Língua Natural - MP1

Hugo Guerreiro 83475

Matilde Ramos 83526

1 Introdução

Neste projeto, foi-nos proposto o desafio de criar uma *"métrica de similaridade que permita identificar o tipo de uma questão dada sobre cinema"*. Este trata-se de um problema de categorização de texto (também conhecido como classificação) onde, dada uma frase, é atribuída uma etiqueta dependendo do seu conteúdo e significado. Para resolver este problema utilizámos métodos de aprendizagem supervisionada sobre o corpus fornecido (208 frases previamente etiquetadas) e um conjunto de ficheiros de recursos auxiliares.

2 Proposta de solução

Para resolver o problema de classificação proposto decidímos utilizar uma abordagem direcionada à aprendizagem supervisionada. Este tipo de aprendizagem é fundamentalmente representada por 4 passos (figura 1):

- Preparação do corpus
- Escolha das features
- Treino do modelo
- Melhoria da performance do classificador treinado

Nas próximas secções, descrevemos as opções escolhidas durante o desenvolvimento do projeto.

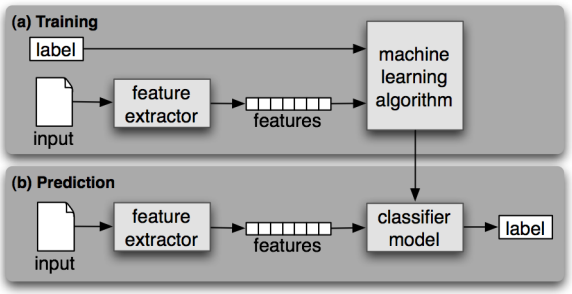

Figure 1: Aprendizagem supervisionada

2.1 Preparação do Corpus

Neste primeiro passo começámos por preparar o corpus da seguinte forma:

- **Ler o corpus** - ler o corpus do ficheiro e analisá-lo para separar as etiquetas das questões de treino.
- **Retirar a pontuação e tornar todas as letras minúsculas** - este passo é fundamental para não existirem palavras iguais mas que são consideradas diferentes devido às letras maiúsculas. Dado que todas as frases são questões, também não faz sentido considerarmos a pontuação (i.e não contribuem com informação para o modelo).
- **Substituir as palavras mais relevantes** - neste passo, começamos por ordenar as listas de palavras fornecidas por ordem alfabética e da palavra com maior comprimento para a menor. De seguida, percorremos cada frase do corpus e substituímos cada palavra conforme o seu significado. Por exemplo, uma palavra (ou palavras) que representa o nome de um filme é substituida por _MOVIE_. Isto permite que nomes compridos (ex: Pirates of the Caribbean: On stranger Tides) sejam melhor modelados por, por exemplo, unigramas (bag-of-words).
- **Lematização e remoção de stop words** - para cada palavra no corpus extraímos o seu lema de forma a diminuir o número de variantes de palavras com a mesma semântica (ex: act, actor, actors). Também removemos as stop words, isto é, palavras como "and", "the" e "him", que se presumem serem pouco informativas na representação do conteúdo do texto, evitando-se predições erradas.

2.2 Escolha das features

Neste passo começámos, numa primeira fase, por tentar extraír as features que achávamos mais relevantes "manualmente". Este método rapidamente se revelou ineficaz pois, por um lado estávamos a introduzir assumpções à informação que tínhamos e por outro as features extraídas podiam não ser as melhores.

Posto isto, decídimos utilizar o algoritmo *TF-IDF* para a extração de features. Este algoritmo calcula um valor para cada palavra num documento (no nosso caso, um documento é uma questão), relativamente à percentagem de documentos em que ela aparece. Valores muito altos indicam que a palavra é relevante para o documento em que elas aparecem.

2.3 Treino do modelo

Agora que o corpus está processado e as features extraídas, passamos à escolha do algoritmo de aprendizagem e do treino do mesmo. Para isso, foram testados uma série de classificadores diferentes da biblioteca sklearn, os quais comparamos na secção seguinte.

2.4 Melhoria da performance do classificador treinado

Após a escolha de um classificador, procedemos à otimização dos seus parâmetros de forma a obter melhor resultados. Nomeadamente, foi tipo em conta que o corpus não se encontra balanceado, logo as classes com mais amostras passaram a ter menos relevância do que as com menos.

3 Resultados experimentais e discussão

Para o treino do classificador decidimos experimentar vários algoritmos. Os algoritmos que levaram a uma melhor performance foram os seguintes:

- MultiLayer Perceptron
- K Neighbors
- Linear SVC
- Decision Tree
- Random Forest

Para avaliar a performance do algoritmo utilizámos duas medidas: a accuracy e o cross validation score. Não utilizámos apenas a accuracy pois esta não é uma boa medida para o quão generalizável está o nosso modelo, o que não nos permitiria decidir informadamente sobre qual o melhor algoritmo a utilizar.

Numa primeira fase escolhemos testar primeiro os algoritmos que são, normalmente, mais utilizados em análise textual (como por exemplo regressão multinomial, Naive Bayes e SVM) mas não obtivemos bons resultados. Assim, experimentámos outros algoritmos e, tal como podemos ver pela tabela 1, o algoritmo que teve melhor performance foi o Random Forest. Dado que utilizámos o TF-IDF, obtivemos um feature space onde as features extraídas são maioritariamente relevantes e, dada a natureza do algoritmo Random Forest, conseguimos obter bons resultados.

Dado que é um corpus muito pequeno (menor que 100 mil amostras) o algoritmo Linear SVC também demostra uma boa performance.

Classificador	Accuracy (%)	Cross-Validation Score (%)
MultiLayer Perceptron	100	77.1
K Neighbors	95.24	68.31
LinearSVC	100	81.93
Decision Tree	97.62	73.42
Random Forest	100	86.34

Table 1: Algoritmos testados e respetivo resultado

4 Conclusões e trabalho futuro

Ao longo do projeto surgiram vários desafios, entre eles tentar obter bons resultados face ao reduzido tamanho e diversidade do corpus. Contudo, dado um bom processamento e a utilização do TF-IDF foi possível contornar este problema. Posto isto, e uma vez que o corpus não estava balanceado, é possível que os resultados não sejam tão precisos para um conjunto de teste mais diversificado. Futuramente, seria ideal proceder à população do corpus de maneira a torná-lo mais balanceado.

References

[1] Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, Second Edition: Daniel Jurafsky James H. Martin 2009 Prentice-Hall

[2] http://scikit-learn.org/stable/modules/feature_extraction.html

[3] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[4] https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

[5] https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

[6] http://cs229.stanford.edu/proj2013/ShiraniMehr- SMSSpamDetectionUsingMachineLearningApproach.pdf