

Documentação PLN - Módulo 6 - Inteli

Nexus

Ana Carolina Cremonezi Martire, Daniel Zular, Davi Rosalino Glória Motta, Enzo Boccia Pagliara, Erik Batista da Silva, Lucas da Silva Barbosa, Paulo Octavio de Paula



Índice

- 1. Introdução
- 2. Problema
 - 2.1. Proposta de Solução
 - 2.2. Justificativa
- 3. Objetivos
- 4. Análise de Negócios
 - 4.1. Matriz de Avaliação de Valor Oceano Azul
 - 4.2. Análise Financeira do Projeto
 - 4.3. Value Proposition Canvas
 - 4.4. Matriz de Riscos
- 5. Análise de Experiência do Usuário
 - 5.1. Personas
 - 5.2. User Stories
- 6. Análise Descritiva
- 7. Documentação do Pré-Processamento
 - 7.1 Segmentação
 - 7.2 Tokenização
 - 7.3 Stop-Words
 - 7.4 Lematização
 - 7.5 NER
- 8. Vetorização
 - 8.1. Bag of Words
 - 8.1.1. Introdução

- 8.1.2. Método
- 8.1.3. Resultados
- 8.1.4. Conclusão
- 8.2. Word2Vec
 - 8.2.1. Introdução
 - 8.2.2. Método
 - 8.2.3. Resultados
 - 8.2.4. Conclusão
- 9. Modelo Escolhido
- 10. API
 - 10.1. Rotas de Limpeza
 - 10.2. Rota de Vetorização
 - 10.3. Rota de Classificação
 - 10.4. Rota Geral
 - 10.5. Testes das rotas
 - 10.5.1. Teste 1
 - 10.5.2. Teste 2
 - 10.5.3. Teste 3
 - 10.5.4. Teste 4
 - 10.6. Conclusão
- 11. Arquitetura Macro
- 12. Diagrama
- Referências

1. Introdução

A Uber, multinacional americana que presta serviços eletrônicos na área de transporte, firmou uma parceria com o Inteli para a criação de uma solução que aprimore a conexão da Uber com seus usuários. A Nexus, grupo composto por 7 estudantes do Inteli, criou tal solução. O projeto se concentra em desenvolver uma aplicação de processamento de linguagem natural (PLN) para mensurar os sentimentos dos clientes em relação à empresa. Com esta ferramenta, a Uber pretende aprimorar sua gestão de redes sociais, além de deixar de terceirizar a análise dos sentimentos de seus usuários.

2. Problema

2.1. Proposta de Solução

A solução proposta para a Uber pelo grupo Nexus consiste em desenvolver um modelo de processamento de linguagem natural que será um "Termômetro de Sentimentos integrado com o Slack". Este modelo irá analisar e interpretar os sentimentos expressos nos comentários dos usuários em relação a Uber, mais especificamente na rede social X. Utilizando algoritmos de machine learning, a aplicação classifica os comentários em sentimentos positivos, negativos ou neutros, e identifica padrões para uma futura ação por parte da Uber.

Além da detecção de sentimentos, o sistema também será configurado para gerar alertas automáticos em situações de potencial crise. Isso será feito por meio de um aviso na plataforma Slack, onde a Uber poderá visualizar os sentimentos de seus usuários naquele momento. Esses alertas permitirão à equipe da Uber reagir de acordo com a visão do público. A expectativa é que, com a implementação deste sistema, a Uber não apenas melhore o entendimento do que seus usuários sentem, mas sim promova uma maior satisfação do cliente e uma melhor percepção da marca.

2.2. Justificativa

O desenvolvimento desta aplicação se dá pela necessidade da Uber em ter melhor conhecimento das opiniões de seus usuários. Atualmente, a empresa terceiriza a análise dos comentários das redes sociais, que apresenta um problema de eficiência, visto que este processo é feito manualmente e em um prazo mensal. A implementação do projeto permitirá uma análise rápida e eficiente de grandes volumes de dados, o que facilita a identificação de padrões e tendências que antecedem crises, reduzindo assim riscos e melhorando a satisfação geral do usuário final da Uber. Este projeto ajuda a Uber a alcançar seus objetivos de fortalecer a marca e gerar satisfação aos seus clientes ao ouvi-los, o que também fortalece a relação entre a Uber e seus clientes.

3. Objetivos

Tendo em vista o problema apresentado, a equipe de desenvolvimento, em colaboração com a equipe de Gestão de Redes Sociais da Uber, propôs o desenvolvimento de uma Inteligência Artificial (IA) utilizando processamento de linguagem natural (PLN). Este sistema visa realizar o monitoramento e análise contínua dos sentimentos expressos pelos usuários da Uber nas redes sociais, mais especificamente no Twitter.

O objetivo principal da solução de PLN é permitir um rastreamento em tempo real, análise e interpretação das mensagens e comentários postados pelos usuários. Com esta capacidade, a aplicação tem como meta identificar as percepções e emoções dos clientes de forma precisa e eficiente. Implementada por meio de algoritmos de aprendizado de máquina, a solução é treinada para detectar e classificar sentimentos e palavras-chave nos comentários, permitindo uma resposta mais ágil e informada por parte da Uber em relação às necessidades e expectativas de seus usuários. Através desta iniciativa, a Uber busca aprimorar sua gestão de redes sociais e fortalecer sua relação com os clientes, contribuindo para uma percepção positiva da marca e a satisfação geral dos usuários.

4. Análise de Negócios

4.1. Matriz de Avaliação de Valor Oceano Azul

A Matriz de Avaliação de Valor de Oceano Azul é uma ferramenta baseada na Estratégia de Oceano Azul, apresentada no livro homônimo escrito por W. Chan Kim e Renée Mauborgne, representantes da escola de negócios INSEAD. Esta possui como principal objetivo ajudar a traçar estratégias capazes de levar o negócio a um ambiente onde não há uma guerra constante contra a concorrência devido ao diferencial de valor entregue ao cliente. Desta forma, o objetivo é realizar uma avaliação da Uber, a empresa para a qual este projeto está sendo conduzido.

Para isso, elencamos os principais atributos da empresa que estamos avaliando e estimamos o quanto desenvolvida ela e suas respectivas concorrentes estão em relação a cada atributo. Isso possibilita a visualização do ambiente no qual a empresa está inserida e a elaboração de passos futuros. Estes atributos são:

Aumentar:

- Privacidade dos Dados;
- Eficiência.

Reduzir:

- Preço;
- Sugestão de Ações.

