Universidade de Fortaleza

Tópicos Avançados em Processamento de Linguagem Natural

DANIEL MORAES

Desenvolver módulo NLU para aplicação de Chatbot

Fortaleza, CE

2021

RESUMO

Este trabalho desenvolve um módulo de NLU (Natural Language Understanding) de um sistema para chatbot fim-a-fim que recebe dois textos em linguagem natural e define se o 2o texto é resposta para o primeiro texto. Será fornecido um dataset com 180000 pares de sentenças que representam diálogos positivos, em inglês. Os diálogos negativos deverão ser *"montados"* pela equipe.

Palavras-chave: PLN. Chatbot. Aprendizado de Máquina. Inferência. BERT.

1. Introdução

O Processamento de Linguagem Natural (PLN) é uma subárea de estudo da Inteligência Artificial (IA) que tem como principal objetivo estudar a capacidade e as limitações dos computadores compreenderem a linguagem dos seres humanos. Desta forma, é possível reconhecer o contexto, sintaxe, semântica, morfologia e analisar sentimentos.

Há inúmeros trabalhos no campo de extração de informações utilizando PLN. Como por exemplo “SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods” (RIBEIRO et al, 2016), “Survey on mining subjective data on the web” (TSYTSARAU; PALPANAS, 2012), “Approaches, Tools and Applications for Sentiment Analysis Implementation” (D’ANDREA et al., 2015), entre outros. Textos em linguagem natural representam um conjunto relevante e significativo para análise e produção de conhecimento. Este trabalho propõe uma análise das etapas de pré-processamento e de treinamento de um classificador de textos, que utiliza duas sentenças em linguagem natural para que o classificador defina a relação de inferência entre ambas.

2. Metodologia

2.1. Conjunto dos Dados

Inicialmente foram disponibilizados 180.000 pares de sentenças rotulados para treinamento e teste do modelo de classificação com as seguintes features: pair_ID (identificação do par de frases), message (texto da pergunta em inglês), response (resposta da pergunta em inglês), entailment_label (anotação da relação de inferência entre o par de frases para POSITIVAS e NEGATIVAS).

2.2. Método

O método descrito a seguir trata-se de uma pesquisa empírica de natureza exploratória utilizado no treinamento do modelo de classificação que permite inferir a relação entre duas sentenças. Seguindo a ordem de entrega dos produtos e utilizando a linguagem de programação Python 3, o processo de treinamento e teste do modelo teve as seguintes fases: pré-processamento, feature extraction, vetorização, treinamento e avaliação dos resultados com dados novos.

Na primeira etapa, os dados foram extraídos do arquivo fornecido em formato XLSX e em seguida foi realizada a normalização dos textos dos campos “message” e “response” com as seguintes atividades: remoção de quebra de linhas de aspas e apóstrofes; e a substituição de tabulações e espaços duplos por um espaço em branco.

A fase seguinte, denominada de feature extraction ou extração de características, compreendeu a análise morfológica e sintática das palavras por meio das tarefas de stemização, lematização, POS Tagger e reconhecimento de entidades nomeadas.

Na fase de vetorização foi utilizada a abordagem TF-IDF que considera a relevância da palavra no texto, complementado com a representação distribuída BERT que melhor caracteriza o sentido das palavras nos seus diversos contextos. Nesta etapa, os dados das sentenças normalizadas junto com os dados produzidos nas tarefas de análise morfológica e sintática das fases anteriores foram transformados em uma matriz de números multidimensional, de forma que os algoritmos de Aprendizagem de Máquina possam processá-los.

A última etapa caracterizou-se pela seleção dos algoritmos, implementação e refinamento dos parâmetros dos modelos para obter uma melhor performance. Foram executados diversos cenários de experimentação utilizando o modelo BERT.

2.2.1. Visualização dos Dados

Como parte do processo de treinamento, três colunas foram consideradas no conjunto de dados - 'entailment_label', 'message’ (premissa) e 'response' (hipótese): o 'entailment_label' é a coluna que indica o rótulo dado ao par de sentenças. Havia três rótulos - 'POSITIVE' e 'NEGATIVE', descrito na tabela 1.

	pair_ID	message	response	entailment_label
0	0	b'have you heard of the upcoming black panther...	b'i have and i am so in love with the trailer ...	POSITIVE
1	1	b'it looks remarkable so far!\r\n'	b'chadwick really is a good actor for black pa...	POSITIVE
2	2	b'i agree!. he really is suitable for the role...	b'the trailer was kind of sad though.\r\n'	POSITIVE
3	3	b'how so? did it bother you when his father di...	b'yes, it did, but i hope he become one of the...	POSITIVE
4	4	b'i second that statement!\r\n'	b'but the movie will be a while until it is re...	POSITIVE

Tabela 1 – Amostra dos Dados

2.2.2. Exploração e Tratamento dos Dados

Como exemplificado anteriormente diversas transformações foram realizadas para transformar o dado e conforme descrito na tabela 2 abaixo.

