Língua Natural - 1º Mini-projeto

Grupo 13 Sofia Aparício 81105 Rodrigo Lousada 81115
Rogério Cabaço 81470

November 4, 2017

Exercicio 1- Abordagem

O primeiro exercício do projeto tinha como objetivo final devolver um sumário de um documento dado, por outras palavras as frases mais relevantes desse mesmo documento, correndo apenas um documento txt e comparando cada frase que constitui o documento pelo documento inteiro. O documento escolhido para testar este exercício é um ficheiro que se encontra na biblioteca TeMario com o nome de: "ce94jl10-a.txt"

Sendo a primeira exigência a existência de um TF score normalizado e de um IDF score logaritimicamente escalado, tal como discutido nas aulas e presente nos slides, o projecto foi iniciando com a análise das bibliotecas scikit-learn e nltk. Após uma análise intensiva, reparou-se que a biblioteca scikit-learn embora tenha uma classe TfidfVectorizer que facilita o trabalho na geração dos Vector Space Models, não cumpria com essa exigência. Para solucionar este problema o grupo tencionava:
- tirar o máximo proveito das qualidades destas bibliotecas - não exigir ao corpo docente que descarregasse uma versão nova do scikit-learn que tivesse apenas esta ligeiras alterações do grupo no código

Optou-se então por criar as classes TfidfVectorizer_2 e TfidfTransformer_2 que utilizam as propriedades de herança nas classes TfidfVectorizer e TfidfTransformer, respetivamente, de forma a apenas rescrever as funções necessárias. As alterações foram efetuadas de forma a que o valor do TF-IDF não fosse alisado e respeitasse a fórmula dada nas aulas teóricas.

Aplicando essa formula é possível criar um vectorizer para, posteriormente, fazer fit com as frases do documento e transform às frases do documento e ao documento inteiro. Após a operação de cosine_similarity entre os VSMs é criado um dicionário para emparelhar o índice de cada frase com a sua similaridade. Acedendo ao dicionário é possível calcular quais são as três (valor manipulável) melhores frases, sendo estas impressas no terminal quando corremos o programa.

Ao chamar a função exercise_1_main poderemos escolher o ficheiro a analisar e o número de frases que queremos no "top".

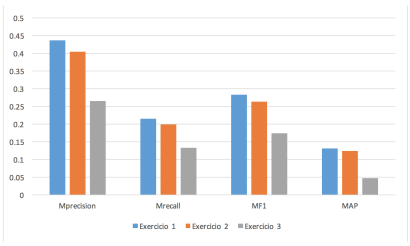
Exercicio 2 - Abordagem

Este exercício exigia não só que o exercício anterior pudesse processar documentos Portugueses (algo corrigido facilmente no exercise_1.py) mas que a comparação de cada frase do documento não fosse só feita relativamente ao documento mas à coleção inteira. O grupo optou por aproveitar funções criadas no exercício anterior onde os procedimentos fossem os mesmos.

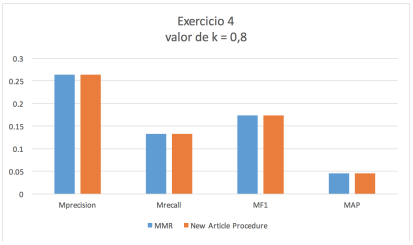
Uma vez que teríamos de correr a coleção de documentos toda, foi feito um loop para comparar cada documento com a coleção de documentos, através da função similarity. Nesta função vai ser feito o fit do ficheiro do qual se pretende extrair o sumário e em seguida a transformação tanto deste ficheiro como uma lista que tem apenas uma string com todos os documentos. De forma a comparar os resultados desta abordagem com os resultados do exercício um, cada ficheiro é corrido quer pela abordagem do exercício 1 que pela abordagem

do exercício 2, sendo calculados os valores de precisão, recall, F1 e AP com o conjunto de frases relevantes a serem extraídas dos ficheiros disponibilizando os sumários ideias.

Tudo isto, com vista a obter as médias das medidas acima descritas. Pelos seguintes gráficos podemos observar que a abordagem do exercício um apresenta uma performance ligeiramente melhor, algo que se ampliaria caso os ficheiros referissem tópicos muito diferentes:



(a) Comparação entre ex 1, 2 e 3



(b) Estatistica exercicio 4

Figure 1: The same cup of coffee. Two times.

Exercicio 3 - Abordagem

No exercício 3 o grupo optou por preprocessar os dados retirando as stopwords, o que demonstra um ligeiro improvement na performance dos algoritmos referidos.

Na alínea 1 optamos por encontrar as tags morfologicas das palavras e retirar as stopwords, com o intuito de ótimizar o resumo.

Na alínea 2 a solução passou por criar novas classes BM25Vectorizer e BM25Transformer que, à semelhança do exercício 1, herdam das classes TfidfVectorizer e TfidfTransformer, aplicando as seguintes formulas no cálculo do IDF e do score final de cada termo, na criação dos Vector Space Models.

Ao correr o programa podemos ver uma comparação entre as 3 abordagens, revelando esta um desempenho bem pior, quando comparados com os sumários ideais.

Exercicio 4 - Abordagem

O último exercício acabou por se revelar o mais simples a nível de programação, não exigindo um estudo adicional de novas funções e bibliotecas, sendo apenas a aplicação direta da fórmula descrita no enunciado.
Foi ainda criado uma função main do exercício 4 de forma a haver uma fácil edição do número de frases a serem comparadas, e do valor de lambda, onde poderemos observar uma ligeira alteração na performance quando alterado. Ao correr, podemos ainda comparar as estatísticas deste com o simples procedimento onde assumimos que as primeiras 5 frases são as mais relevantes.