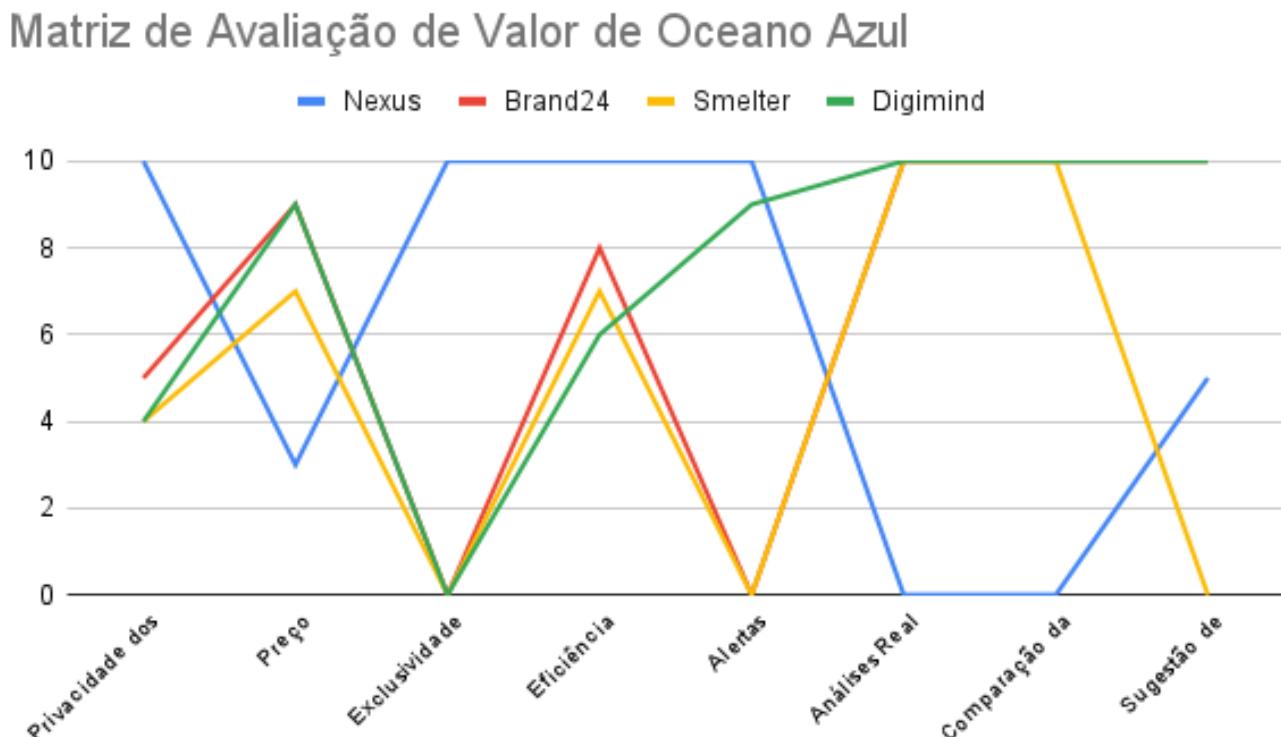
Criar:

- Exclusividade;
- Alertas Personalizados.

Eliminar:

- Análises em Tempo Real;
- Comparação da marca com concorrentes.

Figura 1 - Matriz de Avaliação de Valor de Oceano Azul



Fonte: Material produzido pelos autores (2024)

É válido ressaltar que, por limitações da ferramenta utilizada, não é possível visualizar o nome completo dos atributos no gráfico. No entanto tal informação está disponibilizada acima do mesmo.

Tais atributos foram definidos tendo como base os serviços e produtos oferecidos pela Nexus e outras plataformas de Processamento de Linguagem Natural em redes sociais. No que diz respeito aos atributos que devem ser aumentados, foram elencados a "Privacidade de Dados" e a "Eficiência". No que tange à "Privacidade de Dados", a Nexus traz isso como destaque quando comparada aos concorrentes, visto que ela gera um modelo que se tornará propriedade do cliente, fazendo com que este não precise ceder nenhum tipo de dado de suas bases para a contratada. Tal fator não ocorre com as outras empresas, visto que nestas o cliente precisa usar plataformas externas ao seu ecossistema, onde toda a atividade da marca e insights, ou seja, seus dados, serão exibidos e estarão sob sua posse.

Em relação à "Eficiência", a Nexus realiza análises totalmente automatizadas, as quais são disponibilizadas no formato desejado pelo cliente. Tal fator acarreta em uma maior "Eficiência" em processos para o cliente, dado que este pode receber os outputs das análises no formato e plataforma desejados. Em contrapartida, outras empresas oferecem informações apenas em suas próprias plataformas e formatos, o que faz o cliente gastar mais tempo, pois ele terá que organizar essas informações em um formato adequado para comunicações da empresa e depois compartilhá-las entre as equipes.

No que diz respeito a criar, a Nexus traz "Exclusividade" à Uber, dado que esta possuirá uma solução que ainda não é utilizada no mercado por nenhuma outra empresa, tornando-se então a única a desfrutar de seus benefícios. Além disso, tal exclusividade, alinhado ao acompanhamento do desenvolvimento do projeto por parte da Uber, faz com que esta tenha uma solução que atende em detalhes suas dores. Tal fato é diferente daquilo que ocorre com as outras empresas, as quais oferecem o mesmo produto para todos os seus clientes, o que evidencia a falta de exclusividade.

O outro atributo relacionado a criar é a existência de "Alertas Personalizados", dado que os outputs (alertas) gerados pelo modelo do grupo Nexus possuem um formato totalmente moldado pelo cliente, o que faz com que estes sejam muito mais orientados às suas reais dores. No entanto, as outras empresas não possuem tal atributo, dado que oferecem apenas em suas respectivas dashboards as análises dos dados coletados - ou ainda análises integradas com outras plataformas mas tendo formatos pré-definidos, muitas das vezes não atendendo da melhor forma às dores existentes.

Além disso, no que diz respeito aos atributos que serão reduzidos, pode-se citar o "Preço", o qual, com a solução Nexus, é significativamente menor, dado que é uma solução de código aberto e os únicos custos atrelados a sua utilização são os de implementação de sistemas pelo próprio cliente. Assim, para utilizar a solução das outras empresas, este precisará não só arcar com os custos supracitados, como também necessariamente pagará uma taxa de uso das plataformas, podendo estas ainda variar de acordo com a quantidade de dados utilizados para gerar os insights.

Além do preço, outro atributo reduzido foi "Sugestão de Ações". Isto ocorre, pois a Nexus não se propõe a sugerir ações detalhadas, apesar de os alertas cumprirem parte desta ação ao trazerem ciência sobre pontos críticos a serem tratados, consequentemente sugerindo que uma ação seja tomada tendo como base os alertas. No entanto, concorrentes como a Brand24 e Digimind são capazes de gerar mais sugestões de ações ao utilizar IA Generativa na análise dos dados obtidos.

Ademais, alguns atributos importantes, como as 'Análises em Tempo Real', foram temporariamente eliminados no MVP da Nexus. Essa função é fundamental para monitorar menções em redes sociais de forma instantânea, crucial para uma estratégia proativa de mídia social. A falta dessa capacidade, embora justificada

pela limitação de tempo de desenvolvimento de 10 semanas, é um ponto crítico que difere da oferta dos concorrentes e precisa ser considerada para futuras atualizações a fim de melhorar a competitividade e resposta da Uber no mercado digital.

