Relatório do 2º Mini-Projeto de Língua Natural

Vasco Piussa - 93762
Rafael Candeias - 93748

Instituto Superior Técnico, Lisboa, Portugal

10 de Novembro de 2021

1. Introdução

Para a resolução do projeto que nos foi apresentado, após muita pesquisa e planeamento, criámos dois modelos diferentes, o **M1** e o **M2**. Ambos possuem o mesmo ficheiro de treino e de teste, *trainWithoutDev.txt* e *dev.txt*, respetivamente. Têm um funcionamento geral similar, e apresentam linha a linha no ficheiro *results.txt* a categoria que calcularam para cada par: pegunta/resposta do ficheiro input. No que toca a avaliação, foi criado um ficheiro python que calcula a "accuracy" de cada modelo, comparando linha a linha a categoria do ficheiro de resultados com a do ficheiro de teste.

2. Modelos

O modelo M1 contém os seguintes atributos e métodos: uma lista para cada categoria, pergunta e resposta, um ficheiro de treino, um ficheiro de teste, um objeto que suporta o algoritmo Snowball Stemmer, um valor de threshold, um método stem que aplica o algoritmo Snowball Stemmer a uma frase e devolve a sua simplificação, um método jaccard que calcula a distância entre duas frases, e por fim um método compute que executa o modelo. A sua execução segue os seguintes passos: Em primeiro lugar, dividimos cada linha do ficheiro de treino em três segmentos: categoria, pergunta e resposta. A categoria é adicionada a uma lista de categorias. Por outro lado, tanto a pergunta como a resposta são separadas palavra a palavra (tokenization). A cada palavra aplica-se o algoritmo Snowball Stemmer que a reduz à sua raíz (stemming). Terminado o pre-processing, ambos os segmentos são adicionadas à sua lista. Também dividimos cada linha do ficheiro de teste pelos mesmos 3 segmentos. Ignorando o primeiro, aplicamos o mesmo pre-processing aos restantes, (à pergunta e à resposta). Posteriormente, calculamos o jaccard da pergunta da atual linha do ficheiro de teste com todas as perguntas do ficheiro de treino. Caso o seu valor seja igual ou inferior à threshold selecionada, repetimos o processo com a pergunta seguinte no atributo lista de perguntas. Contudo, se não for o caso, aplica-se o jaccard entre a resposta da linha de teste com a resposta à pergunta atual do ficheiro de treino. Somam-se os jaccards da pergunta e da resposta e reponde-se com a categoria da linha de treino que tiver o maior valor. zNo que toca o modelo M2, para realizar o pre-processing, utilizámos tokenization por palavra a palavra, o algoritmo Porter's Stemmer, stop words e stop chars. As stop chars seguem o mesmo conceito que as stop words, mas focam-se nos caractéres presentes numa palavra. Para calcular a probabilidade de uma linha ser de uma categoria, optámos por uma vertente do algoritmo Naïve Bayes, o Complement Naïve Bayes (CNB). Este modelo contém um ficheiro de teste, um ficheiro de treino, um conjunto com todas as palavras existentes em cada categoria, um dicionário com o número de palavras em cada categoria, um vocabulário, um atributo com o número de palavras no vocabulario, um atributo com o número de linhas do ficheiro de treino, um dicionário com a probabilidade de cada categoria, um conjunto de stopwords, um conjunto de stopchars, um objeto que suporta o algoritmo Porter's Stemmer, um método que realiza todo o pre-processing, excepto tokenization, um método que calcula a probabilidade de uma palavra não ser de uma categoria, um método que calcula a probabilidade do CNB e um método que executa o modelo. O modelo inicia com a leitura de cada linha do ficheiro de treino e o seu pre-processing, onde reune informação necessária para o CNB. De seguida, aplica o CNB a cada linha do ficheiro de teste, e devolve a categoria que obteve a maior probabilidade.

3. Configuração Experimental

Para o modelo M1, começámos por exeperimentar o algoritmo Jaccard, método que calcula a proximidade entre dois objetos. O algoritmo segue a seguinte fórmula: —A.inter(B)—/—A.union(B)—. Tomámos esta decisão, uma vez que o algoritmo é rápido, intuitivo, fácil de implementar e suficiente para a baseline. No entanto, uma vez que temos de calcular a distância da pergunta de treino (Q1)

