2º Mini-Projecto de Língua Natural

Realizado pelo grupo 31:

- Nuno Cordeiro nº 84980
- Rui Loureiro nº 80845

Introdução

Para este segundo mini-projecto de Língua Natural, foi-nos apresentada a tarefa de etiquetar questões sobre um dado tema (cinema) com base num set de 16 etiquetas já pré-definidas para o problema, sendo estas: actor_name, budget, character_name, genre, keyword, original_language, original_title, overview, person_name, production_company, production_country, release_date, revenue, runtime, spoken_language e vote_avg. Foram-nos fornecidas também questões devidamente etiquetadas pelas quais deveríamos basear a nossa previsão. Para além deste ficheiro de "treino", tivemos também acesso a um ficheiro com novas questões para etiquetarmos de acordo com a técnica de previsão utilizada.

Proposta de Solução

Para o tipo de problema descrito, o método abordado nas aulas que nos pareceu mais adequado foi o de **número mínimo de edições**, usando as palavras como tokens. Optámos pela linguagem Python, não só por estarmos familiarizados com a mesma, mas também pelo facto de existirem bibliotecas, como a **nltk**, que fornecem já muitas funções úteis para o problema em questão. Assim, optámos numa primeira abordagem por usar a função da biblioteca **nltk**, com todos os parâmetros default.

Decidimos que esta configuração era adequada para obter o baseline, tendo obtido uma accuracy de cerca de 71.43%.

O objetivo seria agora otimizar o algoritmo de número mínimo de edições, bem como processar os ficheiros de entrada.

Começámos pelo pré-processamento do ficheiro de treino (QuestoesConhecidas) bem como o de NovasQuestoes. Tal como foi referido na introdução, foram-nos disponibilizados ficheiros com listas (tokens) de nomes de atores, nomes de filmes, géneros de filmes, posições na área do cinema, nomes de personagens e keywords relacionadas com filmes.

Usando as seguintes frases como exemplo:

- What is the actor in Titanic
- What is the actor in Reservoir Dogs

Para alguém que saiba que 'Titanic' e 'Reservoir Dogs' são nomes de filmes, estas duas queries são exatamente iguais, do ponto de vista de tokens.

Assim, decidimos substituir nos dois ficheiros referidos, usando dicionários, todas as ocorrências de nomes de filmes por *movie*, atores por *actor*, e assim sucessivamente para todos as listas.

Com esta alteração obtemos uma accuracy de, aproximadamente, 73.81%.

Notámos, no entanto, que estava a ser substituído o token *keyword* em grande parte das sentences, em muitos sítios erradamente (do ponto de vista semântico), pois este ficheiro contém strings muito gerais. Retirámos então a substituição de keywords, tendo a accuracy subido para cerca de 76.19%.

Sendo que não vimos uma melhor maneira de utilizar o ficheiro de keywords, decidimos não o utilizar.

A ordem pela qual substituímos é também relevante, pois quando corremos o programa tendo definido a ordem aleatoriamente, notámos que as listas com strings mais pequenas e mais gerais, como é o caso de 'list_characters', estavam a substituir partes de ocorrências de strings maiores e mais específicas, como é o caso de 'list_characters'. Assim, definimos a ordem: list_movies, list_people, list_characters, list_companies, list_genres, list_jobs. Com esta alteração, a accuracy subiu para 85.71%.

Não podemos dizer que esta ordem, apesar de fazer sentido, é óptima, pois com este corpora a accuracy manteve-se para ordens diferentes, não tendo conseguido subir o accuracy mais do que o referido.

Decidimos agora retirar toda a pontuação da frase, pois não nos parecia adicionar significado à mesma, tendo a accuracy subido para aproximadamente 88.1%.

Decidimos de seguida retirar as 'stop words', isto é, aquelas palavras muito comuns em frases que não adicionam significado à mesma, como é o caso de 'the', 'is', 'are', etc. Para este efeito, usámos a lista de stopwords fornecida pela biblioteca nltk.

Com esta alteração, a accuracy subiu para cerca de 92.86%.

Ao analisar os ficheiros de QuestoesConhecidas, observámos que algumas substituições estavam a ser mal feitas. Alguns tokens mais pequenos estavam a substituir partes de outros tokens maiores. Por exemplo, uma frase com o token character 'Uniform Outside Station' estava a ser transformada para '*character* Outside Station', pois existe também um token para 'Uniform'. Assim, ordenámos os dicionários de tokens por ordem decrescente de tamanho.

A accuracy manteve-se igual com esta alteração.

Decidimos agora tentar otimizar o algoritmo de número mínimo de edições.

Começámos pela atribuição de custo à inserção, remoção e substituição. Pela análise do ficheiro QuestoesConhecidas, foi claro que a remoção deveria ter baixo custo, face às outras duas operações. Isto porque no caso de frases com mais palavras, muitas destas não acrescentam nada ao significado da frase, de acordo com o objetivo deste projeto.

No entanto, não conseguimos alterar os custos usando a função da bilbioteca nltk, pelo que decidimos escrever nós o algoritmo, utilizando o pseudo-código fornecido na sebenta da cadeira, com umas pequenas alterações. Assim, atribuímos à operação de remoção um custo nulo, e às restantes operações inserção e substituição um custo de 1.

Com esta alteração a accuracy subiu para 100%.

Foram feitas outras alterações mais simples que não contabilizámos para a alteração da accuracy como a eliminação de linhas em branco, passar todas as strings para lower case, etc.

Conclusão

Concluimos, contudo, que este tagger é apenas capaz de identificar o tipo de questões com base num set pré-definido de tags e em questões cuja sintaxe está também pré-definida e inalterável (ficheiro QuestoesConhecidas.txt) pelo que qualquer questão que fuja à sintaxe das questões já existentes poderá vir a ser mal etiquetada. No entanto, as que seguirem a sintaxe delineada pelo ficheiro de "treino" serão sempre etiquetadas de forma correcta, tendo em conta a técnica utilizada.

Concluimos também que, apesar de o método de número minimo de edições ter tido um bom desempenho para este projeto, tal não aconteceria se fosse necessário processar um corpus de treino maior. Este método realiza todo o processamento (percorrer todo o corpus de treino), sempre que queremos etiquetar uma nova frase, sendo impraticável para corpus grandes. Para corpus maiores, um método baseado num classificador seria bem mais eficiente, porque apenas temos que percorrer o corpus quando queremos construir o modelo, sendo as posteriores classificações de frases desconhecidas realizadas em tempo constante, não dependendo estas do tamanho do corpus.

Bibliografia

Todos os conteúdos teóricos usados para a elaboração deste projecto foram devidamente extraídos a partir da sebenta da Cadeira de LN.