Da mesma forma, a eliminação da 'Comparação da Marca com Concorrentes' limita a análise estratégica da Uber frente aos seus rivais. Essa capacidade permite que empresas ajustem suas estratégias baseadas em desempenhos comparativos, oferecendo uma visão crítica sobre como a marca está posicionada no mercado. Apesar da decisão inicial de focar na análise do sentimento dos usuários, entender a posição relativa da Uber perante a concorrência é vital para o planejamento estratégico, representando outro aspecto que deve ser reintroduzido para fortalecer as futuras versões do produto.

Deste modo, pode-se concluir que a solução do grupo Nexus traz inúmeras vantagens para a Uber, como uma maior exclusividade, dado que é uma ferramenta nova no mercado, eficiência, pelas informações serem exibidas no lugar e da maneira desejada pela Uber, e por fim preço, dado que se trata de um projeto open source. Também é válido ressaltar que a Nexus ainda possui pontos de desenvolvimento, como a falta de análises em tempo real e uma ferramenta de comparação com concorrentes.

Assim, a Matriz de Oceano Azul não destaca apenas os pontos fortes mas também as áreas-chave para o aprimoramento contínuo da solução. Todas as informações utilizadas nesta análise foram obtidas por meio de pesquisas detalhadas, tendo como fontes os sites oficiais das empresas mencionadas, bem como o próprio time Nexus, responsável pela construção da solução de processamento de linguagem natural aqui apresentada.

4.2. Análise Financeira do Projeto

A análise financeira deste projeto aborda os investimentos necessários para sua implementação e manutenção ao longo de um ano, além de considerar o orçamento disponibilizado pelo parceiro. Consequentemente, procedeu-se à estimativa dos seguintes custos:

Tabela 1 - Custos Estimados

| Item | Descrição | Custo |
|-------------------------|---|--|
| Coleta de dados | Utilização da Twitter API Premium para acesso ampliado aos dados de tweets, permitindo análises mais profundas e abrangentes. | RS: 12.000,00 |
| Supporte Técnico | Refere-se aos custos de infraestrutura na Google Cloud, incluindo duas instâncias do tipo n1-standard-4 ativas 24/7, armazenamento de 50TB e o uso da API Cloud Natural Language. Frequência: Mensal | RS: 6.300,00 (excluindo API de NLP) ou R\$: 16.800,00 (incluindo API de NLP) |
| Recursos Humanos | Envolve uma equipe de três especialistas: um desenvolvedor para o backend, um para o frontend e um cientista de dados. A composição da equipe varia entre níveis pleno e sênior. | RS: 45.000,00 (plenos) ou RS: 75.000,00 (sêniores) |
| Economia por Eficiência | Estimativa da economia gerada pela melhoria em eficiência e gestão de crises, calculada como 15% dos custos operacionais totais. | RS: 96.840,00 a RS: 186.840,00 |

| Item | Descrição | Custo |
|------------------|--|--------------------------------|
| Manutenção Anual | Estimativa dos custos de manutenção e atualizações tecnológicas, considerando 10% do investimento inicial anual. | RS: 64.560,00 a RS: 124.560,00 |
| | | |

Fonte: Material produzido pelos autores (2024)

Lembrando que todos os valores estão sendo calculados anualmente.

Investimento Total Estimado

Considerando o acima, o investimento total estimado para o período de um ano varia de RS 645.600,00 a RS 1.245.600,00, dependendo da estrutura da equipe escolhida e da inclusão da API de Processamento de Linguagem Natural. A economia estimada pelas melhorias em eficiência pode variar de RS 96.840,00 a RS 186.840,00, e os custos de manutenção anual são projetados entre RS 64.560,00 a RS 124.560,00. É importante destacar a possibilidade de otimizar custos através da contratação de um profissional versátil ou negociação de valores.

Análise de Break-even

A análise de break-even indica que, assumindo a economia anual máxima de 15% em custos operacionais, a Uber poderia começar a ver um retorno sobre o investimento em aproximadamente 3,45 anos. Este cálculo baseia-se no cenário de menor investimento (RS 645.600,00) e maior economia anual (RS 186.840,00), considerando a redução de custos e a preservação de receitas como principais fontes de retorno.

Considerações Finais

A implementação desse sistema, apesar de não resultar diretamente em receita, tem o potencial de fornecer insights valiosos para a antecipação de tendências negativas e a identificação precoce de aspectos críticos na percepção de usuários e parceiros, evitando possíveis crises. Além disso, as melhorias em eficiência e gestão de crises podem gerar economias significativas, fortalecendo o caso de investimento.

Fontes

- [Twitter API](#) - Coleta de dados
- [Google Cloud](#) - Suporte técnico
- [Glassdoor](#) - Recursos humanos

