

Jeonerator - Previsão de categorias para perguntas no Jeopardy

Processamento Computacional da Língua

Trabalho 2

Diogo Fernandes 95000

Jorge Gonçalves 96180

20 de novembro de 2021

1 Introdução

Este trabalho teve como objetivo o desenvolvimento de uma ferramenta capaz de prever uma categoria dado uma pergunta e resposta.

As categorias a considerar são as seguintes: GEOGRAPHY, MUSIC, LITERATURE, HISTORY e SCIENCE.

A ferramenta utiliza um modelo de língua estatístico treinado por um corpus que tem associações de perguntas e respostas às suas respetivas categorias.

Deste modo, a ferramenta é ser capaz de, dado um conjunto de questões e respostas, identificar a categoria de cada uma delas.

Esta ferramenta é útil quando se tem como objetivo aumentar o leque de perguntas de um jogo e se quer evitar categorizar as mesmas manualmente.

2 Opções Tomadas

Para a implementação da ferramenta foi escolhida a linguagem de programação Python em conjunto com algumas bibliotecas de apoio. Foi usada a biblioteca Natural Language Toolkit (NLTK) para várias operações, nomeadamente tokenização, geração de ngramas e contagem de ngramas. Utilizou-se também o pandas para algumas operações com DataFrames e spacy para atribuição de labels as palavras.

2.1 Geração dos unigramas e bigramas

Para a geração dos unigramas e bigramas começou-se por obter todas as categorias presentes no DataFrame e, para cada categoria, tokenizaram-se as frases em listas de palavras, originando uma matriz em que cada linha representa uma frase [1].

Após isso, para cada uma das categorias, geraram-se os ngramas de cada frase (com o padding de início e fim de frase) [2].

Acumulados todos os ngramas, fez-se a contagem de cada um deles e escreveu-se para o ficheiro correspondente à categoria.

2.2 Pré-processamento

Para o pré-processamento criaram-se várias funções de transformação que aplicam uma transformação específica a cada palavra de uma dada frase.

- `to_lemma`: Converte as palavras para o seu lemma
- `remove_stopwords`: Remove as stopwords do inglês

- `clean_words`: Limpa tudo o que não seja sequência alfanumerica
- `to_lower`: Converte para minúsculas
- `to_year`: Converte os anos (e.g. 1997) para a string `__YEAR__`
- `to_stem`: Reduz as palavras para o seu radical e.g. `cooking` ou `cooked` para `cook`
- `to_label`: Atribui a determinado grupo de palavras uma etiqueta e.g. `Lisboa` -> `__GPE__`

2.3 Classificador

O classificador implementado tem suporte a 3 tipos de modelos: modelo de unigramas, de bigramas e de bigramas com alisamento (UNIGRAM, BIGRAM e BIGRAM_SMOOTHING, respetivamente). Recebe como argumentos o tipo de modelo, a pasta com os ficheiros de ngramas e o nome do ficheiro onde estão as questões a avaliar:

Listing 1: run classifier

```
#!/bin/bash
python3 classifier.py BIGRAM counts2 eval-questions.txt
```

Destacamos a implementação de dois dos três modelos:

- **Unigrama[3]**: Adicionamos um `threshold` para lidar com as palavras desconhecidas o que nos permitiu alcançar bons resultados nos nossos testes em comparação com os outros dois modelos.
- **Bigram Smoothing[4]**: Optámos pelo Add-K Smoothing em vez do Laplace Smoothing, em vez de simplesmente adicionarmos 1, adicionamos um valor fracional (k).

3 Resultados Experimentais e Discussão

3.1 Resultados sem pré-processamento

Sem Pré-Processamento			
Modelo	Precisão	Recall	F1-Score
UNIGRAM	83%	83%	83%
BIGRAM	84%	12%	4%
SMOOTHING	78%	78%	78%

3.2 Resultados com pré-processamento

Com Pré-Processamento (case folding, lemmatization e handle year)			
Modelo	Precisão	Recall	F1-Score
UNIGRAM	82%	82%	82%
BIGRAM	85%	12%	4%
SMOOTHING	79%	79%	79%

4 Análise dos Resultados e Avaliação Final

Nas nossas experiências verificamos que com pré-processamento o nosso modelo bigrama com alisamento foi melhorando com a implementação de algumas técnicas de normalização de texto, e que o modelo unigrama piorava com cada alteração que efectuávamos aos documentos. Experimentamos varias abordagem com objetivo de melhorar os resultados obtidos:

- stemming
- lemmatization

- case folding
- remoção de stopwords e pontuação
- atribuição de etiquetas as palavras como anos, pessoas ou locais

O melhor resultado que obtivemos nas várias experiências que efectuamos foi com o modelo unigrama sem pré-processamento, onde alcançamos 83% (f1-score).

4.1 Classificar questões com modelos de língua

O modelos de línguas são uma óptima ferramenta para classificação de temas/tópicos, ao agregarmos os dados no nível de ngramas, podemos retirar instantaneamente temas que, de outra forma, seriam impossíveis de identificar ao analisar os documentos na sua totalidade.

5 Conclusões

No final deste trabalho foi possível concluir que os modelos de língua são uma ferramenta poderosa na classificação e que podem ser aplicados em vários contextos distintos.

Observámos que certas ações de pré-processamento do texto nem sempre levam a uma melhor classificação e que essas transformações têm que ser alvo de análise para o contexto em questão.

Com a utilização de outras abordagens de alisamento mais adequadas aos modelos de língua, tais como backoff, interpolation e Kneser-Ney Smoothing, podiam ter sido observados melhores resultados.

Referências

- [1] N-GRAM Language Model with NLTK | Kaggle. Disponível em: <https://www.kaggle.com/alvations/n-gram-language-model-with-nltk>.
- [2] NLTK__INIT__PY at develop, nltk/nltk, GitHub. Disponível em: https://github.com/nltk/nltk/blob/develop/nltk/lm/__init__.py.
- [3] NLP Programming Tutorial 1 - Unigram Language Models | NAIST. Disponível em: <http://phontron.com/slides/nlp-programming-en-01-unigramlm.pdf>.
- [4] DANIEL, J.; MARTIN, J. H. Speech and language processing. 2021.