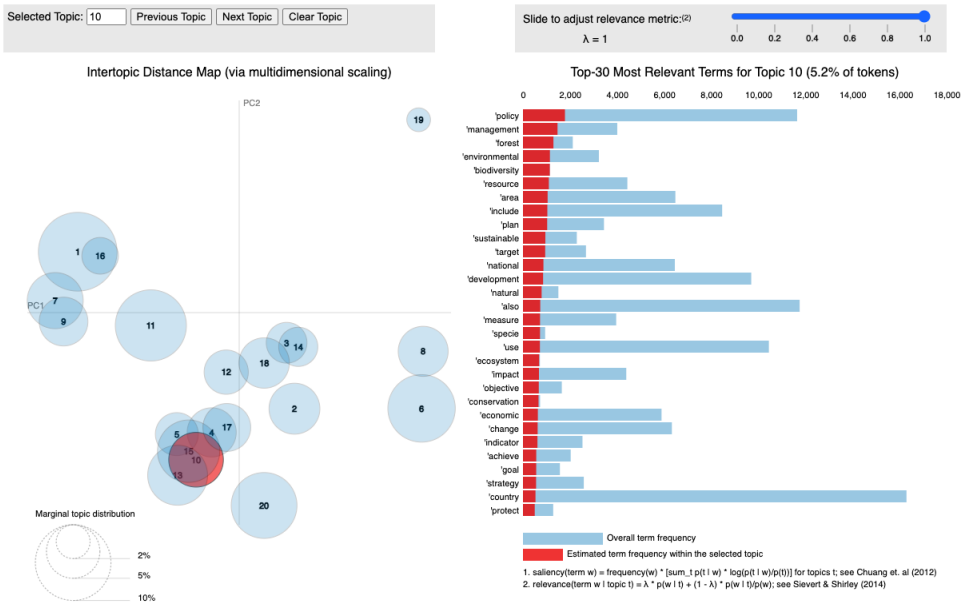

Figure 8: Description of topic 10 by using pyLDAvis package

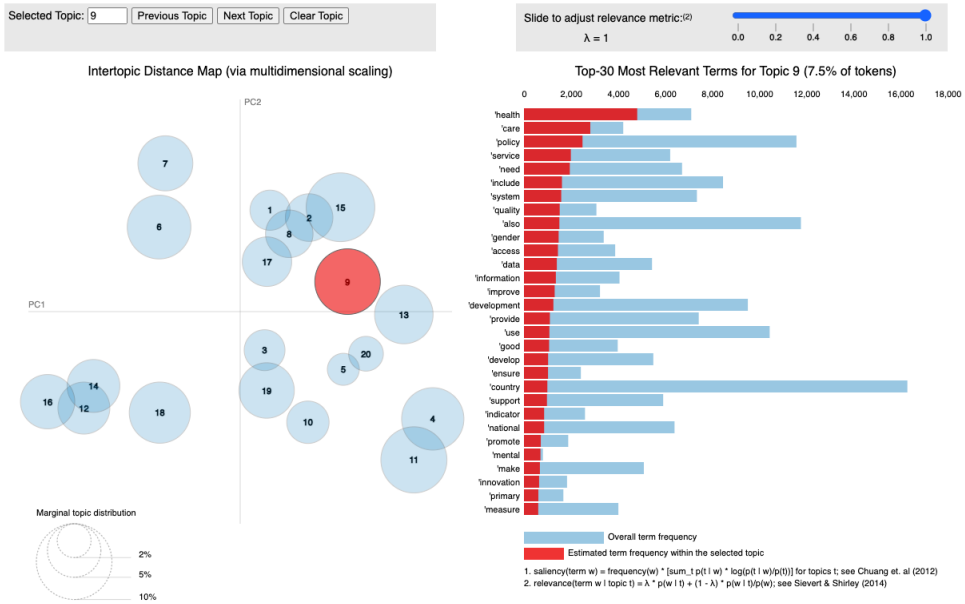

Figure 9: Description of topic 9 by using pyLDAvis package

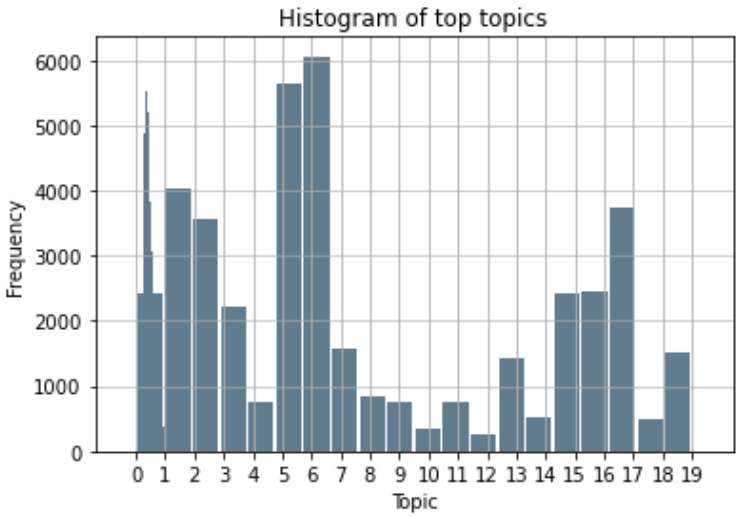

Figure 10: Histogram of Highest Probability Topics

5 Dataset with true labels

Next, we decided to create a new dataset where we have true labels for each text. The reason is twofold: We need to have true labels to be able to calculate accuracy for the upcoming approaches. Second, we need true labels during fine-tuning process of BERT so that it can update its' parameters with respect the the classification task that we describe.

In an effort to improve model efficiency and accuracy, we have curated a dataset consisting of 200 correctly labelled texts per SDG. The decision to pare down the dataset was made due to the fact that training the models on the full dataset was not yielding effective results and was contributing to low accuracy scores in the BERT classification model. By filtering the data and reducing its volume, we not only enhanced the model's performance but also decreased the computational power needed to run these models. A snapshot of the refined dataset is provided in Figure 6.

Index	text_id	text	sdg
0	ed13ca20769b783a9bcc7f4	Since these institutions in many countries plagued by widespread poverty d…	1
1	017ddae178f0e5fe3daecce	Other complementary factors—most importantly a recipient s entrepreneurial…	1
2	f7b9c8d70427b4f27d78a24	Figure 3.4 shows the poverty gap relative to the absolute poverty (account…	1
3	21b5c120c626abef6a8e87a	In 2012 more than 200 million adults worldwide were unemployed. Vulnerable…	1
4	bf51a418855d3688fc50af8	This increase coincided with the lowest income groups making income gains …	1
5	afef7651853913f49e00931	Specifically, the cost for a benefit of $2.50 a day for all older people a…	1
6	598ce3141ccf30345ebd80b	Care systems must also be established or strengthened, by fostering the pr…	1
7	e591318dfd77ddc7ebb4f58	In Latvia, the very large spikes in exits from SA benefits must probably b…	1
8	92c1805a7af0732a1282b48	This part of the population can be considered as potentially poor. In othe…	1
9	43dc9947a7195e6781f0a03	The economic crisis has halted a iong-term gradual decline in both inequal…	1

Figure 11: Head of the new dataset

6 BERT fine tuning

In our study, we fine-tuned the BERT model prior to extracting the embeddings. Fine-tuning is a crucial step in the application of pre-trained models like BERT. Essentially, it involves training the pre-existing model on our specific task, allowing the model to adjust and adapt to the nuances and specifics of our dataset. This is done by adding an additional layer relevant to the task at hand (for instance, a classification layer for a