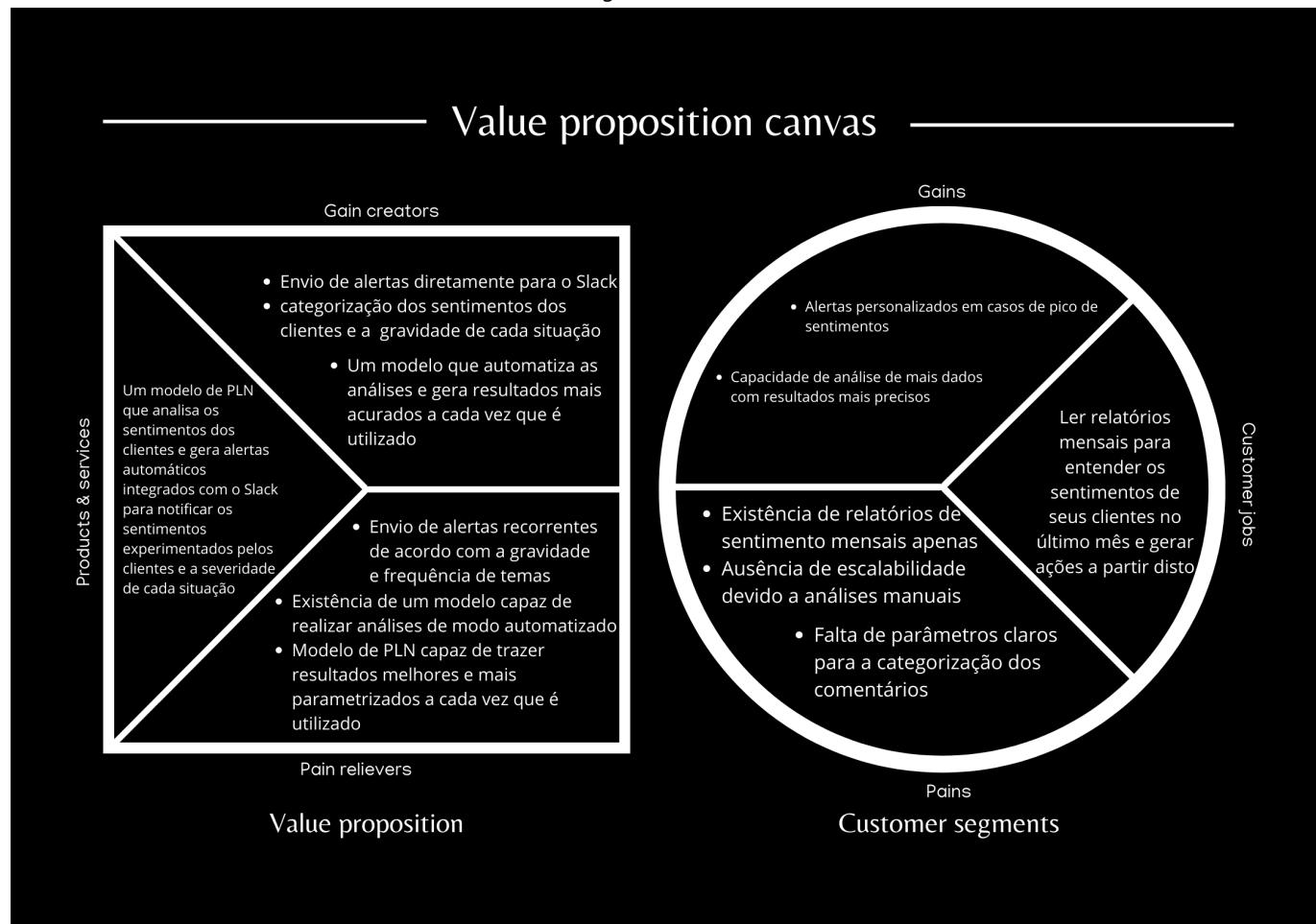
4.3. Value Proposition Canvas

O projeto "Nexus", em colaboração com a Uber, utiliza a metodologia do Value Proposition Canvas (VPC) para mapear e testar a proposta de valor da solução. Segundo o grupo G4 educação, o VPC é uma ferramenta que permite aos empreendedores desenhar, testar e visualizar o valor do produto para os clientes de forma intuitiva. A relevância dessa abordagem para o projeto "Nexus" reside na sua capacidade de identificar claramente as expectativas e necessidades dos usuários da Uber, transformando esses insights em características tangíveis do produto.

Ao lado direito da imagem do VPC abaixo, há o perfil do cliente, que detalha os ganhos esperados, as dores atuais e as tarefas relacionadas ao uso do produto. Ao lado esquerdo, o mapa de valor mostra como o

produto proposto atende a essas necessidades, destacando seus criadores de ganho e o alívio das dores do cliente.

Figura 2 - Título



Fonte: Material produzido pelos autores (2024)

Produtos e Serviços (Products & Services):

- Um modelo de PLN que analisa os sentimentos dos clientes e gera alertas automáticos integrados com o Slack para notificar os sentimentos experimentados pelos clientes e a severidade de cada situação.

Criadores de Ganho (Gain Creators):

- Envio de alertas diretamente para o Slack:* Alertas são enviados automaticamente quando necessários, com a Uber tendo controle total sobre as condições de envio através de uma API programável, permitindo ajustes e melhorias conforme desejado.
- Categorização dos sentimentos dos clientes e a gravidade de cada situação:* Modelo que automatiza as análises e gera resultados mais acurados a cada vez que é utilizado, adaptando-se para oferecer respostas mais eficazes.

Alívio de Dores (Pain Relievers):

- *Envio de alertas recorrentes de acordo com a gravidade e frequência de temas:* Flexibilidade na configuração dos alertas para garantir a relevância e a temporalidade das notificações.
- *Modelo capaz de realizar análises de modo automatizado:* Aumenta a escalabilidade da solução, eliminando a necessidade de análises manuais e proporcionando uma implementação mais eficiente.

Ganhos (Gains):

- *Alertas personalizados em casos de pico de sentimentos:* Assegura que a Uber possa reagir prontamente aos sentimentos dos clientes durante eventos críticos.
- *Capacidade de análise de mais dados com resultados mais precisos:* Permite uma compreensão mais profunda das tendências e padrões dos sentimentos dos clientes, facilitando a tomada de decisão estratégica.

Tarefas do Cliente (Customer Jobs):

- Ler relatórios mensais para entender os sentimentos de seus clientes no último mês e gerar ações a partir disto.

Dores (Pains):

- *Existência de relatórios de sentimento apenas mensais:* Não há um programa que gere relatórios pode limitar a capacidade de resposta rápida a mudanças nos sentimentos dos clientes.
- *Ausência de escalabilidade devido a análises manuais:* A dependência de processos manuais para análise pode reduzir a eficiência e limitar a capacidade de escalabilidade da solução.
- *Falta de parâmetros claros para a categorização dos comentários:* A ausência de definições claras nos parâmetros de categorização pode comprometer a precisão das análises.

Ao aplicar a Proposta de Valor para o desenvolvimento do projeto "Nexus", a equipe conseguiu identificar de maneira clara e detalhada os benefícios essenciais que a solução irá oferecer aos futuros usuários, assim como entender profundamente as necessidades e expectativas dos clientes da Uber. Esta abordagem estruturada permitiu uma visualização integral das vantagens e do impacto potencial do projeto, orientando a criação de uma solução altamente alinhada com as demandas e desejos do público-alvo.

4.4. Matriz de Riscos

A Matriz de Riscos é uma ferramenta muito valiosa para o desenvolvimento de um projeto, pois permite visualizar e analisar os riscos ou oportunidades. Além de mensurar a importância de cada risco ou oportunidade pela probabilidade de ocorrência e seu impacto no projeto, um plano de ação é disponibilizado para lidar com os riscos. O plano define medidas de prevenção para evitar a ocorrência, reações para reverter o impacto, um responsável do grupo para verificar continuamente se o grupo está trabalhando de acordo com as medidas preventivas estabelecidas e uma quantificação do impacto. Esta abordagem detalhada garante que a equipe esteja preparada para lidar com imprevistos, transformando potenciais problemas em oportunidades para o sucesso do projeto.

Quadro 1 - Matriz de Riscos

| Probabilidade | Ameaças | | | | | Oportunidades | | | | |
|---------------|-------------------------|--|---------------------------|--|--|--------------------------------|--|--------------------------------------|-------|-------------|
| | 90% | 70% | 50% | 30% | 10% | Muito Baixo | Baixo | Moderado | Alto | Muito Alto |
| | AUSÊNCIA DE INTEGRANTES | LIMITAÇÕES NA INTERPRETAÇÃO DE LINGUAGEM NATURAL | CARÊNCIA TÉCNICA DO GRUPO | MÁ DISTRIBUIÇÃO DE TAREFAS (ATRASOS E SOBRECARGAS) | VIES NO MÉTODO DE ANALISE DE SENTIMENTOS | DESENVOLVIMENTO DE HABILIDADES | FORTALECIMENTO DA RELAÇÃO UBER- INTELI | RESOLVER CONFLITOS INTERNOS DO GRUPO | | |
| 90% | | | | | | | | | | |
| 70% | | | | | | | | | | |
| 50% | | | | | | | | | | |
| 30% | | | | | | | | | | |
| 10% | | | | | | | | | | |
| | Muito Baixo | Baixo | Moderado | Alto | Muito Alto | Muito Alto | Alto | Moderado | Baixo | Muito Baixo |
| | Impacto | | | | | | | | | |

