**MP2 – Segundo Miniprojecto de Língua Natural**

Grupo 19:

Gonçalo Bustorff Silva nº82050 Guilherme Gomes nº89378

**Descrição do problema**

É necessário desenvolver uma métrica de similaridade que permite identificar o tipo de uma questão de cinema. Dado o ficheiro **QuestoesConhecidas.txt**, onde se encontram questões pré-anotadas, e outros recursos como lista de filmes ou pessoas consideradas na solução, deverá ser possível classificar as novas perguntas com um dos 16 tipos possíveis:

*actor\_name, budget, character\_name, genre, keyword, original\_language, original\_title, overview, person\_name, production\_company, production\_country, release\_date, revenue, runtime, spoken\_language, vote\_avg*

Exemplificando, diga-se que nas **QuestoesConhecidas.txt** encontram-se as perguntas:

*budget How much budget did Valkyrie have?*

*budget What was the budget of Furious 7?*

Dada a nova questão: *What was American Gangster's budget?*

A métrica implementada deverá classificar a pergunta do tipo: *budget*

**Proposta de solução**

A métrica desenvolvida é uma junção de várias, devolvendo a moda de voto de 3 classificadores já implementados.

Primeiramente, esta carrega os ficheiros de recurso **list\_movies.txt, list\_people.txt e list\_companies.txt**, linha a linha para cada lista correspondente. Da mesma forma é guardada a lista de perguntas pré-anotadas do ARG1. As novas perguntas, presentes no ARG2, são lidas linha a linha, *tokenizadas* por frase, e caso se encontrem ocorrências de *substrings* presentes de forma exata nas 3 listas de recurso processadas, ocorre uma substituição por *movie\_title*, *person\_name* e *prod\_company (replace\_with\_word(sentence,word\_list,word)).* Assim evita-se que títulos de filmes ou nomes de atores/personagens influenciem o peso da decisão, fazendo com que estes se tornem palavras muito comuns, diminuindo a informação/peso que estas carregam. Exemplos deste processo:

*Which are the most relevant actors in Bad Boys? → Which relevant actors movie\_title?*

*What movies did Michael Bay direct? → What movies person\_name direct?*

Apresenta alguns erros devido ao *matching* exato na lista, mas parcial na questão:

*Which characters were there on Goldfinger? → Which characters movie\_titleldfinger? –* Existe “Gold” na list\_movies.

*Which are the most relevant actors in Great Gatsby? → Which relevant actors Great Gatsby? –* Não existe “Great Gatsby”, mas sim “The Great Gatsby”.

Visto que estes pequenos defeitos não têm um efeito notável na aprendizagem efetuada (demonstrado na secção de resultados), mantivemos a substituição. As possíveis substituições dos elementos com **list\_characters** e **list\_keywords** não justificam o processamento que estes dados necessitam, desde modo ignoradas. As restantes foram também excluídas visto considerarmos que as substituições possíveis influenciaram o peso ou de forma pouco evidente, ou de forma errada. No fim são retiradas as *stop words*, utilizando o conjunto “*english*” disponibilizado pela biblioteca NLTK.

Em seguida é processada a *train\_data.* Após lido e guardada cada linha numa posição o ARG1, a linha é separada em (pergunta, tag). À pergunta é feito um processamento semelhante ao feito às novas questões.

Na fase final de processamento de dados, as perguntas já conhecidas são tokenizadas por palavra, transformadas em minúsculas, a lista transformada num *set*, e para cada tag é verificada a existência de cada palavra nesse set, sendo que este processo transforma a *raw data* num *feature set* válido para treino – (‘pergunta’,’tag’).

Após verificação dos dados de treino, os classificadores são definidos e treinados. São usados os classificadores *LogisticRegression, SGDC* e *LinearSVC*, com parâmetros recomendados. Estes 3 classificadores são usados num argumento de um outro classificador explicado posteriormente (*voted\_classifier*).

As perguntas a serem testadas (ARG2 processado anteriormente) são tokenizadas por palavra, transformadas em minúsculas, classificadas pelo *voted\_classifier* e os resultados guardados numa lista, mais tarde impressa e comparada com a lista de resultados esperados de modo a calcular a precisão do classificador.

**O *voted\_classifier. classify()*****não classifica os dados *per se*.** Este recolhe os dados dos 3 classificadores treinados e devolve a moda das votações, i.e., o que mais classificadores concorda. Caso os 3 discordem no resultado, sendo a lista de resultados de igual tamanho ao *set* de resultados, é devolvido o resultado do *LogisticRegression,* visto este ter sido o que apresentou em média melhores resultados. Encontra-se implementado na classe *VotedClassifier*, onde também se encontra incluída a função *confidence*, que devolve a confiança (o rácio de concordância entre classificadores) de cada classificação.

