Língua Natural Projeto 1 – Grupo 40

Alexandre Guerra Nº86370 João Barata Nº86450
Paulo Dias Nº86492 Tiago Mesquita Nº86520

1. Introdução

O problema consiste em criar um agente conversacional, *retrieval-based*, que recebe como input dois ficheiros, o primeiro com FAQs do "Balcão do Empreendedor" (**KB.xml**), o e o segundo um ficheiro com perguntas (**test.txt**).

O objetivo deste agente é encontrar para cada uma destas perguntas, a do FAQ que seja mais análoga a esta, devendo no final retornar a lista de ID's correspondentes. (esta deve ser escrita num ficheiro com o nome **resultados.txt**). Caso não exista para uma das perguntas uma que seja suficientemente semelhante, o programa deve retornar 0 como ID para a pergunta.

O objetivo deste relatório é assim descrever os processos utilizados neste âmbito, assim como os raciocínios que justificaram a sua utilização.

2. Proposta

Foi decidido como estrutura para a FAQ, um dicionário para (listaPerguntasFAQ), com chave correspondente ao ID, cujos valores correspondem a uma lista com as perguntas a si associadas. Esta estrutura pareceu apropriada, pois segue a estrutura do próprio XML, facilitando assim o acesso à informação. O processo de extração desta, é feito através da função extractXML, que percorre o **KB.xml** e extrai deste os ID's e as respetivas perguntas. Para as perguntas do ficheiro **test.txt,** decidiu-se que uma lista seria suficiente para organizar as perguntas (listaPerguntasTXT), cuja extração foi feita a partir da função extractTXT, correspondente a função nativa do **python**, **File**.**readlines**().

Após feita a extração das perguntas, o programa avança para o pré-processamento destas (o pré-processamento é feito para as listas listaPerguntasFAQ e listaPerguntasTXT).

A escolha das experiências reflete parte do processo de aperfeiçoamento. ***Jaccard*** foi escolhido como método controlo, dada a sua simplicidade explícita, assim como a sua eficácia no contexto deste problema, pois as frases do *Corpora* são de reduzida dimensão, o que evita repetições de palavra (por frase), o que torna a utilização de **sets** (por palavra) aceitável, uma vez que há pouca perda de informação.

O pré-processamento de controlo foi definido pelas técnicas de pré-processamento: **lowercasing,** colocando-se todas as letras em minúsculas; **punctuation**, onde é removida toda a pontuação; **stopwords**, onde são removidas todas as palavras cuja presença é pouco relevante. Estes fatores foram escolhidos por causarem grande impacto nos sets gerados (a partir das questões), potencialmente prejudiciais para algoritmos de distância baseados em

sets, nomeadamente ***Jaccard*** e ***Dice.*** Estas técnicas são aplicadas nas funções: **preProc** (**lowercasing** e **punctuation**), iterando cada caráter e aplicando as respetivas substituições; **removeStopWords**, utilizando a lista obtida através do ficheiro **stopwords.txt**.

Stemming, caracterizado como o encurtamento de palavras através de remoção sufixos reduzindo-as a raízes comuns, foi decidido como variável de teste devido à sua eficácia extremamente contextual, pois aumenta as interseções entre sets, correndo, no entanto, o risco de perda de especificidade. Por esta razão a sua eficiência é altamente dependente do ***Corpus*** nomeadamente a linguagem e a variância de vocabulário, sendo um excelente candidato a experimentação. Esta técnica é feita através da função **tokStem**, que efetua o processo de **stemming** para cada *token*, por frase, retornando a lista com estas devidamente processadas.

Remoção de acentos foi considerada, no entanto descartada pelo papel crucial que tomam na língua portuguesa. Este método é eficaz quando a correção da gramática do ***Corpus*** não é garantida, eliminando grande parte das discrepâncias resultantes, no entanto assume-se que não é o caso do ***Corpus*** fornecido (apesar disto foi testado não tendo reproduzido alterações visíveis).

Não havendo mais modos de processamento a notar (***Lemmatization*** é talvez complexo de mais dada a simplicidade do problema).

Foi assim decidido testar um algoritmo de distância. As razões acima discutidas para a eficácia de métodos baseados em sets, contrariam métodos como o **edit_distance**, que é mais eficaz em frases grandes com potencial repetição de palavras. Posto isto, para método teste optámos pela utilização do algoritmo ***Dice***.