Fonte: Material produzido pelos autores (2024)

Plano de Ação para os Riscos

- **Carência técnica do grupo:**
 - **Prevenção:** Fazer os autoestudos disponibilizados à turma.
 - **Reação:** Estudo em grupo fora do período de aula.
 - **Responsável:** Paulo Octavio de Paula.
 - **Impacto Quantitativo:** 5/5.
- **Má distribuição de tarefas (atrasos e sobrecargas):**
 - **Prevenção:** Fazer um bom grooming durante a planning.
 - **Reação:** Redistribuição de tarefas.
 - **Responsável:** Enzo Boccia Pagliara.
 - **Impacto Quantitativo:** 5/5.
- **Limitações na interpretação de linguagem natural:**
 - **Prevenção:** Consultar os professores durante o desenvolvimento.
 - **Reação:** Investir tempo para melhorias da solução.
 - **Responsável:** Erik Batista da Silva.
 - **Impacto Quantitativo:** 4/5.
- **Ausência de integrantes:**
 - **Prevenção:** Evitar faltar o máximo possível.
 - **Reação:** Providenciar uma conversa em equipe.

- **Responsável:** Davi Rosalino Glória Motta.
- **Impacto Quantitativo:** 3/5.

- **Viés no modelo de análise de sentimentos:**

- **Prevenção:** Treinar o modelo com diversidade.
- **Reação:** Treinar o modelo novamente.
- **Responsável:** Ana Carolina Cremonezi Martire.
- **Impacto Quantitativo:** 5/5.

- **Falta de dedicação individual ao projeto:**

- **Prevenção:** Estar presente nos encontros e reuniões.
- **Reação:** Providenciar uma conversa em equipe.
- **Responsável:** Enzo Boccia Pagliara.
- **Impacto Quantitativo:** 4/5.

- **Projeto não se adequar aos requisitos do Escritório de Projetos:**

- **Prevenção:** Fazer a documentação em conformidade.
- **Reação:** Designar uma tarefa de correção.
- **Responsável:** Lucas da Silva Barbosa.
- **Impacto Quantitativo:** 3/5.

- **Desentendimento interpessoal entre integrantes:**

- **Prevenção:** Escutar e respeitar mutuamente.
- **Reação:** Providenciar uma conversa em grupo e acompanhamento do orientador.
- **Responsável:** Daniel Zular.
- **Impacto Quantitativo:** 5/5.

- **Mudança no escopo do projeto:**

- **Prevenção:** Fazer perguntas ao parceiro e pedir feedback.
- **Reação:** Conversar com o orientador e modificar o necessário.
- **Responsável:** Davi Rosalino Glória Motta.
- **Impacto Quantitativo:** 4/5.

- **Utilização inadequada do GitHub:**

- **Prevenção:** Fazer commits nomeados e Git Flow.
- **Reação:** Melhorar a organização e renomear arquivos.
- **Responsável:** Erik Batista da Silva.
- **Impacto Quantitativo:** 3/5.

Ao adotar esta Matriz de Riscos e seus planos de ação correspondentes, a Nexus está preparada para antecipar desafios e responder com eficiência. Este cuidado na gestão de riscos é crucial para direcionar o projeto para o sucesso, garantindo que cada passo dado esteja alinhado com nossos valores.

5. Análise de Experiência do Usuário

5.1. Personas

Para um desenvolvimento mais eficaz, é de suma importância a criação de personas. Estas são representações fictícias de usuários reais, com o intuito de compreender mais profundamente as dores do consumidor final da solução proposta neste documento. Com isso, foram desenvolvidas três personas, todas elas funcionárias da Uber, tanto nos setores de marketing quanto ciência de dados.

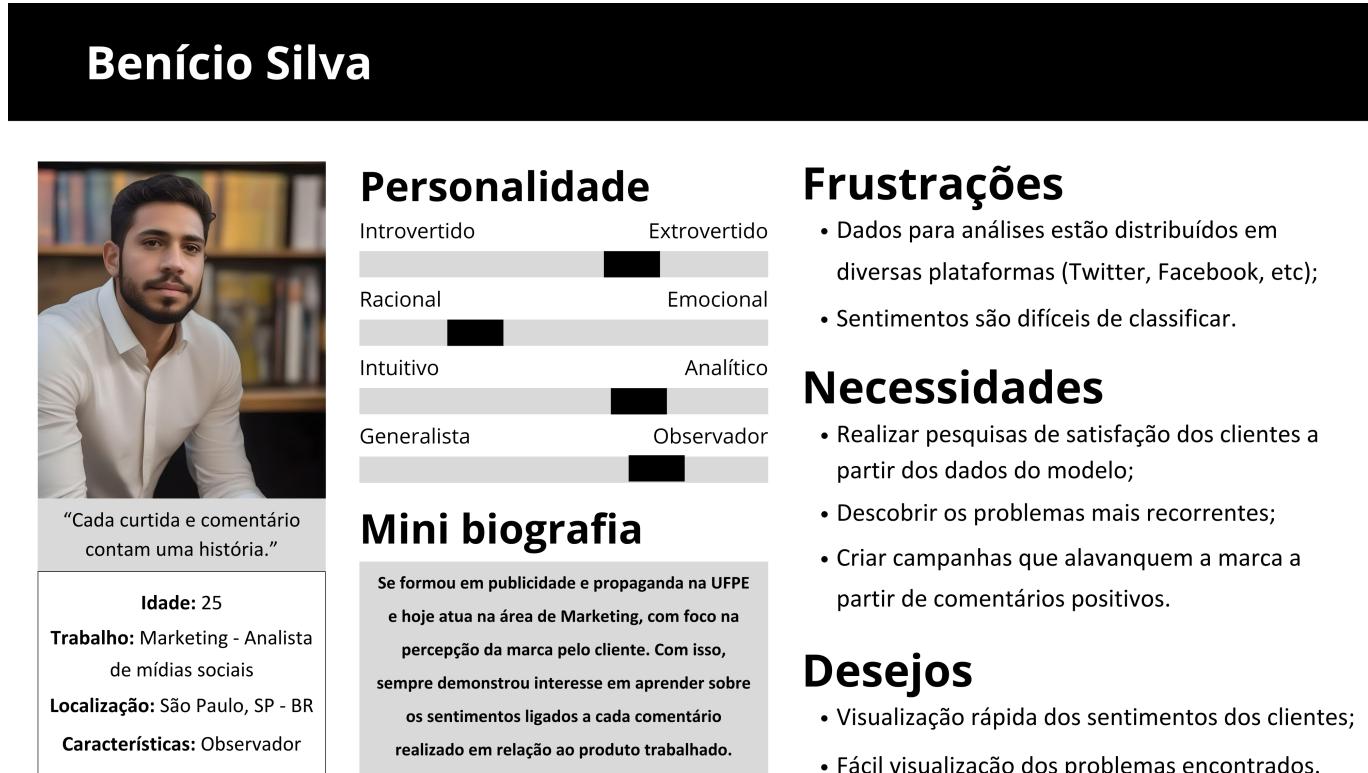
Figura 3 - Persona 1 - Giovanni Silva



Fonte: Material produzido pelos autores (2024)

A primeira pessoa desenvolvida está ligada à implantação, manutenção e melhoria do modelo proposto. Pensando na Uber como uma grande empresa, é de extrema importância que a empresa possua um setor responsável pelo manuseio dos dados e do modelo entregue. Com isso, Giovanni Silva foi pensado como um cientista de dados que seria responsável pelo entregável do presente projeto.

