*Information Retrieval*

# ASSIGNMENT-1 REPORT

Neev Swarnakar (2020390)

## Q1: Data Preprocessing

In the submitted code, I have focused my approach on text preprocessing to enhance the quality and consistency of textual data. My primary steps started with converting text to lowercase, tokenising using regular expressions, removing stopwords, eliminating punctuation, and discarding empty tokens.

**Methodology-**

1. Lowercasing and Tokenization:
    - Converts text to lowercase for uniformity.
    - Utilises regular expressions for tokenisation, extracting meaningful words while excluding non-alphanumeric characters.

2. Stopword Removal and Punctuation Handling:
    - Removes stopwords and filters out punctuation, promoting cleaner and more meaningful tokens.

3. Folder Creation and File Processing:
    - Creates a folder ("all_PreprocessedTextFiles") to store preprocessed files.
    - Processes text files from the "all_TextFiles" dataset, saving preprocessed versions in the designated folder.

**Assumption-**

1. Dataset Structure:
    - Assumes a collection of text files in the "all_TextFiles" folder.
    - Excludes the stopwords file ("stopwords.txt") from preprocessing.

2. File Encoding:
    - Assumes input text files are encoded in UTF-8 format.

3. Print Frequency:
    - Prints original and preprocessed content samples for every 50 files.

**Result-** All 999 .txt files in the all_TextFiles folder were preprocessed and put in the all_PreprocessedTextFiles folder, and 5 of these were printed for verification, which is as follows->

```
PS C:\Users\Neev vero\OneDrive\Desktop\semesters\semester-8\IR\Assignment-1> python q1.py
Before preprocessing - file1.txt:
Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge and want the most out of your springs than
these are the way to go.

After preprocessing - file1.txt:
loving vintage springs vintage strat tension stability floating bridge springs

Before preprocessing - file10.txt:
Awesome stand!

Tip: The bottom part that supports the guitar had a weird angle when arrived, making the guitar slide back, becoming almost 100% on a vertical.
To solve this, I assembled the product and the put a some pressure on the support frame, making it bend a little. Now my guitar sits perfectly. Check photos!

After preprocessing - file10.txt:
awesome stand tip bottom supports guitar weird angle arrived guitar slide becoming 100 vertical solve assembled product pressure support frame bend little guitar sits perf
ectly check photos

Before preprocessing - file100.txt:
This amp is the real deal.  Great crunch and gain tones and with some tweaking, not half bad clean"ish" tones.  I've played this through the two 8" Orange cabs (had to get
 those too as they were just TOO cool ((and cute)) and not crazy money) and the sound is very pleasing and revealing for a practice amp.  I primarily play it through my Bl
ackstar stack that I've fitted with Celestion V30s... Wow...there it is~!!!  You would never know this thing was such a tone monster... Even with just a few knobs it's eas
y to get lost for hours playing this thing.  My favorite match is with my Chapman ML-1 Hotrod...which only has a volume "tone" control (EVH fans get this).  Not a lot of m
ucking around with too many knobs or too many options on either the guitar or this amp... Just tone up and go.  I see the Micro Dark just came out...that's probably next~!
  Higher gain, buffered effects loop and speaker emu at the headphone out for recording direct (if that's your thing).

After preprocessing - file100.txt:
amp real deal crunch gain tones tweaking half bad clean ish tones ve played 8 orange cabs cool cute crazy money sound pleasing revealing practice amp primarily play blacks
tar stack ve fitted celestion v30s wow tone monster knobs easy lost hours playing favorite match chapman ml 1 hotrod volume tone control evh fans lot mucking knobs options
 guitar amp tone micro dark probably gain buffered effects loop speaker emu headphone recording direct
```

```
Before preprocessing - file101.txt:
You can do a lot with this mixer. its great for podcasting. has 4 outputs that can be used to monitor, record, cue audio...The mute to 3/4 figure on every channel is fanta
stic and the three source switch to headphone/control room is a must for podcasting. Also has aux return inputs that can be used as extra stereo inputs and be volumed by t
he aux return knobs.

Only thing I didn't like about this mixer is the XLR outputs in back that require adaptors to use with RCA or 1/4 plugs. get the adaptors with it

After preprocessing - file101.txt:
lot mixer podcasting 4 outputs monitor record cue audio mute 3 4 figure channel fantastic source switch headphone control podcasting aux return inputs extra stereo inputs
volumed aux return knobs didn mixer xlr outputs require adaptors rca 1 4 plugs adaptors

Before preprocessing - file102.txt:
<div id="video-block-R2VOQ5CBZHFCKL" class="a-section a-spacing-small a-spacing-top-mini video-block"></div><input type="hidden" name="" value="https://images-na.ssl-image
s-amazon.com/images/I/E1%2B6MhK2MfS.mp4" class="video-url"><input type="hidden" name="" value="https://images-na.ssl-images-amazon.com/images/I/21-Jk5lxqsS.png" class="vid
eo-slate-img-url">&amp;nbsp;This mic is a BOSS and a lot better than just about any other mic I've seen or used out-of-the-box for voice over. It sounds great even before proc
essing, and with some compression and EQ, it sounds fantastic. It rejects a ton of background noise and sounds amazing.

It runs very HOT! So you'll want clean pre-amping as to get a clean signal, but this is an amazing mic for the price.

After preprocessing - file102.txt:
div id video block r2voq5cbzhfckl class section spacing spacing top mini video block div input type hidden name value https images na ssl images amazon com images e1 2b6mh
k2mfs mp4 class video url input type hidden name value https images na ssl images amazon com images 21 jk5lxqss png class video slate img url nbsp mic boss lot mic ve seen
 box voice sounds processing compression eq sounds fantastic rejects ton background noise sounds amazing runs hot ll clean pre amping clean signal amazing mic price

Before preprocessing - file103.txt:
This is a practical and versatile instrument tuner. It includes a metronome and tone generator and a variety of settings. The clamp attaches easily to an instrument and th
e tuner can rotate to the angle that is convenient for seeing it. The power and function buttons are easy to use. This does have a few limitations, such as the frequency r
ange limit of 435-445 hz. The meter shows you if you are flat or sharp or right on the note, and this is a precise tuner - you won't get the green light if you are slightl
y off. I have tried multiple tuners and this one is good, with the extra features that make it a great value and go to accessory for musicians. I received a sample at no c
ost for my honest, unbiased review.

After preprocessing - file103.txt:
practical versatile instrument tuner includes metronome tone generator variety settings clamp attaches easily instrument tuner rotate angle convenient seeing power functio
n buttons easy limitations frequency range limit 435 445 hz meter flat sharp note precise tuner green light slightly tried multiple tuners extra features value accessory m
usicians received sample cost honest unbiased review
```

