IST 2016
**Processamento e Recuperação de Informação**
Project Report - Part 1 - Group 4

## 1 - Simple approach based on TF-IDF

Começamos por ler um english textual document (parte de um capítulo de Alice In Wonderland) onde filtramos as palavras. Tiramos os símbolos de pontuação, menos o apóstrofo (para manter palavras como “don’t” intactas), e de seguida as stopwords. A seguir calculamos bigramas através dessas palavras e juntamos às mesmas. Assim ficamos com uma lista de candidatos com todas as palavras e bigramas.

Depois vamos buscar ao 20newsgroups o train.data e juntamos a esse conjunto os candidatos recolhidos anteriormente. Desta maneira, ao calcular o IDF, todos os termos do documento que fornecemos existem no vocabulário, permitindo que haja smoothing ao IDF.

De seguida calculamos o TF do english textual document e calculamos o TF-IDF. Em seguida, ordenamos os resultados por ordem crescente e selecionamos os 5 valores maiores.

## 2 - Evaluating the simple approach

O primeiro passo é a leitura dos 30 documentos do FAO30 e das suas keys pelas várias equipas, juntando-as para cada documento (utilizando Python Sets para excluir repetições).
De seguida, utilizando um TFIDFVectorizer, obtemos os TF-IDF para cada par documento-termo.
Para cada documento, essas pontuações por termo são usadas para reordenar crescentemente uma lista de índices, que correspondem às posições de cada termo no vocabulário do Vectorizer. Depois podemos obter os cinco termos com maior índice, por documento, pois também terão os valores TF-IDF mais altos.

Para as medidas de desempenho, para cada documento, as listas de termos relevantes e as de termos que o programa indica como relevantes são comparadas.
A mean-average-precision é calculada em 5 iterações, correspondendo aos 5 termos obtidos para cada documento.
Comparando com as classificações feitas pelas equipas, os nossos resultados são fracos, com precision, recall, F1 e mean-average-precision baixos. Isto deve-se a, por vezes, entre os termos relevantes indicados pelas equipas e pelo programa haver somente um resultado em comum. Esta “simple approach” não é, portanto, a indicada.

## 3 - Improving the simple approach

O primeiro passo neste exercício foi reescrever a gramática do enunciado de modo a ser legível pelo pacote “re”. Uma das alterações que tivemos de fazer foi acrescentar “$” no fim da gramática de modo a que, quando um termo fizesse match à expressão, fosse rejeitasse

caso a seguir houvesse mais algum símbolo. Acrescentámos também “[A-Z]*” no lugar do “.*” para permitir que fosse possível fazer match com qualquer letra dentro da tag “<NN>”, como por exemplo “<NNP>” para nomes próprios.

Após a reescrita da gramática é-nos possível filtrar os n-grams que definimos como candidatos. De seguida, calculamos as frequências dos termos por documento usando um CountVectorizer, usando um tokenizer custom[1] que faz a verificação pela expressão regular.

Em seguida, calculamos o IDF segundo a fórmula do enunciado, depois usado na fórmula do BM25. O seu parâmetro f(t,D) é obtido na matriz esparsa criada pelo Vectorizer.
O ordenamento dos candidatos é feito como nos exercícios anteriores, retornando também 5 resultados por documento.

**4 - A more sophisticated approach**

Este programa utiliza parte da lógica presente no que foi feito para a questão 3. Utiliza-se um CountVectorizer com um tokenizer “custom” que, desta vez, não verifica os termos segundo uma expressão regular mas de novo separa as palavras em bigramas e trigramas e mantém mais algumas estatísticas sobre os termos eos documentos. Por exemplo, é calculada a soma das frequências de cada termo em todos os documentos, para ser utilizado no cálculo das probabilidades.

Para o cálculo da informativeness, em que se usam Language Models apenas com unigramas (para P(W) e Q(W)), há independência entre as palavras dentro de termos. Por isso, a probabilidade de ocorrência de um termo é apenas o produto das probabilidades das palavras que o constituem. Consideramos unigramas, bigramas e trigramas como possíveis termos e limitamos o cálculo da probabilidade de ocorrência de sequências a três produtos, ou seja, depender apenas de um número reduzido de palavras anteriores (N = 3).

Para o cálculo da phraseness, utilizamos uma aproximação em que reduzimos o número de cálculos, “cortando” numeradores e denominadores entre as parcelas do produtório usado no cálculo da probabilidade. Este produtório torna-se apenas uma divisão, da seguinte forma:

$$P(W) \approx \prod_{i=1}^{m} \frac{count(w_{i-(N-1)},\dots,w_i)}{count(w_{i-(N-1)},\dots,w_i)} = \frac{count(w_1)}{count(all)} \frac{count(w_1,w_2)}{count(w_1)} \frac{count(w_1,w_2,w_3)}{count(w_1,w_2)} = \frac{count(W)}{count(all)}$$

Nos cálculos envolvendo probabilidades em foreground e background documents, as contagens utilizadas são feitas apenas no documento em avaliação ou em todo o corpus (através de valores acumulados guardados nas estatísticas), respectivamente.

[1] http://scikit-learn.org/stable/modules/feature_extraction.html#customizing-the-vectorizer-classes

A adição de informativeness e phraseness para cada par termo-documento é guardada e ordenada como descrito nas questões anteriores, produzindo, mais uma vez, cinco termos mais relevantes para cada documento.