classification task) and training the model for a few more epochs. This way, the model is not learning from scratch but instead building upon an extensive knowledge base, making it more efficient and accurate. In this case, fine-tuning BERT enabled it to adjust its initial language understanding to better align with our dataset's context. Once the fine-tuning was completed, we were able to generate embeddings. These embeddings, rich in contextual information, facilitated more nuanced and precise model performance. We have decided to use uncased BERT model which uses 12 layers of transformers block with a hidden size of 768 and number of self-attention heads as 12 and has around 110M trainable parameters. After the fine tuning has been applied to the data, it is used for the clustering technique which will follow up later in the report.

The updating equation for fine tuning the parameters θ in the lower layer is given by

$$\theta_t^i = \theta_{t-1}^i - \eta^i \nabla_\theta^t J(\theta) \tag{2}$$

whereby η is the learning rate and is given by $\eta^{k-1} = \xi \eta^k$ and ξ is the decay factor.

The equation is an update rule based on gradient descent which is a common optimization algorithm for minimizing the objective function. By subtracting the product of the learning rate

and the gradient from the previous parameter values, the parameters are updated in the direction that minimizes the objective function. The learning rate decay factor aims to decrease the learning rate over iterations. By multiplying the current learning rate by the decay factor, a smaller learning rate is applied as the iterations progress. This can fine-tune the model by making smaller updates to the parameters as the optimization process continues.

In figure 7, the accuracy for the fine-tuned model is presented for each epoch. It appears that as the epochs increase the accuracy slightly increases and remains stable afterwards, while the loss decreases. One thing to note about this graph is that the loss of training set do not go below the loss of the validation set. This implies that we do not have an over-identification problem for this model.

Figure 12: Training Validation Loss, Accuracy for Fine-Tuning of uncased BERT

7 BERT embeddings

BERT word embeddings capture more detail in comparison to the Glove and Word2vec word embeddings as BERT embeddings follow a bidirectional approach where both the words to the left and right of the target word are considered in order to derive more information about the context. In addition, the embeddings are based on the transformer architecture, which involves self attention mechanisms to capture contextual dependencies among words. An encoder is stack with multiple layers in the model where each contains self attention and feed forward neural network modules.