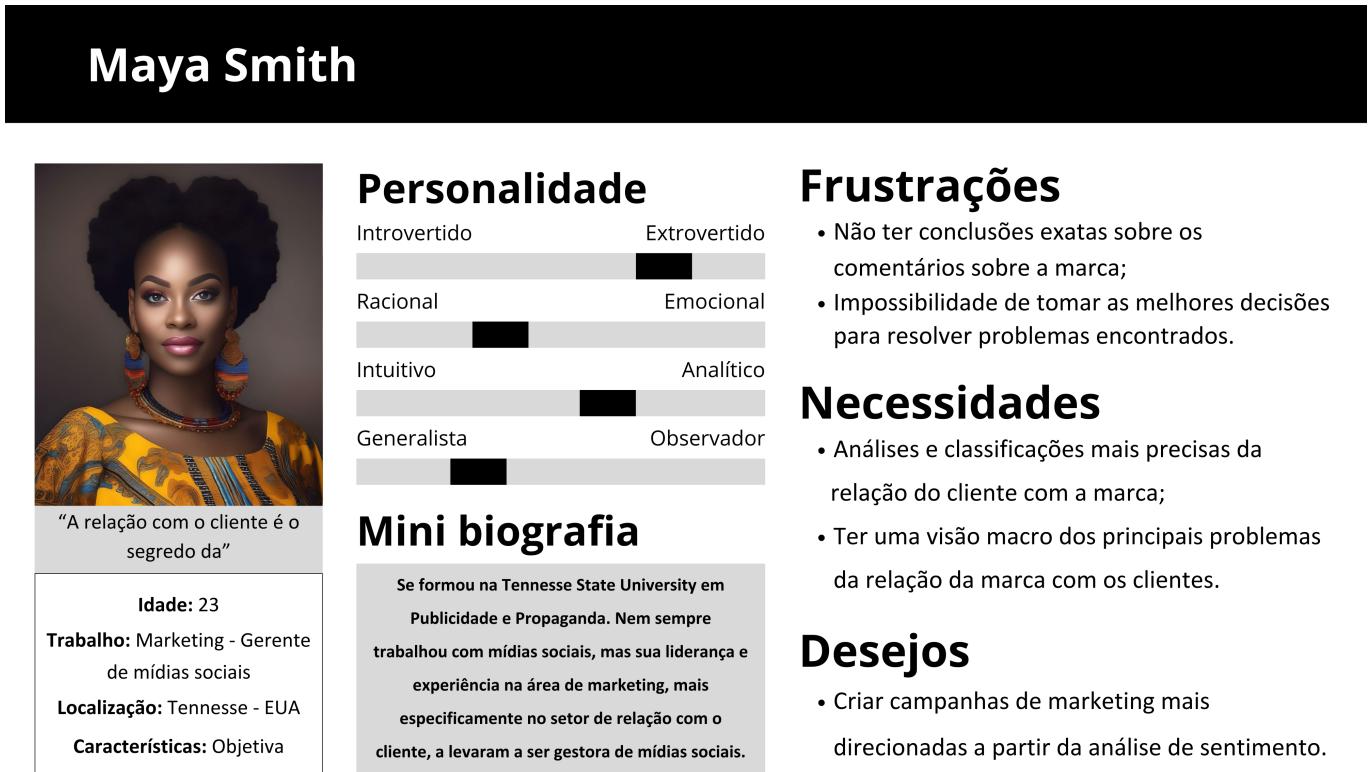
Figura 4 - Persona 2 - Benício Silva



Fonte: Material produzido pelos autores (2024)

A análise da relação da marca com o cliente a partir da interação em redes sociais é extremamente importante. Pensando nisso, a persona "Benício Silva" foi desenvolvida, para que fosse possível entender as necessidades de quem utilizará as informações do modelo proposta para gerar *insights* específicos a partir dos comentários no Twitter (X).

Figura 5 - Persona 3 - Maya Smith



Fonte: Material produzido pelos autores (2024)

Por fim, ainda pensando na análise da relação entre marca e cliente, é de suma importância que haja uma estratégia para combater problemas encontrados e como alavancar os ponto positivos. Uma gestora de mídias sociais é extremamente necessária para tomar decisões estratégicas em um escopo macro dentro da Uber. Portanto, Maya foi desenvolvida para que fossem entendidas as principais necessidades para esse tipo de usuário.

O desenvolvimento de personas contribui para o melhor entendimento do projeto solicitado. Com isso, foram desenvolvidas 3 personas que representam possíveis usuários da solução proposta, abrangendo colaboradores que utilizarão o modelo de forma estratégica e também aqueles que cuidarão da evolução do Modelo de Processamento de Linguagem.

5.2. User Stories

No desenvolvimento de um projeto, criar personas é crucial, pois, dessa forma, os desenvolvedores têm uma ideia muito mais embasada de quem irá usar a solução. Porém, para que se entenda ainda melhor como a solução será utilizada, é necessária a criação de User Stories. De acordo com Willian Stigliani, no site do Cursos PM3, uma User Story (ou história do usuário) "é uma sentença curta e simples sobre uma funcionalidade (escrita sob a perspectiva do usuário que a deseja), utilizada para informar e inspirar decisões de design". Ou seja, são histórias que mostram como a solução entrará em vigor no dia a dia das pessoas já desenvolvidas. [2]

As User Stories geralmente seguem o seguinte padrão de escrita: "Como um *tipo de usuário*, eu quero *uma funcionalidade* para que *algum motivo*." Com base nisso, a Nexus desenvolveu 5 User Stories, dividindo-as entre as 3 personas citadas previamente. Elas estão inseridas nas tabelas a seguir.