	response_em_sentencas	message_em_sentencas	response_tratado	message_tratado	entailment_label	response	message	pair_ID
0	[i have and i am so in love with the trailer a...	[have you heard of the upcoming black panther ...	i have and i am so in love with the trailer al...	have you heard of the upcoming black panther m...	POSITIVE	b'i have and i am so in love with the trailer ...	b'have you heard of the upcoming black panther...	0
1	[chadwick really is a good actor for black pan...	[it looks remarkable so far]	chadwick really is a good actor for black pant...	it looks remarkable so far	POSITIVE	b'chadwick really is a good actor for black pa...	b'it looks remarkable so far\r\n'	1
2	[the trailer was kind of sad though]	[i agree he really is suitable for the role]	the trailer was kind of sad though	i agree he really is suitable for the role	POSITIVE	b'the trailer was kind of sad though.\r\n'	b'i agree!. he really is suitable for the role...	2
3	[yes it did but i hope he become one of the ...	[how so did it bother you when his father died]	yes it did but i hope he become one of the b...	how so did it bother you when his father died	POSITIVE	b'yes, it did, but i hope he become one of the...	b'how so? did it bother you when his father di...	3
4	[but the movie will be a while until it is rel...	[i second that statement]	but the movie will be a while until it is rele...	i second that statement	POSITIVE	b'but the movie will be a while until it is re...	b'i second that statement!\r\n'	4

Tabela 1 – Transformação dos Dados

Durante a fase exploratória percebemos em nosso corpus o resultado abaixo:

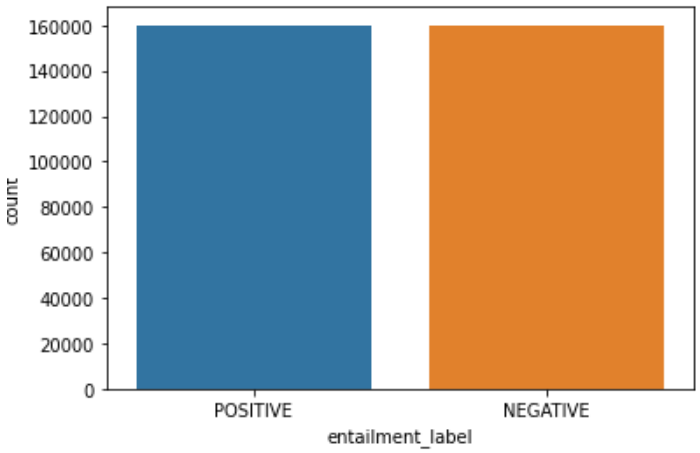

Figura 1 – Plotagem da Amostra

Dividimos o corpus em dados de treinamento e validação na escala de 70% treinamento e 30% validação, conforme abaixo:

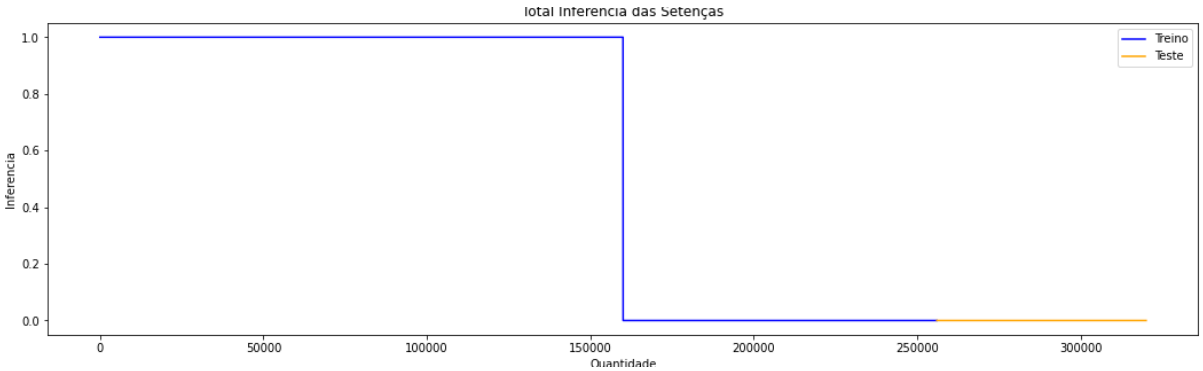

Figura 2 – Amostra Treinamento e Validação

3. Avaliação e Resultados

3.1. BERT

BERT (Bidirectional Encoder Representations from Transformers) é um modelo de linguagem do Google baseado no modelo de transformador codificador-decodificador apresentado neste artigo. Ele usa o mecanismo de atenção dos transformadores para aprender o significado contextual das palavras e as relações entre elas. O BERT, junto com suas modificações como ALBERT , RoBERTa , etc., são conhecidos por alcançar resultados de ponta em várias tarefas de processo de linguagem natural, como responder a perguntas e inferência em linguagem natural.

O nosso artigo trabalha com o conceito de vinculação, para entendermos melhor vamos começar com um exemplo:

1. Jim vai de bicicleta para a escola todas as manhãs.

2. Jim pode andar de bicicleta.

A restrição ocorre se uma premissa proposta for verdadeira. Neste exemplo, se a frase 'Jim vai de bicicleta para a escola todas as manhãs'. for verdade, então a premissa implica que Jim vai para a escola todas as manhãs e que também sabe andar de bicicleta. Consequentemente, isso tornaria a segunda frase, ou a hipótese, verdadeira também.

