Relatório MP2 de Língua Natural

Grupo 11: André Soares 82054, Duarte Cabral 82059

1. Introdução

Para a subárea de informática que é Processamento de Língua Natural (**PLN**), um dos seus principais focos é o de **Interpretação de Língua Natural**, que lida especificamente com o modo como uma máquina/sistema informático lê e interpreta texto [1].

Para o segundo mini-projeto da cadeira de Língua Natural (**LN**), o problema que nos foi proposto foi um inserido neste foco de **PLN**. Especificamente, o projeto envolve a concepção dum *script* que, ao receber um conjunto de questões/ordens relacionadas com a área de cinema, mandava como *output* um conjunto de ***tags*** (entre 16 possíveis) que dizia qual o tipo de pergunta sobre cinema que foi feita. Por exemplo, para a questão "*What are the most relevant actors in the movie Pulp Fiction?*", o *script* desenvolvido iria retornar a ***tag*** "*actor_name"*. Para isto, foi nos entregue um conjunto de ficheiros para nos auxiliar na construção da nossa solução: o ficheiro *QuestoesConhecidas.txt* contém um conjunto de perguntas com as ***tags*** corretas respetivas, i.e., serve do corpus para o sistema; o ficheiro *NovasQuestoes.txt* contém algumas questões sem ***tag*** que podemos usar para executar o sistema; por último, no documento *NovasQuestoesResultados.txt* encontram-se as ***tags*** corretas respetivas a cada pergunta do segundo ficheiro mencionado, podendo ser utilizado para avaliar a solução.

Durante o resto deste relatório, iremos detalhar o modo como abordamos o problema proposto em termos de soluções possíveis, o porquê de termos escolhido a nossa solução, como esta funciona e se compara em termos de performance a duas outras métricas, estas também bastante conhecidas na área de **PLN**.

2. Proposta de Solução

Tendo em conta o problema descrito previamente e a sucessiva pesquisa de métricas de interpretação de Língua Natural, foi-nos possível reduzir a nossa lista de possíveis métricas a usar a apenas 3:

- Distâncias entre palavras;
- Regras de associação de palavras;
- Documentos como Vetores;

Dentro destas possíveis métricas, conseguimos para o nosso contexto criar uma linha de comparação entre o uso de documentos, na métrica de documentos como vetores[1], com o nosso conceito de ***tags,*** sendo que no caso da métrica é explicitado que os documentos são utilizados como forma de entender e calcular o número de ocorrências de frase por documento e para o nosso caso qual o número de ocorrências de palavra por ***tag***. Como tal o uso da métrica de documentos como vetores aplicada ao nosso problema pareceu-nos o caminho mais indicado a tomar. O uso da distância de coseno é usualmente

[1] https://fenix.tecnico.ulisboa.pt/downloadFile/1407993358876766/DocumentsAsVectors.pdf

associada ao uso de documentos como vetores, mas sentimos que para a nossa possível solução esta não seria o tipo de cálculo mais indicado, visto que optámos por usar ***tags*** em vez de documentos, e portanto decidimos que seria benéfico criar novas regras de associação.

Sendo assim, a nossa proposta de solução ao problema irá passar por aplicar uma combinação dos conceitos provenientes das métricas de Documentos como Vetores (uma variação da técnica **term frequency–inverse document frequency (tf-idf)** [2]), e de Regras de Associação de Palavras, no que toca à atribuição final de ***tags***.

Primeiro, antes de desenvolvermos a nossa solução, foi necessário aplicar algumas técnicas de **Pré-Processamento** ao ficheiro que compõe o nosso corpus (e que será posteriormente utilizado com ficheiro de treino do nosso sistema), bem como ao ficheiro com as novas questões a ser classificadas. De uma forma mais específica, aplicamos ***lowercasing*** a todas as palavras, **removemos pontuação e plicas** das frases (?,!,.,:,,,-), transformamos **tabs em espaços** e posteriormente **removemos espaços extras**, aplicamos um processo de **lematização** como forma de remover ocorrências de palavras no plural (como alternativa a *stemming*). Devemos mencionar também que, por motivos de organização, ordenamos ainda o ficheiro de treino por **ordem alfabética** das ***tags***.

No que toca à nossa solução, efetuamos o **treino do nosso sistema** aplicando a técnica de **tf-idf**, mas para o nosso caso em particular, não realizamos uma classificação de frases por documentos mas sim uma **classificação de palavras por *tags***. Ou seja, o nosso objetivo passava por perceber o quão importante uma palavra é para cada uma das ***tags*** no nosso corpus de treino. Pegando no ficheiro de treino pré-processado, criámos um novo ficheiro com todas as palavras de cada frase (excluindo as ***tags***), separadas por linha, ordenadas por ordem alfabética e sem a ocorrência de palavras repetidas, que foi usado como forma de, palavra a palavra, **calcular e atribuir um valor para cada uma das *tags* existentes**. Este valor é o produto entre duas frequências, a frequência do termo (número de vezes que a palavra ocorre por ***tag***), que beneficia palavras que ocorram diversas vezes para a mesma ***tag*** e a frequência inversa do documento (se a palavra é comum ou rara para todas as ***tags***), que prejudica as palavras que ocorrem diversas vezes para diferentes t***ags***, mais formalmente o valor de **tf-idf**. Por exemplo, a palavra *actor* vai obter um valor **tf-idf** alto para a ***tag*** *actor_name*, visto que ocorre imensas vezes em frases com essa tag (tf alto) e ocorre muito pouco frequentemente em frases com outras ***tags*** (idf alto). Por outro lado, a palavra *which* obtém valores baixos, já que ocorre um elevado número de vezes para várias ***tags*** (tf alto e idf muito baixo). No final desta fase, obtemos uma tabela n x 16 (sendo n o número de palavras existentes no corpus usado, e 16 o número de ***tags*** existentes), em que cada entrada corresponde ao valor **tf-idf** para uma certa palavra e uma certa ***tag***.