For the creation of embeddings the process is similar as for fine-tuning the BERT model parameters, except that no additional layers on top of the pre trained BERT model are added which are designated for specific tasks using labelled data. The steps involved such as the tokenisation of the input text and the generation of various hidden states involving a weight matrix, remain the same. It should be noted that we obtained the embeddings at the text level as our aim is to classify texts into SDGs.

8 K-means clustering

K means clustering is a technique used to divide the observations of the dataset into k number of clusters with similarities. In this case the BERT embeddings are divided into 16 different clusters where the top 15 words are represented in order to reduce the dimensionality from the original BERT embeddings. A limitation of this technique is that the euclidean distance among each cluster is the same , which results in an equal representation of each cluster.

To apply k-mean clustering, first we have transformed our dataset in which each of the values in embeddings are saved as features. This way we created 768 features for each 3200 texts. Next, we have chosen the cluster size as 16 and maximum number of iterations the K-means algorithm will run for each single run as 300.

As the cluster numbers do not necessarily overlap with SDGs, we needed to find the most prominent SDGs in each cluster to be able to assess the performance of k-means clustering as a topic modelling approach. Since there is no standard way to measure this, we have followed two different approaches. First, since we know the true label of each text, we decided to group them. Next, we look into the most occuring cluster labels in each SDG group. The results can be seen in the next table. Each segment represents the SDG groups: Segment 1 represents SDG 1 labelled texts, segment 2 represents SDG 2 labelled texts and so on.

```
Segment 1: Most frequent value = 12
Segment 2: Most frequent value = 11
Segment 3: Most frequent value = 1
Segment 4: Most frequent value = 1
Segment 5: Most frequent value = 1
Segment 6: Most frequent value = 11
Segment 7: Most frequent value = 2
Segment 8: Most frequent value = 9
Segment 9: Most frequent value = 2
Segment 10: Most frequent value = 12
Segment 11: Most frequent value = 2
Segment 12: Most frequent value = 6
Segment 13: Most frequent value = 3
Segment 14: Most frequent value = 11
Segment 15: Most frequent value = 6
Segment 16: Most frequent value = 2
```

Figure 13: Most frequent cluster groups per SDG groups

Even though it is hard to interpret this table, it suggests that clustering is not as precise as to provide the true labels for each SDG since it can only group into 7 different cluster groups. Next, we decided to graph the top 15 words in each cluster groups as it may give us some clue about the mapping between cluster groups and SDG groups.

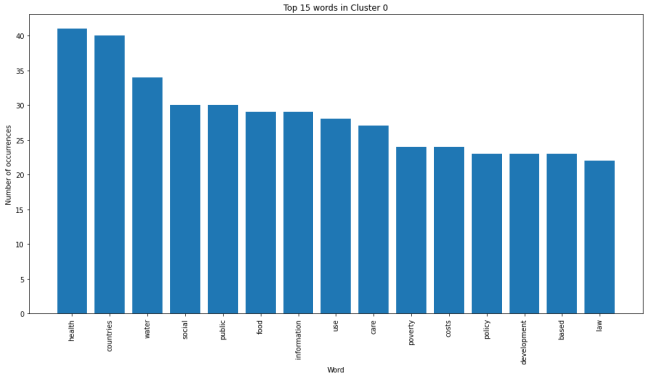

Figure 14: Most frequent words in cluster 0

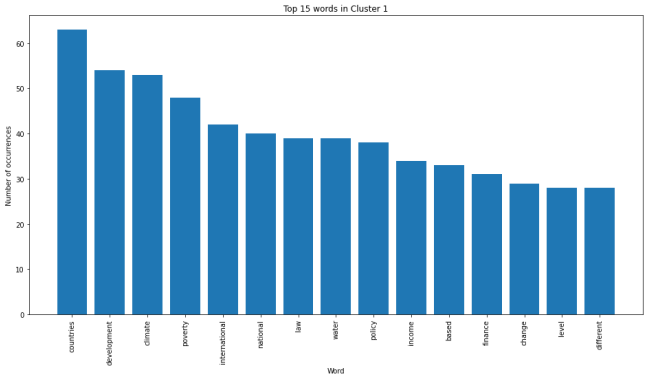

Figure 15: Most frequent words in cluster 1

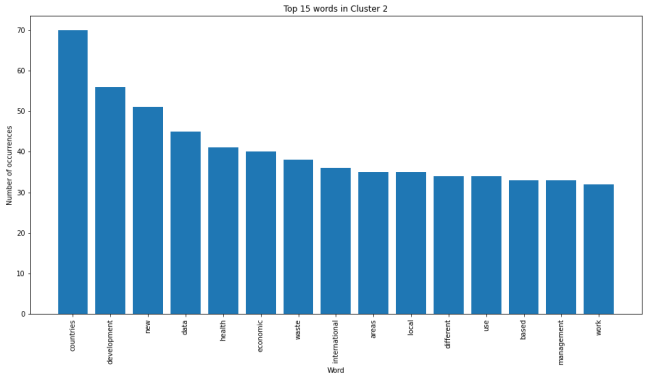

Figure 16: Most frequent words in cluster 2

In the figures 8, 9 and 10, the most frequent words for the first 3 clusters have been reported. As we see similar word in each cluster such as water, nation or development, it is again hard to say whether there is a mapping between SDG and cluster groups. In the next section, we propose a potential solution.