Tabela 2 - User Story 01

| Categoria | Dados |
|------------------------|---|
| Número | 01 |
| Título | Atualização do termômetro de sentimentos |
| Persona | Cientista de Dados |
| História | Como cientista de dados, eu quero ter acesso ao código da solução, para poder adicionar comentários novos que surgem a cada dia, deixando, assim, o termômetro de sentimentos atualizado. |
| Critérios de Avaliação | CR01 - O cientista de dados deve ter acesso ao código da solução; CR02 - O cientista de dados deve poder inserir novos comentários na tabela. |
| Critérios de Aceitação | CR01 - O cientista consegue acessar o código; CR02 - O cientista consegue inserir novos comentários. |
| Teste de Aceitação: | CR01 - Acessou: correto. - Não acessou: errado, deve ser corrigido. correto. CR02 - Inseriu: correto. - Não inseriu: errado, deve ser corrigido. correto. |

Fonte: Material produzido pelos autores (2024)

Tabela 3 - User Story 02

| Categoria | Dados |
|------------------------|---|
| Número | 02 |
| Título | Análise quantitativa dos comentários - comentários categorizados como positivo, neutro e negativo |
| Persona | Cientista de Dados |
| História | Como cientista de dados, eu quero ter acesso ao termômetro de sentimentos, para entender quantos comentários são positivos, neutros e negativos. |
| Critérios de Avaliação | CR01 - O cientista de dados deve ter acesso ao termômetro de sentimentos; CR02 - O cientista de dados deve ter acesso à quantidade de comentários. |
| Critérios de Aceitação | CR01 - O cientista consegue acessar o termômetro; CR02 - O cientista consegue ver a quantidade de comentários. |

| Categoria | Dados |
|---------------------|--|
| Teste de Aceitação: | <p>CR01 - Acessou: correto.</p> <ul style="list-style-type: none"> - Não acessou: errado, deve ser corrigido. correto. |
| | <p>CR02 - Visualizou: correto.</p> <ul style="list-style-type: none"> - Não visualizou: errado, deve ser corrigido. correto. |

Fonte: Material produzido pelos autores (2024)

Tabela 4 - User Story 03

| Categoria | Dados |
|------------------------|---|
| Número | 03 |
| Título | Análise quantitativa dos comentários - palavras semelhantes |
| Persona | Analista de Marketing |
| História | Como analista de marketing, eu quero ter acesso ao termômetro de sentimentos, para saber quantos comentários possuem palavras/frases/orações semelhantes. |
| Critérios de Avaliação | <p>CR01 - O analista de marketing deve ter acesso ao termômetro de sentimentos;</p> <p>CR02 - O analista de marketing deve ter acesso à quantidade de comentários que tenham palavras/frases/orações parecidas.</p> |
| Critérios de Aceitação | <p>CR01 - O analista consegue acessar o termômetro;</p> <p>CR02 - O analista consegue a ver quantidade de comentários com palavras/frases/orações parecidas.</p> |
| Teste de Aceitação: | <p>CR01 - Acessou: correto.</p> <ul style="list-style-type: none"> - Não acessou: errado, deve ser corrigido. correto. <p>CR02 - Visualizou: correto.</p> <ul style="list-style-type: none"> - Não visualizou: errado, deve ser corrigido. correto. |

Fonte: Material produzido pelos autores (2024)

Tabela 5 - User Story 04

| Categoria | Dados |
|-----------|---|
| Número | 04 |
| Título | Análise qualitativa dos comentários - impacto |
| Persona | Analista de Marketing |

| Categoria | Dados |
|------------------------|--|
| História | Como analista de marketing, eu quero ter acesso ao termômetro de sentimentos, para saber o quanto impactantes (tanto negativamente quanto positivamente) os comentários podem ser para a Uber. |
| Critérios de Avaliação | CR01 - O analista de marketing deve ter acesso ao termômetro de sentimentos; CR02 - O analista de marketing deve ter acesso à qualidade dos comentários (ou seja, o quanto impactantes eles são). |
| Critérios de Aceitação | CR01 - O analista consegue acessar o termômetro; CR02 - O analista tem acesso aos comentários ou aos links que levam aos mesmos. |
| Teste de Aceitação: | CR01 - Acessou: correto. - Não acessou: errado, deve ser corrigido. correto. CR02 - Acessou: correto. - Não acessou: errado, deve ser corrigido. correto. |

Fonte: Material produzido pelos autores (2024)

Tabela 6 - User Story 05

| Categoria | Dados |
|------------------------|--|
| Número | 05 |
| Título | Ações necessárias de acordo com as análises de comentários |
| Persona | Gestora de Marketing |
| História | Como gestora de marketing, eu quero ter acesso ao termômetro de sentimentos, para saber quais medidas devem ser tomadas de acordo com a quantidade e qualidade dos comentários. |
| Critérios de Avaliação | CR01 - A gestora de marketing deve ter acesso ao termômetro de sentimentos; CR02 - A gestora de marketing deve saber a quantidade e qualidade dos comentários. |
| Critérios de Aceitação | CR01 - A gestora consegue acessar o termômetro; CR02 - A gestora tem acesso à quantidade e qualidade dos comentários. |
| Teste de Aceitação: | CR01 - Acessou: correto. - Não acessou: errado, deve ser corrigido. correto. CR02 - Acessou: correto. - Não acessou: errado, deve ser corrigido. correto. |

Fonte: Material produzido pelos autores (2024)

Ao criar diferentes User Stories para as diferentes personas existentes, fica mais claro como a solução proposta será utilizada. Essas histórias representam necessidades que a Nexus imagina que cada persona tem e/ou pode ter. Portanto, ao desenvolvê-las, o grupo tem plena noção de quais features (ou características) a solução deve ter para sanar tais desejos.

6. Análise Descritiva

Uma análise descritiva é um método estatístico que busca compreender e resumir os principais aspectos de um conjunto de dados, geralmente por meio de medidas como média, mediana, moda, desvio padrão e quartis. Seu foco é, de acordo com o "Five Acts", é "compreender se, por trás de um ou mais fenômenos que se repetem, existem tendências ou padrões que possam ser mapeados." [3]. Essa abordagem permite aos pesquisadores e analistas identificar padrões, tendências e características essenciais dos dados sem realizar inferências mais complexas sobre a população de onde os dados foram coletados, fornecendo uma visão detalhada e comprehensível do conjunto de dados em questão.

A análise descritiva foi o processo realizado para entender a base de dados da Uber e descobrir quais etapas de pré-processamento faziam-se ou não necessárias. Assim, a primeira etapa foi transformar o arquivo `.csv` em um `dataframe` da biblioteca Pandas, para trazer mais facilidade na visualização e manuseio dos dados. Assim, obteve-se o resultado demonstrado na Figura 6, onde é possível visualizar as 10 primeiras linhas da base de dados. Nesta, encontram-se duas colunas principais, `comment` e `sentiment`. Na primeira são armazenados os comentários retirados da plataforma Twitter, enquanto a segunda mostra o sentimento extraído de cada comentário, podendo este ser positivo (1), neutro (0) ou negativo (-1).

