Authorship Analysis using TextAnalysis.jl

In this notebook we will implement an authorship analysis technique that uses n-gram frequencies for identifying authors. This technique is described in **Grieve 2018** The technique counts token and character n-grams and compares them to a reference corpus. It counts the number of overlapping n-grams in the reference corpus and identifies the author based on which reference corpus contains more of the n-grams from the text.

First off, the necessary imports are made.

```
begin
    using TextAnalysis
    using CSV
    using DataFrames
end
```

Our data will be from **Spooky Author Identification** Kaggle contest. The data comes in the form of a CSV file which contains sentences attributed to one of three authors. The data was manualy cleaned to remove rows that were corropted as well as making sure that each author had approximately equal word counts.

The data is read into a dataframe after which we iterate over the df and turn each sentence into a StringDocument which is then accumalted into a Corpus for each author.

```
referenceData = CSV.read("train.csv", DataFrame);
```

```
A Corpus with 5544 documents:
 * 5544 StringDocument's
 * 0 FileDocument's
 * 0 TokenDocument's
 * 0 NGramDocument's

Corpus's lexicon contains 0 tokens
Corpus's index contains 0 tokens
```

```
begin
    MWS = GenericDocument[]
    EAP = GenericDocument[]
    HPL = GenericDocument[]

    for line in eachrow(referenceData)
        if line.author == "MWS"
            push!(MWS, StringDocument(line.text))
        elseif line.author == "EAP"
```

```
            push!(EAP, StringDocument(line.text))
        else
            push!(HPL, StringDocument(line.text))
        end
    end

    MWS_corpus = Corpus(MWS)
    EAP_corpus = Corpus(EAP)
    HPL_corpus = Corpus(HPL);
end
```

The following code snippet allows one to retrieve the word count of a corpus. The word counts are as follows:

- MWS_corpus: 147914
- EAP_corpus: 147911
- HPL_corpus: 147916

```
begin
    wordCount = 0
    for doc in HPL_corpus
        global wordCount += length(ngrams(doc, 1))
    end
end
```

147916

```
wordCount
```

Now that we have our corpora it is time to extract the ngrams that are needed for the analysis. The original paper used both character ngrams and token ngrams. Due to TextAnalysis.jl's functionality I will be restricting this analysis to only using token ngrams.

```
begin
    MWS_ngrams = []
    HPL_ngrams = []
    EAP_ngrams = []
    for i in 1:4
        push!(MWS_ngrams, map(x -> ngrams(x,i), MWS_corpus))
        push!(HPL_ngrams, map(x -> ngrams(x,i), HPL_corpus))
        push!(EAP_ngrams, map(x -> ngrams(x,i), EAP_corpus))
    end
end
```

We also retrieve the token ngrams from the files we want to analyze and classify.

```
begin
    # first we load our files from memory
    text1 = FileDocument("./MWS.txt")
    text2 = FileDocument("./HPL.txt")
    text3 = FileDocument("./EAP.txt")

    # this is the same as retrieveing ngrams for the reference corpora
    text1_ngrams = []
    text2_ngrams = []
```

```
    text3_ngrams = []
    for i in 1:4
        push!(text1_ngrams, ngrams(text1, i))
        push!(text2_ngrams, ngrams(text2, i))
        push!(text3_ngrams, ngrams(text3, i))
    end
end
```

Finally we will a script that calculates the intersection of the ngrams from the text and the three reference corpora. To try different files edit the second for loop.

```
begin
    MWS_count = 0
    HPL_count = 0
    EAP_count = 0
    for i in 1:4
        for j in text1_ngrams[i]
            for k in MWS_ngrams[i]
                if haskey(k, j[1])
                    global MWS_count += 1
                    break
                end
            end
            for k in HPL_ngrams[i]
                if haskey(k, j[1])
                    global HPL_count += 1
                    break
                end
            end
            for k in EAP_ngrams[i]
                if haskey(k, j[1])
                    global EAP_count += 1
                    break
                end
            end
        end
    end
end
```

The following block will output the author as identified by this script. The correct authors should be as follows:

- text1 = MWS
- text2 = HPL
- text3 = EAP

```
begin
    if MWS_count > HPL_count && MWS_count > EAP_count
        "MWS"
    elseif HPL_count > MWS_count && HPL_count > EAP_count
        "HPL"
    else
        "EAP"
    end
end
```

Conclusion

While this is a very watered down analysis of what the paper presents it is a good starting point and shows that this technique is viable as it correctly predicts the authors of our random test data. One way to improve the current analysis is to add character ngrams as the paper does.

All in all this was a fun project to learn more about Julia and its NLP cabalities. This project only scratches the surface of what Julia has to offer given that the TextAnalysis.jl package offers things such as part-of-speech(POS) tagging, tf-idf measures, a sentiment analysis model, and much more.