9 BERT embeddings PCA

As the BERT embeddings capture a lot of in depth detail for every word which results in high dimensional vectors, the dimensionality can be reduced using principal component analysis. While it is computationally inefficient to run a classification or apply the BERT embeddings directly to k means clustering without reducing the dimensionality, a potential disadvantage of the principal

```
Segment 1: Most frequent value = 12
Segment 2: Most frequent value = 14
Segment 3: Most frequent value = 1
Segment 4: Most frequent value = 6
Segment 5: Most frequent value = 12
Segment 6: Most frequent value = 11
Segment 7: Most frequent value = 1
Segment 8: Most frequent value = 9
Segment 9: Most frequent value = 6
Segment 10: Most frequent value = 12
Segment 11: Most frequent value = 4
Segment 12: Most frequent value = 6
Segment 13: Most frequent value = 4
Segment 14: Most frequent value = 11
Segment 15: Most frequent value = 14
Segment 16: Most frequent value = 2
```

(a) 5 features

```
Segment 1: Most frequent value = 2
Segment 2: Most frequent value = 13
Segment 3: Most frequent value = 2
Segment 4: Most frequent value = 10
Segment 5: Most frequent value = 0
Segment 6: Most frequent value = 4
Segment 7: Most frequent value = 2
Segment 8: Most frequent value = 2
Segment 9: Most frequent value = 10
Segment 10: Most frequent value = 2
Segment 11: Most frequent value = 8
Segment 12: Most frequent value = 2
Segment 13: Most frequent value = 8
Segment 14: Most frequent value = 4
Segment 15: Most frequent value = 13
Segment 16: Most frequent value = 10
```

(b) 10 features

Figure 17: Most frequent cluster groups with PCA per SDG groups

component analysis procedure is that since the variability among the word embeddings is captured, the common words are not taken into account in the procedure, which means that some of the information regarding the frequent occurring words will potentially be lost.

We have applied PCA with 5 and 10 features in our embedding space and then run the k-means clustering again. The most frequent cluster groups with PCA per SDG groups are reported in figure 11. On the left side, PCA with 5 features is applied whereas on the right PCA with 10 features have been used. With 5 features, we can see that k-means clustering can utilize 8 different clusters groups for SDG mapping. As this can signal a more precise mapping between SDGs and clusters, we checked the most occuring word in these clusters. Figure 12, 13 and 14 gives the most frequent words in first 3 clusters with PCA. We do not see a much difference between results with and without PCA.

9.1 Next steps

As the next step, we plan to remove the most occuring words among all texts as this might be the reason of seeing similar words across clusters. Figure 15 plots the wordcloud for most frequent words. These words are indeed overlaps with the most frequent words in our clusters.

We also aim to clean our data better while we include more texts with true labels. We plan to utilize LDA with this newer dataset while we aim for a higher number of topics.

