DOCUMENTAÇÃO PROJETO DOTZ

André Sacilotti

andre.sacilotti@gmail.com

1. COMPETIDOR

Nome da competiçao: Hackathon DOTZ.

Pontuação Placar Publico: 0.8679.
Colocação Placar Publico: 3º

2. RESUMO DA ABORDAGEM

O conceito inicial é a classificação, trabalhando com processamento de linguagem natural. No presente projeto fora usado conceitos como vetorização TF-IDF e Word Embeding, sendo usado em modelos de aprendizado de maquina como naive bayes, multi layer perceptron e o atual estado da arte, roBERTa. Devido a pequena quantidade de dados, fora usado a versão destilada, conhecida como DistilRoBERTa, além de um modelo pretrainado em textos em portugues.

3. PROCESSAMENTO DE TEXTO

Fora realizado experimentos relacionados ao processamento de dados, como remoção de numero, remoção de virgulas, e outras mudanças, no entanto, o maior score geral foi obtido com os dados brutos. Seguidamente, para processar tais dados brutos, fora usado o algoritmo TF-IDF, limitando a quantidade maxima de features a dez mil e o preprocessamento de dados do RoBERTa. Além disso foram feitos experimentos relacionados a vetorização de palavras com word2vec, BOW, fasttext, no entanto, apenas os dois supracitados obtiveram os melhores scores. É importante ressaltar o resample em registros em que pertenciam a uma Sub-Categoria que tinha apenas 1 registro, permitindo uma divisão estratificada para testes.

4. MODELOS DE APRENDIZADO DE MAQUINA

Para ambos os targets, foram ajustados minuciosamente algoritmos como, Maquina de Suporte de Vetores, Naive Bayes, Redes Neurais, Florestas Aleatorias e K-Vizinhos. Na Classificação da Categoria, os algoritmos K-vizinhos, Florestas tiveram uma participação positiva, diferentemente do que aconteceu na Sub-Categoria, onde, sua presença ocasionava Overfiting, portanto, obteve-se um acrescimento de 10% em relação ao dataset de validação, porem, ocasionou um decrescimento de aproximadamente 5% no score final. Analisando a f1 score de cada categoria, em cada classificador, foi possivel perceber que modelos diferentes, classificavam classes corretamente de maneira diferente, o que abriu caminho para o ensemble de modelos. O modelo de rede neural do DistilRoBERTa permitiu atingir um patamar de classificação elevado, obtendo 94% de acerto referente a Categoria dos dados de validação, em comparação a 82% obtido usando TF-IDF com perceptron.

5. ENSEMBLE

Dado ao fato descoberto, de que diferentes modelos classificam de maneiras diferentes, a estrategia adotada para aumentar o score foi o Stacking, que, dados n predições em forma de probabilistica, procura os pesos ideais a cada modelos, com o objetivo de obter o melhor f1-score. Para cada feature foi realizado um experimento diferente

5.1. Sub-Categoria

Após diversos testes, o melhor stacking encontrado é formado pelo seguintes algoritmos:

- Multinomial Naive Bayes [Alpha = 0.01]
- Complement Naive Bayes [Alpha = 1.0]
- MultiLayer Perceptron [optimizer = Adam, (200,0) e (2700,540) dense layers, activation = relu, output_activation = softmax]
- DistilRoberta [maxlen = 100, 20 epochs, batch_size = 128]
- BERT-Base-Portuguese-Cased [maxlen = 100, 25 epochs, batch_size = 128]

5.2. Categoria

O Stacking obtido para a feature Categoria, foi diferente, incluindo outros modelos

- Multinomial Naive Bayes [Alpha=0.0025]
- Complement Naive Bayes [Alpha = 1.0]
- MultiLayer Perceptron [optimizer = Adam, (500,0) e (2700,540) dense layers, activation = relu, output_activation = softmax]
- DistilRoberta [maxlen = 100, 20 epochs, batch_size = 128]
- KNN [n_neighbors=12,weights=distance]
- Random Forest [class_weight= balanced, n_estimators=80]

6. CONCLUSÕES

Dado o fator de ser minha primeira experiencia com processamento de linguagem natural, o resultado obtido na competição foi muito bom, garantindo a terceira colocação no placar não oficial, no entanto, há algumas tratamentos que poderiam incrementar os resultados, como a inserseção de mais dados, já que features como a sub-categoria estão completamente desbalanceadas, possuindo muitas vezes apenas um unicos registro referente. Trabalhar nativamente com modelos pretreinados poderia ajudar a realizar um fine-tuning nos hyperparametros, adicionando mais droupouts por exemplo, aumentando o score.

7. TESTANDO O MODELO

Para roda-lo, foi necessario dedicar aproximadamente 3GB de memoria RAM. Ter um gpu pode acelerar o processo relacionado a classificação dos modelos pre-treinados

No Repositorio é possivel encontrar um arquivo referente ao ipython, onde é possivel acompanhar passo a passo do algoritmo, porem, executando o seguinte comando

```
pip install -r requirements.txt

python ./main.py data/Hackathon_Base_Teste.csv
```

Substituindo o endereço do arquivo conforme quiser classificar uma nova base, é possivel gerar o dataset já classificado. Portanto, ao final, sera gerado um arquivo, dentro da pasta out, com o nome ANSWER.CSV.

O algoritmo rodará todo o processo necessario, e ao final salvará um arquivo com as classificações. Dependendo das configurações do computador pode-se demorar 0.03 segundos/dado até 0.3 segundos/dado, conforme testes realizados com diferentes CPU e GPUS

8. CODIGO E AGRADECIMENTOS

Todo codigo está disponivel no github https://github.com/Andre-Sacilotti/Hackathon-DOTZ

Ademais, gostaria de agradecer a DOTZ por permitir uma oportunidade tão incrivel como essa, e a toda equipe tecnica envolvida!