Information retrieval Final project:

Selected Query : "fairness measurement"

The project consists of several tasks in the field of natural language processing and in particular tasks towards an information retrieval system construction.

Language modelling: → We did the implementation of the language modelling in python

1. Tokenizing the text: → for that task we used the NLTK python library. nltk.tokenize package
   - Documentation in the following link → https://www.nltk.org/api/nltk.tokenize.html

2 Perform a set of linguistic operations (it is suggested to keep separate files with results – for analysis later on)

   - Remove stop words (a list is in the joint folder):

```
import nltk
from nltk.corpus import stopwords
print(stopwords.words('english'))
```

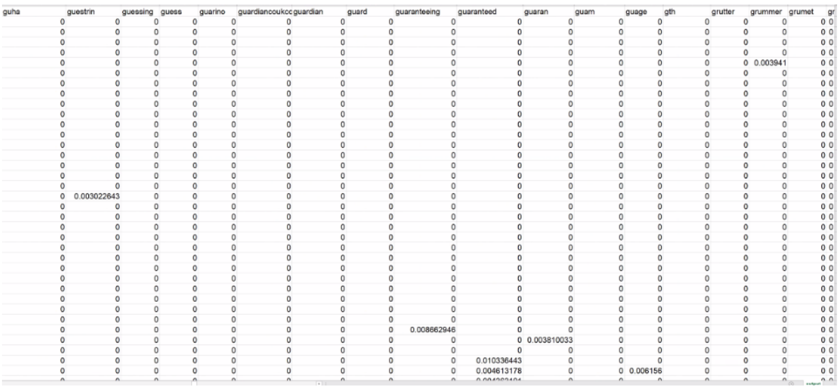
   - Perform Case Folding

```
# Loop over the filenames and read the contents of each file
docs = []
for filename in filenames:
    with open(os.path.join(doc_dir, filename), "r", encoding="ISO-8859-1") as f:
        # Read the contents of the file
        text = f.read()
        # Remove punctuation and digits
        text = text.translate(str.maketrans('', '', string.punctuation + string.digits))
        # Lowercase the text
        text = text.lower()
```

   - Perform Stemming, use porter (should be easy to find implementations)

```
# Define a stemmer and stem the tokens
stemmer = PorterStemmer()
tokens = [stemmer.stem(token) for token in tokens]
```

   - Create a new language model for your collection after every step in 2 above



guha	guestrin	guessing	guess	guarino	guardiancoukcc	guardian	guard	guaranteeing	guaranteed	guaran	guam	guage	gth	grutter	grummer	grumet	gr
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.003941	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.003022643	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.008662946	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0.003810033	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.010336443	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.004613178	0	0	0.006156	0	0	0	0	0

- Compare and discuss the difference → pros and cons.

Comparison of different language models constructed by the collection:

* Building model with **only remove stop words and lemmatization**: {remove_stop_words_lemma.csv}

General data:

- shape: 50 x 16,713 → means we have 16,713 tokens and 50 docs

In the context of sentiment analysis, removing stop words can be problematic if context is affected.

For example suppose your stop word corpus includes 'not', which is a negation that can alter the valence of the passage.

So you have to be cautious of exactly what is being dropped and what consequences it can have.

* Building model with **lower case and stop words removal**: {lower_case_stop_words_removal.csv}

General data:

- shape: 50 x 16,181 → means we have 16,181 tokens and 50 docs

The simplest idea is to make sure that our LM is not big with no need. We have no benefit of saving 'ferari' and 'Ferari'. Removing to lower case can be helpful to keep our LM smaller.

* Building model with **stop words & lower case & stemming**: {stop_words_lower_case_stemming_.csv}

General data:

- shape: 50 x 12,714 → means we have 12,714 tokens and 50 docs

Using the combination of those filtering tactics cas help reduce the size of the LM.

stemming the docs is also an operation that can be helpful as well.

Stemming is a straightforward normalization technique, most often implemented as a series of rules that are progressively applied to a word to produce a normalized form**.**

* Building model with **only case folding**: {case_folding.csv}

General data:

- shape: 50 x 17,600 → means we have 17,600 tokens and 50 docs

Over- all analysis:

- We can see that when we add more filtering we get a smaller language model.
  advantages: faster retrieval system.
  disadvantages: we might lose important information and context between words.

Text classification

1. Select randomly 50 documents from other directories (different ones)
2. Label the documents – belong to your category or not
3. Perform linguistic operations
   a. Tokenize
   b. Remove stop words

➔ THIS PART IS SIMILAR TO THE FIRST TAKS OF PRE- PROCESSING.

4. Use a machine learning library that is freely available (WEKA[1] for instance or SCIKIT[2])
    a. Select 4 classifiers of your choice →
    {multinomial naïve bayes, logistic regression, random forest and decision tree }
    b. Perform 10-fold cross validation – train with 90% of the documents and test with the rest – WEKA does it for you
5. Report and discuss the results, look at several erroneously categorized documents and explain why it happened

Discussion on the results:

- **<u>Important:</u> When training dataset is 90% we got accuracy of 1.0, hence why we decided to check accuracy for 20,30,50 precent.**

  If we perform the training with 30% of the data:

```
MultinomialNB:
Accuracy: 0.70
Precision: 1.00
Recall: 0.48
F1 Score: 0.65

LogisticRegression:
Accuracy: 0.83
Precision: 1.00
Recall: 0.70
F1 Score: 0.83

DecisionTree:
Accuracy: 0.96
Precision: 0.96
Recall: 0.98
F1 Score: 0.97

RandomForest:
Accuracy: 0.97
Precision: 0.96
Recall: 1.00
F1 Score: 0.98
```