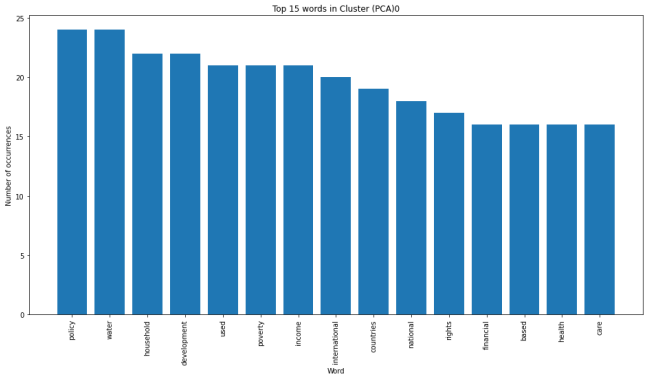

Figure 18: Most frequent words in cluster 0 with PCA

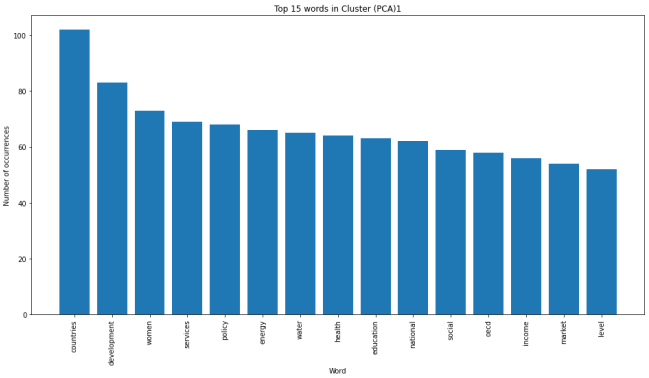

Figure 19: Most frequent words in cluster 1 with PCA

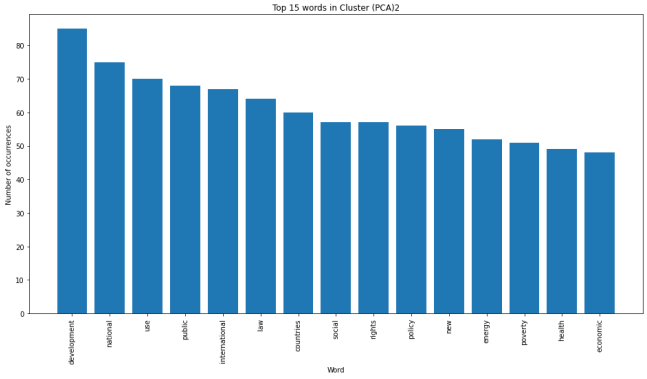

Figure 20: Most frequent words in cluster 2 with PCA

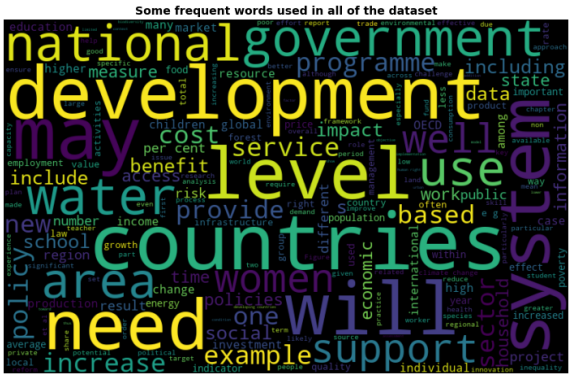

Figure 21: Most frequent words in all texts.

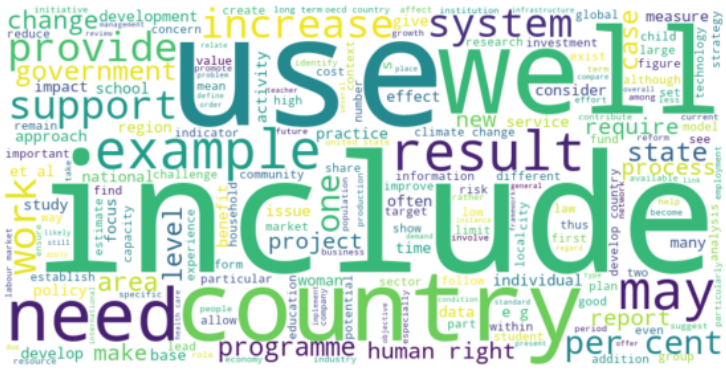

Figure 22: Most frequent words in all texts - further pre processed data

10 Conclusion

To conclude, the results from the LDA analysis using word2vec embeddings, indicate distinct topics that align with the goals however further data pre processing needs to be applied in order for the algorithm to model the topics more accurately. In addition, so far the K means clustering which uses the BERT fine tuning model as an input does not indicate clear results in terms of each of the goals. The next steps will aim to resolve this issue by further pre processing the data. Based on our analysis, we found that the LDA topic modeling approach provided good results in identifying the underlying topics in the dataset. The generated topics showed an association with different SDGs, highlighting the potential of LDA for categorizing text documents based on their SDG relevance. The various keywords were associated with one of the 17 SDG's to a different extend, some were

more clearly deduced than others depending on the number of keywords clearly associated to a specific SDG goal. Nevertheless, the k-means clustering approach, both with and without PCA, did not yield accurate mappings between the clusters and SDGs. The results were not as clear and interpretable as with LDA. This suggests that using BERT embeddings and k-means clustering alone may not be the most effective method for SDG categorization. To improve the k-means clustering approach the exclusion of common words would likely lead to better results, in order to reduce noise and enhance the differentiation between clusters. Additionally a larger dataset with more diverse text documents would improve the clustering results and enable more accurate mapping between clusters and SDGs however at this time we only had this dataset available. Another issue was that the level of agreement is not always as consistent, and the labels are not precise. As a consequence the results were not as consistent as we hoped for, for the k-means clustering method with PCA and Bert embeddings.

Our analysis provides valuable insights into the application of natural language models and topic modeling techniques for categorizing text documents based on the SDGs. By leveraging these techniques, we can enhance our understanding of the global issues addressed by the SDGs and contribute to more effective solutions for sustainable development.

11 Appendix

References

Blei D, Ng A, J. M. (2003, April). Latent dirichlet allocation. https://zenodo.org/record/7816403.ZF6Fa3ZBxXR.

