Avaliação Final – NLP

Objetivo: elaborar, desenvolver, avaliar e entregar um produto de dados baseado em IA e NLP.

Parte 1: definir produto e buscar base de dados

1) Defina um produto de IA/NLP baseado em dados (não pode ser um chatbot).

a) Qual é o problema que esse produto resolve?

R:

Na atual startup em que eu trabalho, é feita a venda de dados sobre restaurantes nos EUA. Essas informações podem ajudar outras empresas a tomarem decisões estratégicas.

Imagine que você tem uma empresa que fornece os sistemas para pagamento ou controle dos pedidos feitos em um restaurante.

É interessante para você saber informações sobre potenciais clientes para fazer um marketing assertivo.

Por exemplo, os endereços das lojas é uma informação que sendo obtida em larga escala é extremamente importante (exemplo, imagine que de uma rede de lojas X, você já fornece produto para 3 filiais, mas você não sabia que existiam outras 2 filiais. Ao saber que essas outras filiais existem, você pode aumentar a prospecção de novos clientes, de forma mais facilitada)

b) Qual é o público a que o produto atende?

R:

Empresas que atuam como fornecedores de algum serviço/produto para restaurantes nos EUA.

c) Como esse produto se relaciona a machine learning?

R:

Com o objetivo de obter endereço de lojas, você pode elaborar um crawler que acessa o website dessas lojas, porém a informação extraída está desestruturada.

Para distinguir strings que representam a localização das lojas, de outras strings genéricas, pode-se utilizar um modelo de ML para aprender a fazer essa distinção.

2) Escolha uma base de dados (maior que a memória do seu computador – se precisar, faça webscrapping ou use APIs!)

a) Baixe a base de dados

R: Como fonte de endereços, eu como baixei os dados a partir daqui

- openaddresses

Como fonte para textos que não representam endereços, eu baixe uma base de tweets genéricos:

- Kaggle

b) Organize os dados de forma que possam ser lidos em minibatches pela API do Keras:

R: Nessa etapa eu falhei, não utilizei a API do Keras para ler minibatches

c) Defina qual (ou quais) métrica (s) poderiam ser usadas na sua base de dados para avaliar se IA e NLP são soluções viáveis para a realização do seu produto?

R: O mais importante é validar se o modelo consegue classificar com uma alta taxa de assertividade endereços e textos genéricos.

Além disso a solução precisa ser escalável (baixa latência)

Parte 2: experimentar estratégias de machine learning

1) Escolha algumas estratégias de machine learning para testar, incluindo, no mínimo:

a) Uma abordagem tradicional “baseline” (por exemplo, as disponíveis no sklearn)

R: Feito, desenvolvi um modelo com Regressão Logística

b) Uma abordagem com Deep Learning treinada integralmente in-house

R: Feito, desenvolvi um modelo com LSTM

c) Uma abordagem com Deep Learning que usa redes pré-treinadas para alimentar uma rede neural treinada in-house

R: Feito, utilizei um BERT pré treinado no lugar da LSTM

d) Uma abordagem com Deep Learning que usa integralmente uma rede pré-treinada, com o mínimo de pós-processamento (não precisa ser a mesma rede do ítem (c))

R: Feito, utilizei o DeepParse: https://github.com/GRAAL-Research/deepparse

2) Compare o desempenho de cada uma das estratégias escolhidas. Para isso:

a) Treine cada uma das redes in-house várias vezes, e encontre um intervalo de confiança de 95% para o desempenho.

R: Feito

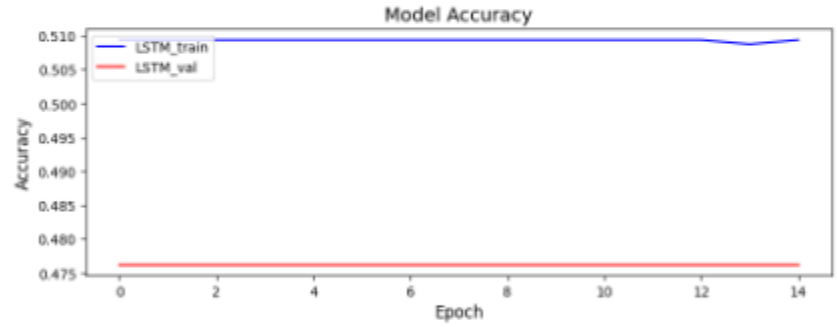
b) Compare o desempenho das quatro abordagens que você testou. Reporte o desempenho em um gráfico de barras com intervalos de confiança.