[2] https://scikit-learn.org/stable/index.html

If we perform the training with 20% of the data:

```
MultinomialNB:
Accuracy: 0.72
Precision: 1.00
Recall: 0.50
F1 Score: 0.67

LogisticRegression:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00

DecisionTree:
Accuracy: 0.87
Precision: 0.81
Recall: 1.00
F1 Score: 0.90

RandomForest:
Accuracy: 0.95
Precision: 0.92
Recall: 1.00
F1 Score: 0.96
```

If we perform the training with 50% of the data:

```
MultinomialNB:
Accuracy: 0.89
Precision: 1.00
Recall: 0.81
F1 Score: 0.90

LogisticRegression:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00

DecisionTree:
Accuracy: 0.94
Precision: 0.94
Recall: 0.97
F1 Score: 0.95

RandomForest:
Accuracy: 0.98
Precision: 0.97
Recall: 1.00
F1 Score: 0.98
```

Classification analysis:

- We have 50 relevant and 50 non relevant docs.

  the 50 non relevant docs were taken from different queries in google scholar.

  This queries are different from the relevant docs query so we initially get different docs by default.
- That might explain why the models are getting high accuracy even with little data for training. The relevant and non relevant docs might be very different and easy to divide.

We also performed analysis for our model's false predictions. (at training on 20% from the data)

As seen below:

```
False Positive Records:
     aa  aaai  aaaiacm  aadirupa  aai  aamc  aaron   ab  ababneh  abadi  ...  \
80  0.0   0.0      0.0       0.0  0.0   0.0    0.0  0.0      0.0    0.0  ...
76  0.0   0.0      0.0       0.0  0.0   0.0    0.0  0.0      0.0    0.0  ...
65  0.0   0.0      0.0       0.0  0.0   0.0    0.0  0.0      0.0    0.0  ...

    ï³wealth  ï³dt  ï³kbì  ï¼tj   ïµ  ïµi  ïµidtï  ïµijt  ïµisdï  ïµì
80       0.0   0.0    0.0   0.0  0.0  0.0     0.0    0.0     0.0  0.0
76       0.0   0.0    0.0   0.0  0.0  0.0     0.0    0.0     0.0  0.0
65       0.0   0.0    0.0   0.0  0.0  0.0     0.0    0.0     0.0  0.0

[3 rows x 31423 columns]
False Negative Records:
Empty DataFrame
Columns: [aa, aaai, aaaiacm, aadirupa, aai, aamc, aaron, ab, ababneh, abadi, abandon, abat, abbi, abbrevi, abdel, abdollahpour
i, abduct, abeb, abedi, abil, abilityâ, abingdon, abl, abnorm, abort, abovement, aboveâ, abridg, abroca, abrooktechnionacil, ab
senc, absent, absolut, absolv, abstract, abstractâ, abstrus, abund, abus, abusitta, ac, academ, academi, academia, academica, a
cc, accent, accept, access, accesscontrol, accessâ, accid, accident, accom, accommod, accommoda, accommodationsmodif, accommoda
tionsâ, accommodationâ, accompani, accomplish, accomplishmentâ, accord, accordingli, account, accp, accpâ, accra, accru, accrua
l, accu, accumu, accumul, accur, accuraci, accuracyfair, accus, accusatori, accustom, ace, acheiv, achenbach, achiev, achieveme
ntbas, achievementdomin, achievementlevel, achievementori, achievementâ, achin, ackerman, acknowledg, acl, acm, acmiee, acmsia
m, acp, acpca, acquiesc, acquir, acquisit, ...]
Index: []

[0 rows x 31423 columns]
```

We now can examine using the doc_id column why those docs mis- classified.

Text clustering

1. Take documents from three additional directories – according to the instructions in the google sheet (you will get the numbers of directories to take documents from).

➔ We constructed a full language model for those 4 directories of docs returned by 4 different queries.
so if we apply k-means with k = 4 we should get some kind of separation between points in high dimensions.

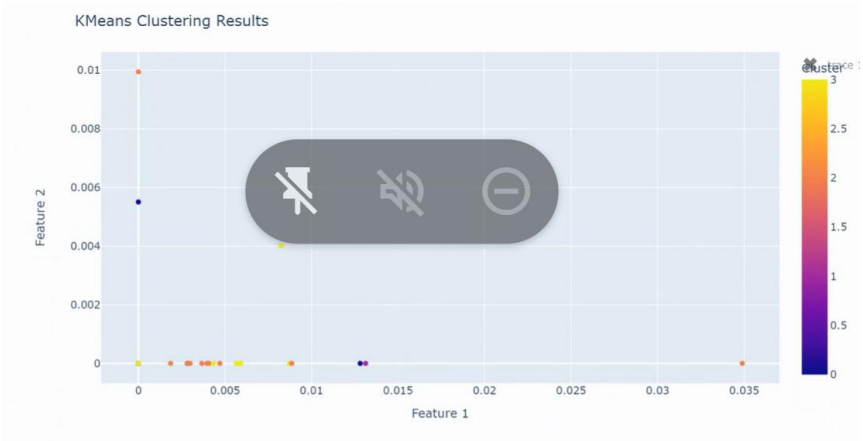
2. Use a machine learning library that is freely available (WEKA for instance or SCIKIT)
   a. Perform clustering using K-means and 4 clusters
   b. Analyze the results and Explain errors

Over all analysis for k-means with k = 4:

* We suggest the following explanation.
For very bad separation between points in high dim we believe that the queries were pretty similar to each other so the IR system returned docs that might be similar.

If we got good separation then we believe that the queries used in the IR system are very different so the docs returned are significantly different then we can see it when plotting the different clusters.

We used dim reduction using PCA to plot the data in a visual good manner.



We plotted each cluster with a different color.

We can see that the yellow cluster is like the orange cluster so it’s harder to decide and find a good separation.