OSDG, U. I. S. A. Lab, and PPMI (2023, April). Osdg community dataset (osdg-cd). https://zenodo.org/record/7816403.ZF6Fa3ZBxXR.

12 Code

```
# -*- coding: utf-8 -*-
"""
Created on Wed May  3 16:22:39 2023
Title: NLP project on SDG
@author: Nazlican
"""
#install packages
#installed nltk and other packages via pip by using console'
import nltk
import pandas as pd
dataset = pd.read_csv('columns_arranged.csv')
print(dataset.shape)
dataset.head()

#We will use package NLTK to do tokenization first
nltk.download('punkt')
from nltk import word_tokenize,sent_tokenize

example = """Founded in 2002, SpaceX's mission is to enable humans to become a spacefaring
species by building a self-sustaining city on Mars. In 2008, SpaceX's Falcon 1 became the f
liquid-fuel launch vehicle to orbit the Earth."""
sent_tokenize(example) #This splits documents into sentence
word_tokenize(example) #This splits documents into words, and punctuation.
#We thought using punkt package would be a better choice because it also saves punctuation
seperately.
#So we will be able to delete them easily whenever we want.

dataset['sentence_split'] = dataset['text'].apply(sent_tokenize)
dataset['word_split'] = dataset['text'].apply(word_tokenize)
dataset.to_csv('tokenized_data.csv')

#Stop words
nltk.download() #download all the popular packages
from nltk.corpus import stopwords
print(stopwords.words('english'))
stop_words = set(stopwords.words('english'))
# Remove stop words from the "word_split" column
dataset['word_split'] = dataset['word_split'].apply(lambda x: [word for word in
```

```
x if word.lower() not in stop_words])
dataset.to_csv('tokenized_data_without_stopwords.csv')

#Punctuation
import string
punctuations = set(string.punctuation)
dataset['word_split'] = dataset['word_split'].apply(lambda x: [word for word in x if word n
in punctuations])

# When I check the word split column, I saw that there are still elements like , \, ’, ‘ or
# Update the punctuation set and do again.

punctuations = set(string.punctuation + '\"’‘©')
dataset['word_split'] = dataset['word_split'].apply(lambda x: [word for word in
x if word not in punctuations])
dataset.to_csv('tokenized_data_without_punctuation_stopwords.csv')

#Remove numbers: I am not sure whether I should do that because some years might
have importance. Ask this one in the lecture
#Answer: Remove numbers
import re
dataset['word_split'] = dataset['word_split'].apply(lambda x: [word for word in x
if not re.match(r'\d+', word)])


#It is also good to have one more column with lemmatization so that we can compare
the quality of both choices. I will again use NLTK package.
#We could simply use function lemmatizer.lemmatize() to do that but this function
assumes that the word is always a noun. That's why, I wanted to try
# lemmatizer.lemmatize(word, pos) which also takes the type of word into account.

#Lemmatization
from nltk.stem import WordNetLemmatizer
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = []
    for word, tag in nltk.pos_tag(word_tokenize(text)):
        if tag.startswith('NN'):
            pos = 'n'
```

```
        elif tag.startswith('VB'):
            pos = 'v'
        else:
            pos = 'a'
        lemmatized_token = lemmatizer.lemmatize(word, pos)
        lemmatized_tokens.append(lemmatized_token)
    return lemmatized_tokens

dataset['lemmatized'] = dataset['word_split'].apply(lambda x: [word.lower() for word
in lemmatize_text(' '.join(x))])
dataset.to_csv('tokenized_data_without_punctuation_stopwords_lemma.csv')

#This also deals with the capital letters.


# -*- coding: utf-8 -*-
"""
Created on Wed May 12 16:22:39 2023
Title: NLP project on SDG
This time we will implement LDA
@author: Nazlican
"""
#Packages:
import pandas as pd
import numpy as np
import nltk # Used to
import gensim.corpora as corpora
import matplotlib.pyplot as plt

from pprint import pprint
from gensim.corpora.dictionary import Dictionary
from gensim.models import LdaModel
from tqdm import tqdm
from gensim import corpora

#Dataset
dataset = pd.read_csv('LDA/tokenized_data_without_punctuation_stopwords_lemma.csv')
print(dataset.shape)
dataset.head()

token_lists = dataset['lemmatized'].tolist()
```

```
token_lists = [d.split() for d in token_lists]

dictionary = corpora.Dictionary(token_lists)

dictionary.filter_extremes(no_below=15, no_above=0.5, keep_n=100000)
_ = dictionary[0] # need this line to force-load the data in the kernel
id2word = dictionary.id2token

corpus = [dictionary.doc2bow(doc) for doc in token_lists]

lda_model = LdaModel(corpus, id2word=id2word,num_topics=20, decay = 0.6,
minimum_probability=0.001)
for topic in lda_model.print_topics():
    print(topic)
#it seems that we have reasonable words in each topic
n_topics = 15
n_keywords = 15
topic_keywords,topic_keyvalues = [],[]
for (topic, values) in lda_model.print_topics(n_topics, n_keywords):
    temp_list_keywords,  temp_list_keyvalues = [],[]
    for value in (str(values).split()):
        if "*" in value:
            value = value.split("\"")
            value_keyword = float(value[0][:-1])
            keyword = value[1]
            temp_list_keywords.append(keyword)
            temp_list_keyvalues.append(value_keyword)
    topic_keywords.append(temp_list_keywords)
    topic_keyvalues.append(temp_list_keyvalues)


# save the output of the model in a dataframe. You can also store the list of
"topic_keyvalues" to save the estimated values.
df_to_save = pd.DataFrame(topic_keywords, index =
[f"Topic {topic_number+1}" for topic_number in range(n_topics)],
            columns = [f"Keyword #{index+1}" for index, _ in enumerate(topic_keywords[0])])

df_to_save.to_csv("results_LDA_model.csv")

#Visualization
#pip install pyLDAvis
```

```
from IPython.display import display
import pyLDAvis.gensim_models as gensimvis
import pyLDAvis
pyLDAvis.enable_notebook()

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

vis_data = gensimvis.prepare(lda_model, corpus, dictionary, sort_topics=False)
display(pyLDAvis.display(vis_data))
pyLDAvis.save_html(vis_data, 'lda.html')
#last line of code saves a html files in the directory
#where i can interactively examine the model. very useful!

#The following code is taken from Week_3_Latent_Dirichlet_Allocation.ipynb
#that is provided in the canvas.
topics = lda_model[corpus[2]] # 0 denotes the document
plt.bar(list(zip(*topics))[0], list(zip(*topics))[1])

topic_dist = pd.DataFrame(columns = ['topics', 'topic_list', 'top_topic','top_topic_prob'])
topic_dist.topics = lda_model[corpus]
topic_dist.topic_list = topic_dist.topics.apply(lambda y: [x[0] for x in y])
topic_dist.top_topic = topic_dist.topics.apply(lambda y: max(y, key = lambda x: x[1])[0])
topic_dist.top_topic_prob = topic_dist.topics.apply(lambda y: max(y, key = lambda x: x[1])[
print(topic_dist.head())

topic_dist.top_topic_prob.plot.hist(grid = True, bins = 20, rwidth = 0.9, color = '#607c8e'
plt.title('Histogram of probabilies of top topic')
plt.xlabel('Probability')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)

topic_dist.top_topic.plot.hist(grid = True, bins = 20, rwidth = 0.9, color = '#607c8e')
plt.title('Histogram of top topics')
plt.xlabel('Topic')
plt.ylabel('Frequency')
plt.xticks(range(20))
plt.grid(axis='y', alpha=0.75)
```