Para classificar uma nova questão, segmentamo-la num *array* de palavras (sem repetições) e, para cada palavra, vamos à sua posição correspondente na tabela anteriormente mencionada e guardamos o valor máximo de **tf-idf** que lhe está associado num outro *array*, juntamente com o índice da ***tag*** correspondente. Se, porventura, a palavra for uma *substring* da ***tag***, o valor **td-idf** desta é incrementado com um bónus (p.ex., *character* está presente em *character_name*). Uma palavra que não esteja presente na base de dados terá como valor default 0 para o seu **tf-idf**, porque assumimos não ser relevante para a nossa abordagem. Cada palavra fica com a ***tag*** cujo valor **tf-idf** correspondente for maior. Por fim, tendo cada palavra uma sido classificada, decidimos que

a ***tag*** mais provável para a frase é aquela que está associada à palavra ao maior valor absoluto de **tf-idf**.

3. Resultados

De forma a conseguir obter uma melhor interpretação dos nossos resultados, optámos por delinear uma **baseline** a partir da qual poderíamos retirar conclusões comparando esta com os nossos resultados. Para tal implementámos métricas de distâncias entre palavras, mais especificamente, um caso com **Jaccard** [3] e outro caso com **Dice** [4].

Usámos um corpus *QuestoesConhecidas.txt* contendo 208 questões com **tags** anotadas. Nesse corpus existe pelo menos um conjunto de 3 perguntas para cada tag existente. O ficheiro *NovasQuestoes.txt* foi usado para teste da nossa solução e cálculo de **baseline**. No que toca aos resultados obtidos para a nossa baseline, verificámos que tanto a distância de **Jaccard** como de **Dice** apresentavam os mesmos resultados, como tal usaremos apenas uma nas comparações. Os resultados pretendidos para o tratamento de **tags** para o ficheiro de *NovasQuestoes.txt* foi-nos proporcionado no ficheiro *NovasQuestoesResultados.txt*, contendo 42 **tags** atribuídas às 42 perguntas correspondentes. Comparando os resultados da nossa **baseline** com estes, verificamos que das 42 **tags** atribuídas, 11 encontram-se erradas, ou seja apresenta uma **accuracy** de 73,8%. Em si não se trata de um mau resultado, mas sabíamos que seria possível fazer bastante melhor, até para um corpus reduzido como aquele que estávamos a utilizar.

No que toca à nossa solução, conseguimos obter resultados bastante próximos dos proporcionados para teste e bastante melhores quando comparados à nossa **baseline**. A nossa solução atribuiu 40 **tags** corretamente em 42, pelo que apresente uma accuracy de 95,2%. Vemos que não se tratam de resultados perfeitos, mas tendo em conta que o Corpus de treino não é muito extenso ou variado, um resultado na ordem dos 95% permite-nos mostrar que a nossa solução proposta é uma boa hipótese para a resolução do nosso problema. Acreditamos que o fator deve-se à atribuição de um bónus a palavras duma frase que estejam contidas no nome de uma **tag**, visto que numa boa quantidade de casos, consegue fazer quase um “mapeamento direto” entre a frase e uma **tag**.

4. Conclusões e Trabalho Futuro

Pegando nos resultados obtidos tanto para a nossa solução como os valores provenientes da baseline, conseguimos constatar que estamos perante uma boa solução para o nosso problema descrito. Nunca conseguindo resultados perfeitos, constatamos que a presença de um **Corpus** de treino mais variado poderia proporcionar-nos uma pequena melhoria nos resultados obtidos.

No que toca a trabalho futuro propomos:

- Adicionar um mecanismo de normalização ao pré-processamento, de forma a obter sinônimos de palavras para frases para a mesma **tag** que seriam bastante diferentes caso não houvesse este mecanismo implementado;

- Implementar no nosso sistema um mecanismo de aprendizagem que, ao receber um input novo, introduziria-o na base de dados existente e recalibraria os valores de decisão usados;
- Garantir um Corpus de treino mais denso e variado.

5. Bibliografia

1. Wikipedia contributors. (2018). Natural-language understanding. In Wikipedia, The Free Encyclopedia. Retrieved 16:20, November 3, 2018, from https://en.wikipedia.org/w/index.php?title=Natural-language_understanding&oldid=866987908
2. Wikipedia contributors. (2018). tf–idf. In Wikipedia, The Free Encyclopedia. Retrieved 16:47, November 3, 2018, from https://en.wikipedia.org/wiki/Tf%E2%80%93idf
3. Wikipedia contributors. (2018). Jaccard index. In Wikipedia, The Free Encyclopedia. Retrieved 17:53, November 5, 2018, from https://en.wikipedia.org/wiki/Jaccard_index
4. Wikipedia contributors. (2018). Sørensen–Dice coefficient. In Wikipedia, The Free Encyclopedia. Retrieved 17:54, November 5, 2018, from https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient