Luan de Tarso Pereira Azevedo e Isaías Lucena

Diego da Silva Rodrigues

Deep Learning com TensorFlow

26 de setembro de 2023

CLASSIFICAÇÃO DE TEXTOS EM PORTUGUÊS PARA CHATBOT

Este trabalho apresenta como foi feito o desenvolvimento de um Jupyter Notebook (ipynb) para a classificação de textos em português, para posteriormente ser utilizado em um chatbot. A maior parte do desenvolvimento se concentra na aplicação de técnicas avançadas de diversos métodos de classificação e regressão, como o “Decision Tree Classifier” e a “Logistic Regression Classifier”, a fim de comparar qual obteve a melhor acurácia.

BASE DE DADOS

A base de dados utilizada para está em formato CSV e é composta por duas colunas e 577 linhas, sendo elas "Patterns" e "Tags" A coluna "Patterns" contém uma série de mensagens que os usuários podem enviar ao ChatBot durante suas interações. Estas mensagens representam consultas, perguntas e solicitações típicas que os clientes da empresa fictícia podem fazer. Cada mensagem nesta coluna serve como entrada para o modelo de ChatBot, e sua variedade reflete a diversidade de interações possíveis entre os usuários e o sistema. Exemplos de mensagens nesta coluna incluem:

- "Boa tarde, gostaria de obter informações"
- "Vocês oferecem serviço de entrega de produtos?"
- "Como posso entrar em contato com vocês?"
- "Quais são os produtos disponíveis para compra?"
- "Qual é o prazo de entrega padrão?"
- "Como faço para agendar uma instalação de ar condicionado?"

A coluna "Tags" é igualmente importante, pois atribui categorias ou rótulos às mensagens presentes na coluna "Patterns". Cada categoria representa um contexto ou uma intenção por trás da mensagem do usuário. Por exemplo, mensagens como "Boa tarde, gostaria de obter informações" e "Como posso entrar em contato com vocês?" podem ser categorizadas como "Cumprimento" e "Contato", respectivamente. A atribuição de tags às mensagens permite que o ChatBot compreenda o propósito da consulta do usuário e forneça respostas relevantes com base na categoria associada.

DESENVOLVIMENTO DO IPYNB

Preparação dos Dados

Para o desenvolvimento do classificador foram utilizadas uma série de bibliotecas, como *pandas, sklearn* e *nltk.* Após o *load* da base de dados no notebook é necessário realizar a "limpeza" dos dados textuais, pois assim é possível fazer a eliminação de ruído, como remoção de números e caracteres especiais, a padronização das palavras, ou seja, eliminar variações de uma mesma palavra e até mesmo correção de erros ortográficos. Para isso foi feito o uso da biblioteca *NLTK* para se obter *stopwords* em português, *Spacy* e *WordNetLemmatizer* para a lematização e criação de um documento *spacy* (*doc*).

Após, com o *doc* já criado e os textos limpos, foi feita a vetorização das *patterns* com o *TF-IDF*. O *TF-IDF* é uma técnica que atribui um valor numérico a cada palavra (ou token) em um conjunto de documentos com base em sua importância relativa dentro de um documento específico e em toda a coleção de documentos.

A última etapa antes do treinamento é fazer a divisão da base de dados, onde 20% será utilizada como teste e o restante como treinamento.

Treinamento e Resultados

1. Gaussian Naive Bayes

O primeiro método utilizado foi o Gaussian Naive Bayes, que é um algoritmo de aprendizado de máquina supervisionado com base no Teorema de Bayes e é uma variação do classificador Naive Bayes. O Naive Bayes é usado principalmente para tarefas de classificação, como categorização de texto, diagnóstico médico, detecção de spam e muito mais. A abordagem "Naive" (ingênuo) pressupõe que os recursos (ou atributos) utilizados para a classificação sejam independentes, o que simplifica o cálculo das probabilidades.

No caso do Gaussian Naive Bayes, a suposição ingênua é que os valores dos recursos seguem uma distribuição gaussiana (normal). Isso significa que, para cada classe, o modelo calcula a média e o desvio padrão de cada atributo nos exemplos de treinamento pertencentes a essa classe.

Abaixo vemos os resultados deste método:

- Training Accuracy score: 0.9826086956521739
- Testing Accuracy score: 0.7586206896551724

2. Multinomial Naive Bayes

Em seguida utilizamos o método Multinomial Naive Bayes. O Multinomial Naive Bayes é uma variação do algoritmo Naive Bayes que é frequentemente usado para tarefas de processamento de texto, como classificação de documentos de texto, análise de sentimentos e detecção de spam.

A principal característica do Multinomial Naive Bayes é que ele é adequado para dados discretos, como contagens de palavras em documentos. Ele assume que os atributos (ou recursos) são representados por contagens inteiros não negativos, geralmente indicando a frequência de ocorrência de palavras em um documento.

Abaixo vemos os resultados deste método:

- Training Accuracy score: 0.691304347826087
- Testing Accuracy score: 0.5517241379310345

3. Logistic Regression Classifier

Os classificadores de Regressão Logística são um tipo de algoritmo de aprendizado de máquina usado para tarefas de classificação binária e, quando estendidos, também podem ser usados para classificação multiclasse. Esse método funciona mapeando uma combinação linear das características de entrada para uma função logística, que produz uma saída entre 0 e 1, representando a probabilidade de um exemplo pertencer à classe positiva.

A Regressão Logística é particularmente útil quando se lida com problemas de classificação simples, onde o objetivo é separar duas classes com um limite de decisão linear. Durante o treinamento, o modelo ajusta os pesos das características de entrada para minimizar a

diferença entre as probabilidades calculadas e as classes reais dos exemplos de treinamento. O limite de decisão é então determinado pela probabilidade de 0,5, onde as instâncias com uma probabilidade maior são atribuídas à classe positiva e as demais à classe negativa.