A similaridade entre as frases do ficheiro TXT e das FAQs é calculada através da função **similarity**, recebendo como argumentos as FAQs (listaPerguntasFAQ), as perguntas (listaPerguntasTXT), uma variável (distance) que dependentemente do argumento que lhe é passado calcula a similaridade, caso este seja ‘’jaccard” usa a **Jaccard distance,** “dice” usa a **Dice distance** e “edit” usa a **Edit distance** e como último argumento recebe uma variável boolean (debug) que pode ser usada para fazer o debug da função (caso o valor desta seja True). Para cada frase de listaPerguntasTXT, a função percorre cada pergunta de listaPerguntasFAQ e calcula a similaridade entre estas usando a respetiva métrica (Edit, Jaccard ou Dice distance, para usar as duas primeiras é feito o respetivo import de nltk.metricas.distance para a Dice distance é usada a função **dice_distance**), caso o valor desta seja inferior ao melhor valor de similaridade (quanto mais inferior o valor, maior é a similaridade entre as perguntas), atualiza este (best) para o novo valor (result) e atualiza o ID da pergunta com maior similaridade (bestId) para o valor da chave da pergunta, que corresponde ao seu ID. No final do ciclo (após comparar a pergunta a todos os elementos de listaPerguntasFAQ), acrescenta o melhor ID à lista de ID’s (ids), caso não exista nenhum elemento que seja suficientemente similar, ou seja, o valor de similaridade nunca é inferior ao valor inicial da melhor similaridade (0.7 para ***Jaccart***, 0.6 para ***Dice*** e 1000 para ***Edit,*** dados parametrizados por experimentação), o ID associado será o melhor ID inicial (0). A função no final retorna a lista com os melhores ID’s (ids).

Por fim, o programa abre o ficheiro com o nome **resultados.txt** e escreve o resultado final neste (lista de ID’s).

Para parametrizar o agente foi criado um ficheiro **training.txt,** contendo frases parafraseadas pelos membros do grupo, de ID conhecido. Com o agente parametrizado, o **test.txt** (o ficheiro fornecido pela professora) foi testado. Deste teste verificou-se manualmente os ID's retornados. Devido à elevada *accuracy* do agente, tornou-se fazível identificar a totalidade dos ID's de teste, pelo que se optou pela sua utilização para a obtenção dos resultados experimentais.

Com o intuito de testar os fatores anteriormente referidos (**stemming** e **Dice**) foram propostas três experiências (as técnicas de pré-processamento anteriormente discutidas são realizadas para as três):

1. ***Jaccard*** sem ***stemming*;**
2. ***Jaccard*** com **stemming*;***
3. ***Dice*** *com* **stemming;**

3. Corpora usado

No HW2 foi estudado o "exemplo.xml" dado pela professora e analisada a sua estrutura, de seguida, em grupo, parafraseámos as frases. Nesta fase optámos por dividir o cada frase parafraseada por entre os membros, de forma a que cada um parafraseasse uma só frase por id. O consenso do grupo foi que desta forma, não só seria mais fácil para cada elemento conseguir uma frase análoga, como também se garantia maior variabilidade de tipos de escrita (intrínsecos a cada membro), bastante apreciável para ao versatilidade do agente.

4. Resultados Experimentais

As experiências anteriormente referidas foram testadas com o ficheiro de teste fornecido, recorrendo ao ficheiro **chatbot.py**, sendo os resultados obtidos através do *debugging* implementado na função **similarity.** Esta funcionalidade retorna as medidas de avaliação do agente, neste caso avaliando o agente, como classificador das perguntas, em pertencentes ao data set (**ID** != 0) ou não (**ID =** 0) (**Nota**: Estes valores não refletem se os ID's retornados são os corretos, apenas se a pergunta pertence ou não ao tipo de perguntas do ***Corpus***). Retorna também a medida de comparação mais relevante, a ID accuracy, que reflete o número de ID's corretamente identificados (ID teste = ID devolvido). De seguida apresentam-se os resultados obtidos (2 e 3 obtiveram os mesmos resultados).