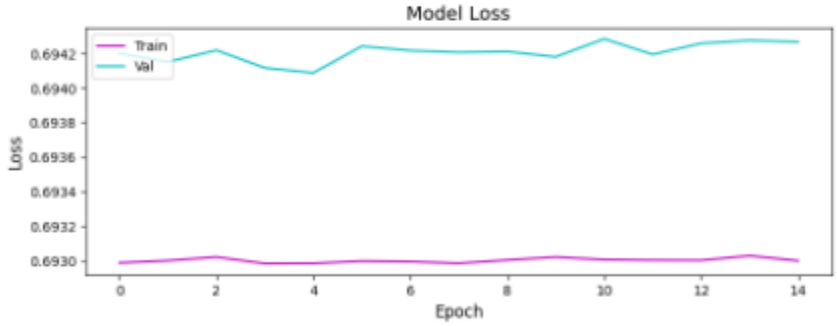
R: Feito

AUC Roc Curve: Baseline 0.9998801797200192

AUC Roc Curse: LSTM: 0.999100673378908

Sobre utilizar a rede-pré treinada com BERT, ela não convergiu:





Utilizando Integralmente uma rede pré-treinada

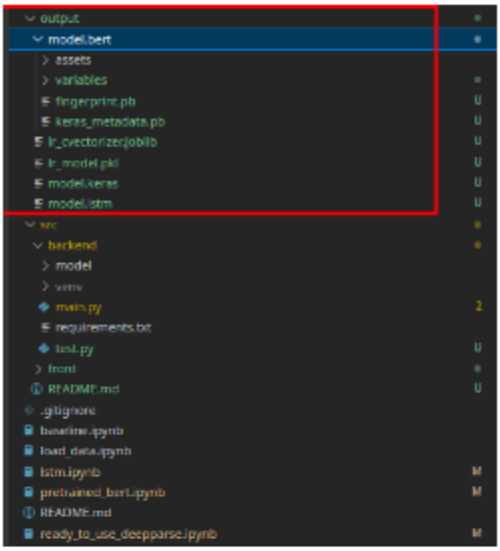
AUC Roc Curve: 0.9561477896450734

c) Encontre casos típicos de erro para cada um dos sistemas treinados.

R: Todos os modelos funcionam muito bem, com exceção do BERT que não deu certo.
O único detalhe, é que os dados de Endereços utilizados para treinamento são no padrão dos EUA, qualquer endereço de outro país não funciona.

d) Salve os modelos treinados

R: Feito, não consegui subir no Git por causa do tamanho, ficava travado o push.



Parte 3: Avalie o custo computacional de inferência dos sistemas

a) Avalie o tempo que cada um dos sistemas demora para responder a uma chamada de inferência (predict).

Baseline:
72.5 µs ± 132 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)

LSTM:
36.4 ms ± 852 µs per loop (mean ± std. dev. of 7 runs, 1 loop each)

Utilizando Integralmente uma rede pré-treinada
2.07 ms ± 32.4 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

b) Compare o tamanho dos modelos e a memória RAM usada por cada um deles ao ser carregado para a memória.

Baseline:
Quase nada GB

LSTM:
Mais que 1GB

Utilizando Integralmente uma rede pré-treinada
Menos que 1GB

c) Levando em consideração as possibilidades de usar VMs gratuitas na Oracle, ou pagas na AWS, decida qual dos modelos é mais adequado para a sua aplicação.

Levando em conta que todos os modelos que deram certo obtiveram uma performance muito parecida (aproximadamente 99% de acurácia), o que mais faz sentido (tanto em questão de utilização de recursos, quanto de tempo de processamento) é o modelo baseline (pois não tem necessidade de GPU, não precisa de instalações complexas (apenas um

ambiente virtual do Python) ocupa pouco espaço, e é rápida.
Embora seja possível rodar esse modelo com uma API dentro da VM gratuita da Oracle, não faz tanto sentido para o destino final: Processamento de grandes quantidades de dados (análise de milhões de possíveis endereços em um curto espaço de tempo)
Dessa forma, a melhor opção seria uma VM paga, com bom custo benefício, alta disponibilidade de rede, e vários núcleos de CPU para poder ser criar uma API que consegue processar várias operações em paralelo.

Parte 4: Faça o deploy do sistema para uma API e uma demo

a) Suba um pequeno servidor usando FastAPI, Django ou a tecnologia a sua escolha. Através de uma API REST, o usuário deve ser capaz de enviar dados ao sistema e receber uma resposta.
Foi feito o deploy dentro da AWS. Um simples o backend com FastAPI expondo o modelo baseline.

b) Faça uma página HTML/Javascript que permita acessar facilmente uma demonstração da API através da web.
Foi feito o deploy dentro da AWS. Um simples front de uma página com HTML/CSS + Javascript.

O resultado final pode ser observado no seguinte vídeo demonstração:
https://www.youtube.com/watch?v=9iAsMAmqN_s

Código fonte:
https://github.com/delattre1/Insper-PF-NLP