Listing 1: bigram

```
# Load required packages
```

```r
install.packages('SnowballC')
install.packages('stringr')
install.packages('gsubfn')
library(SnowballC)
library(stringr)
library(gsubfn)
install.packages('tm', dependencies = TRUE)
library(tokenizers)
library(writexl)
install.packages('tokenizers')
install.packages('openxlsx')
library(openxlsx)
install.packages('english')
library(english)
library(magrittr)
install.packages("scales")
library(scales)

data <- data['text']

library(english)

library(english)
library(readr)

library(english)
library(readr)

# Function to detect and convert percentages to words in a data frame
convert_percentages_to_words <- function(data) {
  converted_data <- data

  # Iterate over columns in the data frame
  for (col in names(data)) {
    # Check if the column contains character values
    if (is.character(data[[col]])) {
      # Identify values that represent percentages
      is_percentage <- grepl("%", data[[col]])

      # Convert percentages to words
      converted_data[[col]][is_percentage] <- sapply(data[[col]][is_percentage], function(x) {
        value <- parse_number(x)
        if (!is.na(value)) {
          english(value) %>% paste("percent")
        } else {
          x
        }
      })
    }
  }

  return(converted_data)
}


converted_data <- convert_percentages_to_words(data)
print(converted_data)
```

```r
# Remove digits
converted_data <- gsubfn("[[:digit:]]+", "", converted_data)
# Remove punctuation
converted_data <- str_replace_all(converted_data, "[[:punct:]]", "")

# Remove stopwords and extra whitespace
data_words <- unlist(str_split(converted_data, "\\s+"))
data_words <- data_words[!data_words %in% converted_data]
datan <- paste(data_words, collapse = " ")

# Stem the words
data <- SnowballC::wordStem(datan)


# Remove extra whitespace
data <- str_trim(data)

# Remove any remaining whitespace
data <- gsub("\\s+", " ", data)

tokenized_data <- tokenize_words(data)

tokenized_text <- strsplit(data, " ")

# Calculate the frequency of each word
word_freq <- table(unlist(tokenized_text))

# Define a threshold for rare words
threshold <- 1

# Create a new list of tokenized text without rare words
filtered_text <- lapply(tokenized_text, function(x) {
  x[x %in% names(word_freq)[word_freq <= threshold]] <- ""
  x[x != ""] # Remove empty elements
})

clean_text <- data %>%
  tibble(data = .) %>%
  unnest_tokens(word, data) %>%
  anti_join(stop_words)

filtered_text <- lapply(clean_text, function(x) {
  x[x %in% names(word_freq)[word_freq <= threshold]] <- ""
  x[x != ""] # Remove empty elements
})

tokens <- tokens(filtered_text)

# Generate bi-grams
bigrams <- tokens_ngrams(tokens, n = 2, concatenator = "_")
print("Bi-grams:")
print(bigrams)

trigrams <- tokens_ngrams(tokens, n = 3, concatenator = "_")
print("Tri-grams:")
print(trigrams)
```

```
# Save the cleaned data
my_df <- data.frame(filtered_text)

# write data frame to Excel file
write.xlsx(my_df, "updated3.xlsx", sheetName = "Sheet1", rowNames = FALSE)

#write_xlsx(filtered_text , "new.xlsx")
writexl::write_xlsx(my_df, "data.xlsx")
```