## Q2: Unigram Inverted Index and Boolean Queries

In the submitted code, a unigram inverted index is created from the 999 preprocessed .txt files in the all_PreprocessedTextFiles folder and implements Boolean queries. The methodology involves reading stopwords, creating an inverted index, and executing various Boolean operations. The code utilises the NLTK library for tokenisation, stemming, and stopwords. The primary goal is to facilitate efficient retrieval of documents based on user-specified queries.

**Methodology-**

1. Stopword Handling:
    - Reads stopwords from the "stopwords.txt" file, essential for subsequent processing steps.

2. Inverted Index Creation:
    - Iterates through the preprocessed dataset, creating an inverted index.
    - Utilises NLTK's tokenisation, lowercasing, and stopwords removal.
    - Tracks document frequency for each term in the inverted index.

3. Boolean Query Execution:
    - Executes AND, OR, AND NOT, and OR NOT queries on the inverted index.
    - Outputs the number of retrieved documents and their names based on the input query and associated operators.

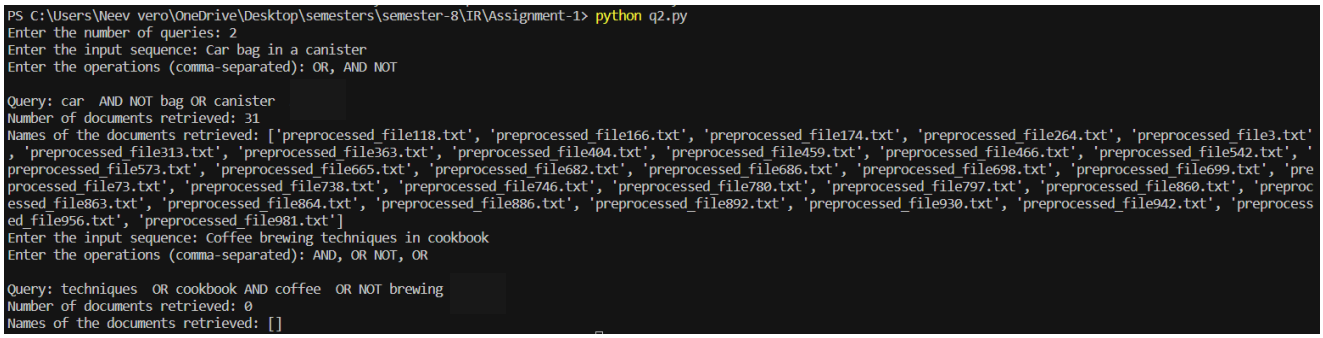
**Assumptions-**

1. Dataset Structure:
    - Assumes a preprocessed dataset in the "all_PreprocessedTextFiles" folder.

2. Stopwords File:
    - Assumes the existence of a stopwords file ("stopwords.txt") for filtering common words.

3. Query Format:
    - Expect user queries and operations to be entered in a specific format.

**Result-** A unigram inverted index was created using the all_PreprocessedTextFiles folder, which was further used to support various boolean operations needed to interpret the combinations of input query and operators. An example result is->



# Q3: Positional Index and Phrase Queries

In the submitted code, a positional index was created using the preprocessed dataset and facilitated phrase queries. It employs NLTK for tokenisation and stopword removal, saving the resulting positional index using Python's pickle module. The key steps involve preprocessing, index creation, and user-driven phrase queries for document retrieval.

**Methodology-**

1. Preprocessing:
    - It uses NLTK for tokenisation, lowercasing, and stopword removal.
    - Generates a list of document-specific tokens considering only relevant characters.

2. Positional Index Creation:
    - Constructs a positional index mapping terms to their positions in documents.
    - Leverages NLTK's stopwords list for filtering common words.

3. Storage of Positional Index:
    - Saves the positional index using Python's pickle module for future use.

4. Phrase Query Execution:
    - Takes user input for phrase queries and executes them based on the positional index.
    - Outputs the number of retrieved documents and their names.

**Assumptions-**

1. Dataset Structure:
    - Assumes a preprocessed dataset in the "all_PreprocessedTextFiles" folder.

2. Stopwords Handling:
    - Utilises NLTK's stopwords list for standard English stopwords removal.

3. Query Format:
    - Expect user queries to be entered in a specific format.

**Result-** A positional index was created using the Preprocessed dataset folder, facilitating efficient retrieval of documents based on phrase queries. This positional index was utilised to support various boolean operations required to interpret combinations of input queries and operators. An example result is->