Esse método é amplamente utilizado em problemas de classificação, interpretação de resultados e é uma base para muitos outros algoritmos de aprendizado de máquina, tornando-o uma ferramenta versátil para uma variedade de aplicações, como detecção de spam, diagnóstico médico e previsões financeiras.

Abaixo vemos os resultados deste método:

- Training Accuracy score: 0.8913043478260869
- Testing Accuracy score: 0.6982758620689655

4. Support Vector Machines

Support Vector Machines (SVM) é um algoritmo de aprendizado de máquina usado principalmente para tarefas de classificação e também para regressão. O objetivo do SVM é encontrar um hiperplano de separação que maximize a margem entre as classes de dados. A margem é a distância entre o hiperplano e os pontos mais próximos de cada classe, chamados de vetores de suporte.

O SVM é eficaz para lidar com dados de alta dimensionalidade e é especialmente útil quando há uma separação não linear entre as classes. Para isso, ele usa transformações não lineares dos dados para projetá-los em um espaço onde a separação linear seja possível. Isso é conhecido como o truque do kernel.

Os SVMs são robustos e têm bom desempenho em muitos tipos de problemas de aprendizado de máquina, incluindo classificação de imagem, reconhecimento de padrões, e problemas de detecção, tornando-os uma ferramenta poderosa na caixa de ferramentas de machine learning.

Abaixo vemos os resultados deste método. É possível notar que a acurácia teve um aumento significativo em relação ao método anterior:

- Training Accuracy score: 0.991304347826087
- Testing Accuracy score: 0.896551724137931

5. Decision Tree Classifier

O Decision Tree Classifier (Classificador de Árvore de Decisão) é um algoritmo de aprendizado de máquina usado para tarefas de classificação. Ele cria uma representação visual em forma de árvore das decisões com base nas características dos dados. Cada nó na árvore representa uma característica, cada ramificação representa uma escolha ou teste com base nessa característica, e as folhas da árvore contêm as decisões de classificação.

O algoritmo toma decisões dividindo o conjunto de dados em subconjuntos menores com base nas características mais importantes. O objetivo é criar uma árvore que tome decisões precisas e generalizáveis sobre novos dados não vistos. As árvores de decisão são amplamente utilizadas devido à sua interpretabilidade e capacidade de lidar com dados numéricos e categóricos. No entanto, elas podem ser propensas a overfitting em conjuntos de dados complexos e profundos.

Abaixo vemos os resultados deste método:

- Training Accuracy score: 0.9978260869565218
- Testing Accuracy score: 0.7241379310344828

6. Ensembling

Ensembling é uma técnica de aprendizado de máquina que envolve a combinação de múltiplos modelos de previsão para melhorar o desempenho geral. Isso é feito tomando as decisões de vários modelos e agregando-as para obter uma previsão mais precisa e robusta. Existem duas abordagens comuns para ensembling:

Bagging (Bootstrap Aggregating): Nesse método, vários modelos são treinados independentemente em subconjuntos aleatórios do conjunto de dados original. As previsões de cada modelo são então combinadas por votação ou média para chegar a uma previsão final. O Random Forest é um exemplo famoso de algoritmo de bagging.

Boosting: O boosting é uma técnica onde vários modelos fracos são treinados sequencialmente, e o foco está em corrigir os erros cometidos pelos modelos anteriores. Os modelos subsequentes são treinados dando mais peso aos exemplos classificados erroneamente pelos modelos anteriores. O AdaBoost e o Gradient Boosting são exemplos de algoritmos de boosting.

Ensembling é amplamente utilizado em competições de ciência de dados e é conhecido por melhorar o desempenho e a robustez dos modelos, reduzindo o risco de overfitting e aumentando a capacidade de generalização. Para esse método foram juntados os classificadores “Decision Tree”, “Logistic Regression” e “Support Vector Machines”

Abaixo vemos os resultados deste método:

- Training Accuracy score: 0.9956521739130435
- Testing Accuracy score: 0.853448275862069

CONCLUSÃO

Com base nos resultados obtidos após a aplicação de diferentes métodos de classificação, é possível tirar algumas conclusões importantes:

O Support Vector Machines (SVM) obteve a melhor acurácia entre todos os métodos testados, com uma pontuação de teste de 89,66%. Isso indica que o SVM é uma escolha sólida para a classificação de textos em português para um chatbot, oferecendo alta precisão na previsão das categorias das mensagens dos usuários.

O Gaussian Naive Bayes, embora tenha alcançado uma alta precisão no treinamento (98,26%), apresentou uma queda significativa na precisão do teste (75,86%). Isso sugere que o modelo pode ter sofrido overfitting, ou seja, ele se ajustou muito bem aos dados de treinamento, mas não generalizou bem para novos dados.

O Decision Tree Classifier teve um desempenho aceitável, mas não tão alto quanto o SVM, com uma precisão de teste de 72,41%. No entanto, este modelo pode ser propenso a overfitting, como indicado pela diferença entre as pontuações de treinamento e teste.

A abordagem de ensembling, que combina os classificadores Decision Tree, Logistic Regression e Support Vector Machines, também apresentou um desempenho sólido, com uma precisão de teste de 85,34%. Isso demonstra a eficácia da técnica de ensembling em melhorar o desempenho geral do modelo.

Em conclusão, o SVM se destacou como o método mais eficaz para a classificação de textos em português para um chatbot, oferecendo alta precisão na identificação das categorias das mensagens dos usuários. No entanto, a abordagem de ensembling também se mostrou promissora, combinando vários modelos para obter resultados sólidos. É importante considerar a escolha do método com base nas necessidades específicas do projeto e na disponibilidade de dados de treinamento.