à pergunta de teste (Q2) e da resposta de treino (A1) à resposta de teste (A2), temos duas possibilidades de cálculo: Jaccard(Q1, Q2) + Jaccard(R1, R2) ou Jaccard(Q1 + R1, Q2 + R2). Testadas as duas alternativas, selecionámos a que nos deu melhor performance: Jaccard(Q1, Q2) + Jaccard(R1, R2) com ACC: 0.706. (Jaccard(Q1 + R1, Q2 + R2) deu ACC: 0.608). Começámos por utilizar o algoritmo Porter's Stemmer devido à sua fama e simplicidade. Com jaccard e Porter's Stemmer, obtivemos uma accuracy de 0.694, que é suficiente para a baseline pretendida. Contudo, após alguma investigação, deparámo-nos com um algoritmo mais otimizado e correto, denominado Snowball Stemmer, que mantém o sentido das palavras, tal como demonstrado nos seguintes exemplo: [porter('generically') to gener; porter('generous') to gener; snowball('generically') to generical; snowball('generous') to generous]. Deste modo, atualizámos o modelo e obtivémos uma accuracy final de 0.706. Para além destes stemmers, também descobrimos o Lancaster Stemmer que tem regras mais agressivas. Devido à sua intensidade, por vezes simplifica demasiado a palavra, retirando-lhe o significado inicial, tal como demonstrado no seguinte exemplo: [snowball('sales') to 'sale'; snowball('salty') to 'salti'; lanc('sales') to 'sal'; lanc('salty') to 'sal']. Mesmo assim, decidimos implementá-lo dando uma accuracy de 0.704. Assim, chegámos à baseline final: tokenization por espaços, Snowball Stemmer e Jaccard(Q1, Q2) + Jaccard(R1, R2).

No que toca o desenvolvimento do modelo M2, inicialmente utilzámos o algoritmo Naïve Bayes, método que aplica o teorema Bayes com a presunção "naive" da independência condicional entre todos os pares de features dados de uma classe. O algoritmo segue a seguinte fórmula: P(Category—Line) = P(Category)*P(wordiLine—Category). Tomámos esta decisão, uma vez que o algoritmo é rápido, intuitivo, conhecido e fácil de implementar. Para stemming experimentámos com o snowball stemmer, já que foi com o que obtivémos melhor resultados no M1, e stop words dadas pela livraria nltk. Com esta implementação, a accuracy deu 0.792. Contudo, após uma profunda investigação sobre o tema, encontrámos uma vertente do algoritmo denominada Complement Naïve Bayes (CNB). Esta alternativa tem em conta a incoerência da frequência das categorias existentes. Sabendo que o nosso corpus está repartido da seguinte maneira: 0.1557 MUSIC, 0.2735 HISTORY, 0.2358 LITERATURE, 0.1066 GEOGRAPHY, 0.2319 SCIENCE, experimentámos o CNB e alcançámos uma accuracy de 0.823. Porém, ainda não alcançámos o valor desejado, por isso investigámos um pouco mais o corpus dado. Deparámo-nos com diversos erros, tais como: " "Bix Biederbecke's bio: ""Young Man With A _""".

Esta frase, tal como muitas outras, tem caractéres indesejados ("_", """). Assim, inicialmente implementámos o seguinte conjunto de caractéres como stop chars : "","," : ","?","!",";","""","","/","(",")","[","]"," – ".

Obtemos assim, uma accuracy de 0.832. Porém, após vários testes concluímos que apenas os seguintes é que melhoram o código: "!", """", "", "-" e substituindo os conjuntos, a accuracy cresceu para 0.838. Pondo os stop chars e stop words de parte, experimentámos o Porter's Stemmer que nos subiu a accuracy para 0.846, e o Lancaster Stemmer, que reduziu a accuracy para 0.834. Concluindo, chegámos a um modelo final: tokenization por espaços, Porter's Stemmer, stop words, stop chars e Complement Naïve Bayes. z

4. Results

	Baseline	Modelo2
accuracy	0.706	0.848
exec time	29.707s	40.496s
accuracy	0.706	0.848
exec time	30.103s	41.342s
accuracy	0.706	0.848
exec time	29.372s	40.582s

5. Error analysis Future Work

Durante a execução do projeto, notámos que deviamos ter feito mais pesquisa de outros algoritmos que nos possam oferecer melhor desempenho, neste caso, melhor precisão. Olhar mais para os resultados, estamos conscientes que não avaliamos todas as opções, tendo ficado demasiado obececados com atingir a precisão da baseline mais "forte". Da mesma forma, deixámos por explorar a maior parte das bibliotecas disponíveis para a execução do projeto, perdendo secalhar alguma precisão pois optámos por escrever o algoritmo Naive Bayes "manualmente". Não termos usado algumas tecnicas de representação de dados" como por exemplo, "bag of words", nos fez ter menos hipoteses em relação aos algoritmos que tinhamos disponíveis. Não aplicamos técnicas de visualização de dados para perceber o que se estava a passar, focando mais, como ja tinha dito, na precisão como a única métrica para nos guiar. Outro erro, foi não avaliar os resultados obtidos e ver o que estava a falhar e tentar descobrir o que tinha de ser mudado. Para futuros projetos, iríamos alocar mais tempo para investigar outras tecnicas de representar e visualizar dados e certamente prestar mais atenção aos resultados obtidos e desenvolver o projeto com base nesses resultados, tentando nao focar só na precisão.