Para definir a implicação em termos simples, diz-se que uma frase Y implica a frase X se X for verdadeiro e Y puder ser logicamente derivado dela. Para o conjunto de dados que usei, um par de sentenças pode envolver uma à outra, ser neutras ou se contradizerem.

Para medir a similaridade entre duas sentenças com BERT, devemos concatená-las com um token [SEP] no meio e alimentar essa sequência por meio de BERT. Podemos então, na outra extremidade, usar o token [CLS] como entrada para um classificador ou regressor simples para nos dizer como essas sentenças estão relacionadas.

Em nossos testes pudemos perceber que o BERT para comparar frases é possível, mas muito lento para aplicações em tempo real. O principal culpado é que o BERT precisa processar as duas sentenças em uma para medir a similaridade.

3.1.1. Ajustando o BERT

A primeira etapa envolveu a criação de um objeto DataLoader para alimentar o modelo com dados. O BERT para classificação de sequência requer que os dados sejam organizados em um determinado formato. O início de cada frase precisa ter um token [CLS] presente e o final da frase precisa de um token [SEP]. Portanto, com nossa sequência consistindo em duas sentenças, ela precisará ser formatada como [CLS] frase 1 [SEP] frase 2 [SEP]. Além disso, cada sequência precisará ter segment_ids associados a ela. A primeira frase na sequência é marcada por [0], enquanto a segunda frase é marcada por [1]. Por último, cada sequência precisa de uma máscara de atenção para ajudar o modelo a determinar qual parte da sequência de entrada não faz parte do preenchimento.

Agora que os objetos DataLoader para os conjuntos de treinamento e validação foram criados, o modelo pode ser carregado junto com seu otimizador. Para este caso, usarei o modelo pré-treinado BertForSequenceClassification. Este modelo oferece um argumento adicional para adicionar um cabeçote de classificação opcional com o número necessário de rótulos. Para este caso, existem três classes. Portanto, eu defino num_labels como três. Isso adiciona uma cabeça de classificação com três unidades de saída como a camada final.

Com os loops de treinamento e validação definidos, podemos ajustar o modelo no conjunto de dados MultiNLI para tentar alcançar o desempenho esperado, como visto pelos valores de perda e precisão abaixo, o modelo parece estar aprendendo enquanto se ajusta um pouco mais.

```
train(model, train_loader, val_loader, optimizer)
```

```
Epoch 1: train_loss: 0.7893 train_acc: 0.6561 | val_loss: 0.5867 val_acc: 0.7534
00:44:13.53
Epoch 2: train_loss: 0.5799 train_acc: 0.7646 | val_loss: 0.5670 val_acc: 0.7556
00:44:36.48
Epoch 3: train_loss: 0.4946 train_acc: 0.8101 | val_loss: 0.5814 val_acc: 0.7590
00:45:22.09
Epoch 4: train_loss: 0.4288 train_acc: 0.8358 | val_loss: 0.6204 val_acc: 0.7663
00:46:05.01
Epoch 5: train_loss: 0.3739 train_acc: 0.8648 | val_loss: 0.6293 val_acc: 0.7652
00:45:42.85
```

Figura 3 – Resultado do BERT

3.2. Considerações Finais

Ao final foi possível criar uma função para pegar as predições e assim realizar perguntas para obter as respostas a partir das predições, o trabalho pode evoluir facilmente para criar um API para consultar o modelo salvo com uma aplicação real de chatbot a partir do modelo treinado.

```
[ ]  def get_prediction(str):
       str = re.sub(r'[^a-zA-Z ]+', '', str)
       test_text = [str]
       model.eval()

       tokens_test_data = mnli_dataset.tokenizer(
       test_text,
       pad_to_max_length=True,
       truncation=True,
       return_token_type_ids=False
       )
       test_seq = torch.tensor(tokens_test_data['input_ids'])
       test_mask = torch.tensor(tokens_test_data['attention_mask'])

       preds = None
       with torch.no_grad():
         preds = model(test_seq.to(device), test_mask.to(device))

       preds = np.argmax(preds)
       print("Intent Identified: ", le.inverse_transform(preds)[0])
       return le.inverse_transform(preds)[0]

     def get_response(message):
       intent = get_prediction(message)
       for i in data['intents']:
         if i["tag"] == intent:
           result = random.choice(i["responses"])
           break
       print(f"Response : {result}")
       return "Intent: "+ intent + '\n' + "Response: " + result
```

```
[ ]  get_response("why dont you introduce yourself")
```

```
[ ]  predictions = model.predict(x_test)
```

Figura 4 – Função Chatbot

4. Referências

A. Natekin and A. Knoll, “Gradient boosting machines, a tutorial,”Frontiers inneurorobotics, vol. 7, p. 21, 2013.19.

G. James, D. Witten, T. Hastie, and R. Tibshirani,An introduction to statisticallearning. Springer, 2013, vol. 112.20.

A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference Adina Williams, Nikita Nangia, Samuel Bowman, 2018