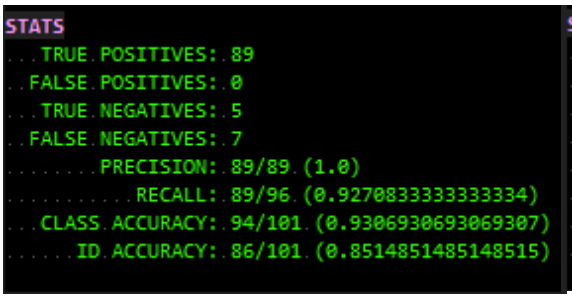

Figura 2 Resultados da experiência 1

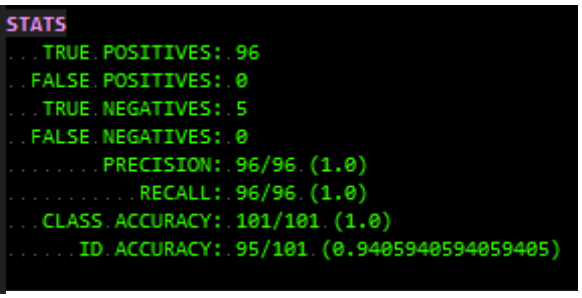

Figura 1 Resultados das experiências 2 e 3

Em primeira análise destaca-se claramente a alta *accuracy* registada nas 3 experiências, pouco usual principalmente quando é fatorizada a simplicidade da metodologia

de *retrieval-based chatbots*. Embora o estudo e parametrização do problema tenham desempenhado um alto papel neste resultado, ao analisar o teste torna-se claro que se deve maioritariamente à simplicidade das perguntas, que embora difiram das da FAQ, partilham ainda assim um grande número de palavras, nomeadamente "palavras chave". Estas são extremamente específicas, como "certificado digital qualificado", limitando as possíveis *matches* a um conjunto mais reduzido, facilitando a escolha através das palavras restantes. Também é de sublinhar o reduzido número de perguntas (100) face ao número de ID's, principalmente quando ocorre grande repetição do mesmo identificador.

Estando esclarecidos os principais fatores externos às experiências, resta avaliar o que foi testado. Como esperado, a utilização de **stemming** foi crucial para a melhoria da accuracy, de 85% para 94%. Analisando os dados do *debug,* destaca-se a redução do número de False Negatives, de 7 para 0. Este dado é indicativo que sem **stemming**, ocorreram ***sets*** sem similaridade suficiente com quaisquer outros. Esta forma de pré-processamento revelou-se então eficaz a aumentar o número de interceções.

Os dados das experiências 2 e 3 são no mínimo caricatos em primeira análise, mas explicáveis com um aprofundamento. Os dados de *debug* não revelam nada de particular, com a exceção de tratar-se de um classificador perfeito. É ao analisar as perguntas falhadas que se tornam claros os resultados. Das 6, 5 referem-se a questão de id 106 "Quais as modalidades de pagamento disponíveis?". Esta é extremamente contextual e pouco específica, sendo particularmente difícil distingui-la das outras questões para um *bot* deste tipo. Assim sendo chegou-se ao acordo que a accuracy atingida foi a máxima para o *testset* fornecido, para um *retrieval-based conversational agent.* Pelas razões expostas acima, na Proposta, para este *dataset,* os algoritmos de distância baseados em ***sets*** "brilham", não ocorrendo sequer diferenciação entre si (os sets formados tinham geralmente o mesmo comprimento, fator decisivo na diferenciação entre ***dice*** e ***jaccart***).

5. Conclusões e Trabalho Futuro

Foi com alguma surpresa que observámos a eficácia de um bot tão simples. De facto, o problema apresentado é facilmente equiparável a problemas reais, nomeadamente o ficheiro xml, que provém mesmo de um caso autêntico. Para estes contextos, de respostas simples e curtas, sem linha de "raciocínio", uma base de dados extensa é capaz de simular comportamento quase "inteligente", sendo uma perspetiva extremamente alcançável, quando comparada a outras lecionadas.

Em termos de melhorias, dificilmente se poderia melhorar o agente em si, estando a melhoria assente no ***Corpus.*** Esta passaria por aumentar consideravelmente o número de paráfrases, nomeadamente garantir as construções gramáticas mais comuns.

Futuramente, seria interessante integrar no *chatbot*, interação com utilizadores, da qual resultaria uma extensão progressiva do corpus.

6. Bibliografia

stopwords.txt https://gist.github.com/alopes/5358189 - @alopes github

Natural Language Processing in a Nut(s)shell – Luísa Coheur

Speech and Language Processing (3rd ed. draft) - Dan Jurafsky and James H. Martin