Listing 2: word frequency

```
library(quanteda)
library(stringi)

data <- data['text']

# Remove punctuation
data <- str_replace_all(data, "[[:punct:]]", "")


corpus <- corpus(data)

# Convert the corpus to a character vector
text <- as.character(corpus)

# Preprocess the text
tokens <- tokens(text, lowercase = TRUE)
tokens <- tokens_remove(tokens, stopwords("english"))
tokens <- tokens_remove(tokens, pattern = "[[:punct:]]")  # Remove punctuation using regex

# Create a document-feature matrix
dfm <- dfm(tokens)

# Get the frequency of words
word_freq <- colSums(dfm)

# Sort the word frequency in descending order
sorted_freq <- sort(word_freq, decreasing = TRUE)

# Print the top 10 words and their frequencies
top_words <- head(sorted_freq, 10)
print(top_words)
```

Listing 3: conversion of percentages into text and pre porcessing

```
# Load required packages
install.packages('SnowballC')
install.packages('stringr')
install.packages('gsubfn')
library(SnowballC)
library(stringr)
library(gsubfn)
install.packages('tm', dependencies = TRUE)
library(tokenizers)
library(writexl)
install.packages('tokenizers')
install.packages('openxlsx')
```

```r
library(openxlsx)
install.packages('english')
library(english)
library(magrittr)
install.packages("scales")
library(scales)

data <- data['text']

library(english)

library(english)
library(readr)

library(english)
library(readr)

# Function to detect and convert percentages to words in a data frame
convert_percentages_to_words <- function(data) {
  converted_data <- data

  # Iterate over columns in the data frame
  for (col in names(data)) {
    # Check if the column contains character values
    if (is.character(data[[col]])) {
      # Identify values that represent percentages
      is_percentage <- grepl("%", data[[col]])

      # Convert percentages to words
      converted_data[[col]][is_percentage] <- sapply(data[[col]][is_percentage], function(x) {
        value <- parse_number(x)
        if (!is.na(value)) {
          english(value) %>% paste("percent")
        } else {
          x
        }
      })
    }
  }

  return(converted_data)
}


converted_data <- convert_percentages_to_words(data)
print(converted_data)


# Remove digits
converted_data  <- gsubfn("[[:digit:]]+", "", converted_data )
# Remove punctuation
converted_data  <- str_replace_all(converted_data , "[[:punct:]]", "")

# Remove stopwords and extra whitespace
data_words <- unlist(str_split(converted_data , "\\s+"))
data_words <- data_words[!data_words %in% converted_data]
datan <- paste(data_words, collapse = "␣")
```

```
# Stem the words
data <- SnowballC::wordStem(datan)


# Remove extra whitespace
data <- str_trim(data)

# Remove any remaining whitespace
data <- gsub("\\s+", " ", data)

tokenized_data <- tokenize_words(data)

tokenized_text <- strsplit(data, " ")

# Calculate the frequency of each word
word_freq <- table(unlist(tokenized_text))

# Define a threshold for rare words
threshold <- 1

# Create a new list of tokenized text without rare words
filtered_text <- lapply(tokenized_text, function(x) {
  x[x %in% names(word_freq)[word_freq <= threshold]] <- ""
  x[x != ""] # Remove empty elements
})



# Save the cleaned data
my_df <- data.frame(filtered_text)


# write data frame to Excel file
write.xlsx(my_df, "updated3.xlsx", sheetName = "Sheet1", rowNames = FALSE)

#write_xlsx(filtered_text , "new.xlsx")
writexl::write_xlsx(my_df, "data.xlsx")
```