**Discussão de resultados**

Inicialmente classificámos as novas perguntas com o classificador Naïve Bayes sem qualquer tipo de processamento de dados (entenda-se, dados *raw* formatados devidamente em *feature sets*). A precisão média era de 64.28%, com resultados bastante instáveis.

O primeiro processamento introduzido foi a remoção de *stop words*, que melhorou a precisão média para os 95.24%. O resultado foi repetido com o classificador DecisionTree.

O próximo processamento foi o *stemming* dos dados. Apesar de nos parecer uma decisão razoável, isto fez com que a nossa precisão sofresse uma quebra considerável, baixando para os 50 a 56%, sendo que este foi abandonado e excluído de testes precedentes.

Os nossos melhores resultados obtidos apresentavam erros recorrentes que, especulámos, seriam devido à presença dos nomes dos filmes nos dados de treino e novas questões, devolvendo resultados muito provavelmente influenciados por estes. Solucionámos o problema olhando para os recursos opcionais. Processámos a lista de filmes e procedemos à substituição “nome do filme” → “*movie”*. Obtivemos uma média de 71.43% com Naïve Bayes e 88.10% com Decision Tree, até nos apercebermos que substituirmos por “*movie”* é má ideia, visto que adiciona esta palavra a tags onde esta era inexistente. Mudámos “*movie”* para “*movie\_title”*, palavra com muito pouco peso dado a frequência, subindo a precisão para 97.62% com o Decision Tree.

Decidimos testar a nossa solução de maneira mais exigente, escrevendo manualmente mais perguntas e respostas esperadas nos ficheiros a testar. Tivemos por base as perguntas já introduzidas e os recursos fornecidos, mas tentámos fazer com que as perguntas fossem ligeiramente diferentes. Sem alterações aos classificadores e processamento de dados, obtivemos uma média de 92%, subindo para 93% após correção de gralha numa das novas perguntas adicionadas (*runtime* → *run time*).

**ALTERAÇÕES AOS FICHEIROS:** Foram adicionadas novas perguntas ao ficheiro **NovasQuestoes.txt**, todas após a última pergunta do ficheiro no estado original, de forma a que existissem perguntas de teste que cobrissem todas as tags possíveis. Também foram adicionados os resultados esperados para cada uma das novas perguntas no ficheiro N**ovasQuestoesResultados.txt.**

Antes da gralha já mencionada ser corrigida, modificámos o algoritmo classificador para *LogisticRegression* e outros como *SGDC*, *LinearSVM* e *MultinomialNB.* Exceto este último, todos apresentaram resultados melhores aos antes testados, sendo que o nosso foco se virou para como melhor aproveitar estes 3 classificadores. Após correção, o classificador *LogisticRegression* apresentava 100% constante, com os outros 2 errando apenas uma pergunta (≈98%). Apesar dos resultados bastante satisfatórios, o nosso corpus de teste não apresentava um tamanho suficiente para termos plena confiança nos nossos algoritmos. Implementámos a classe *VotedClassifier,* que, mantendo os 100% de precisão no corpus de teste atual, especulámos que funcione melhor com corpus maiores. Por fim, expandimos as substituições para a **list\_people** e **list\_companies**, solidificando a nossa solução.

**Conclusão**

Gostaríamos de ter expandido bem mais o nosso corpus de teste, e talvez o de aprendizagem, de forma a garantir melhores resultados, mas estamos satisfeitos com a solução desenvolvida. Cumpre os requisitos e apresenta uma taxa de precisão de 100% para um corpus de teste mais extenso do que o fornecido, este que incluí todas as tags possíveis.

Gostaríamos também de testar outros algoritmos de teste principalmente o TF-IDF, de forma a melhorar o nosso VoteClassifier, e de processar as restantes listas e integrá-las na nossa solução, solidificando as respostas dadas. Transformar o algoritmo numa *Pipeline* seria também uma abordagem interessante.

**Referências:**

<https://www.nltk.org/book/ch05.html> - NLTK book. Como classificar usando Naïve Bayes e Decision Tree

<https://pythonprogramming.net/sklearn-scikit-learn-nltk-tutorial/> - Classificação com o Sklearn

<https://pythonprogramming.net/combine-classifier-algorithms-nltk-tutorial/> - VoteClassifier