Figura 6 - Dataframe

| id | | comment | sentiment |
|-----------|----|--|------------------|
| 0 | 1 | That, my friend, is why The Mighty Swift Radio Cars of Stalybridge retain my costume. | 0 |
| 1 | 2 | Spent 20 minutes in an Uber listening to what I can best describes as ?Eagles B-sides, but about Jesus? | 0 |
| 2 | 3 | via The Guardian Guardian front page, Monday 11 July 2022 - The #Uber files: Leak reveals secret lobbying operation to conquer the world https://t.co/hjsUSc6AVZ | -1 |
| 3 | 4 | My real job is being my girlfriends personal Uber/Uber eats driver Everything else is just a side gig | 0 |
| 4 | 5 | i had a bad drive . i want my refund | -1 |
| 5 | 6 | Uber broke laws, duped police and secretly lobbied governments, leak reveals https://t.co/JMfXJFO6jc | -1 |
| 6 | 7 | We're here to help. Please send us a DM with the phone number associated with your Uber account, so we can assist you further. | -1 |
| 7 | 8 | Too many taxi drivers have told me about Uber cutting access to the app and withholding earned wages - for months at a time - due to spurious customer complaints. Driver evidence made no difference. A big data leak shows more reasons not to use them. https://t.co/DjcGHX9fYx | -1 |
| 8 | 9 | why do uber drivers like to make conversation with me | -1 |
| 9 | 10 | Thing are getting serious with Uber business model,technology isn't an error but Uber do | -1 |

Fonte: Material produzido pelos autores (2024)

Nas primeiras linhas do dataframe já é possível identificar a presença de 'urls' (links) nos comentários - ids: 3, 6 e 8 - , os quais levam a sites de notícias referenciados nos tweets. isto ocorre em diversas linhas da base de dados, o que é até mesmo identificado na nuvem de palavras, espécie de gráfico que sinaliza a frequência das palavras em determinado trecho. Na nuvem de palavras feita tendo, como base, todos os comentários, o vocábulo 'https' - termo que inicia os 'urls' - foi uma das palavras que mais apareceu. No entanto, todos os links existentes levam a notícias que já possuem seus círculos indicados no corpo do tweet. Assim, chegou-se a conclusão de que tais 'urls' poderiam ser removidos no pré-processamento, já que não trazem valor semântico aos tweets e podem apenas enviesar o modelo.

Figura 7 - Nuvem de Palavras Inicial



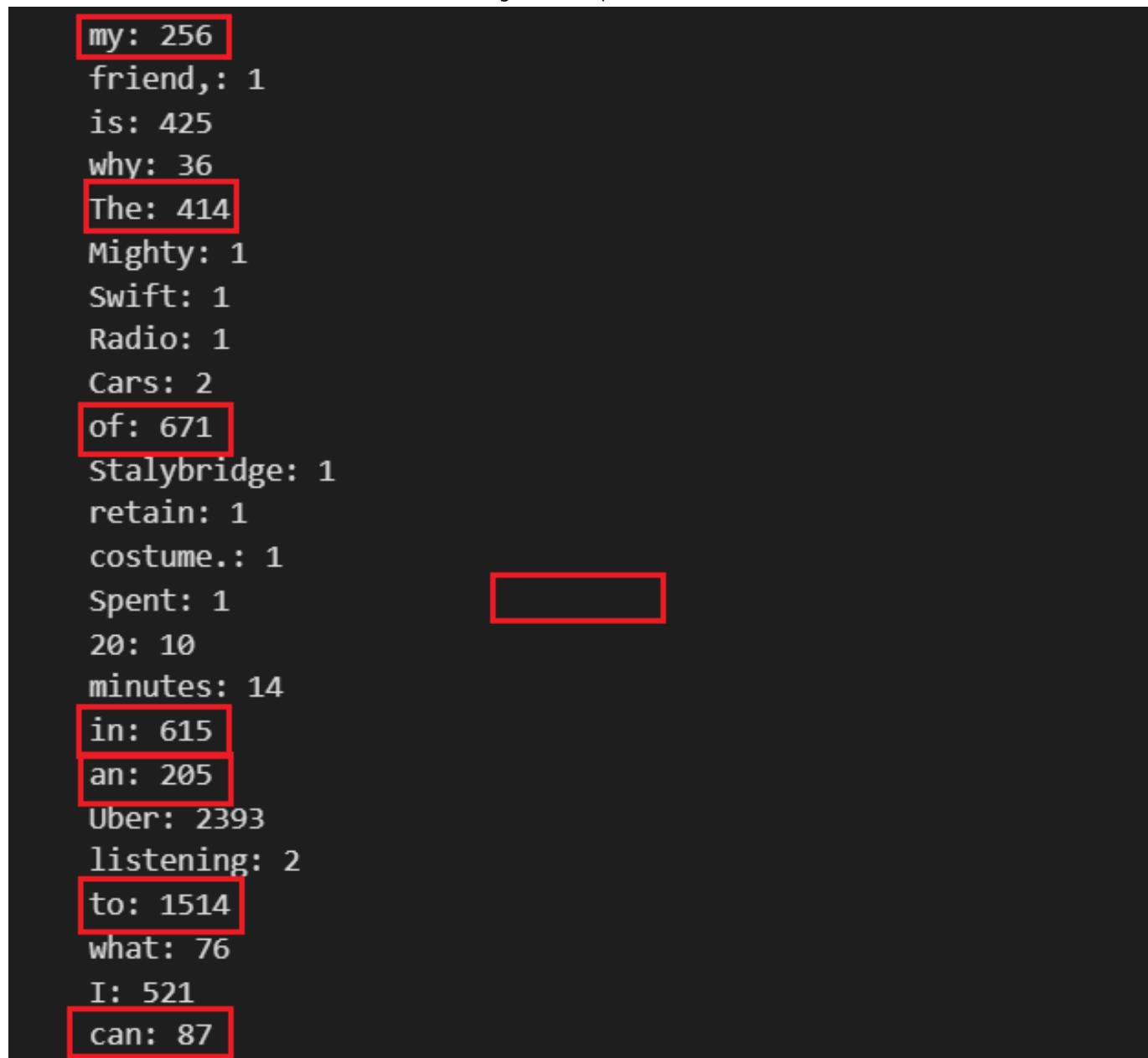
Fonte: Material produzido pelos autores (2024)

Além do termo 'https', também é possível visualizar na Figura 7 acima palavras como "lobbied governments", "lobbying operation", "lobbyimg drive", todas possuindo a palavra "lobby" conjugada em diversos tempos verbais. A fim de resolver problemas como esses, pode-se utilizar a **lemmatização**, etapa do pré-processamento capaz de remover as flexões das palavras, trazendo, por exemplo, os verbos ao infinitivo.

Outro fator importante é que também pode-se ver palavras como "Macron" e "Emanuel Macron" na nuvem, ambas se referindo à mesma pessoa: o presidente da França. Assim, foi utilizado o **NER** (Named Entity Recognition, ou Reconhecimento de Entidades Nomeadas em português) para identificar tais palavras como as entidades que representam, sejam estas pessoas, lugares ou outros.

Após visualizar a nuvem de palavras, foi criado um código para mostrar o número de vezes que cada palavra aparecia no texto. Assim, fez-se possível entender que havia muitas stop-words, palavras que não agregam à semântica das frases. A frequência de algumas palavras pode ser vista nas Figuras X.

Figura 8 - Stop-words

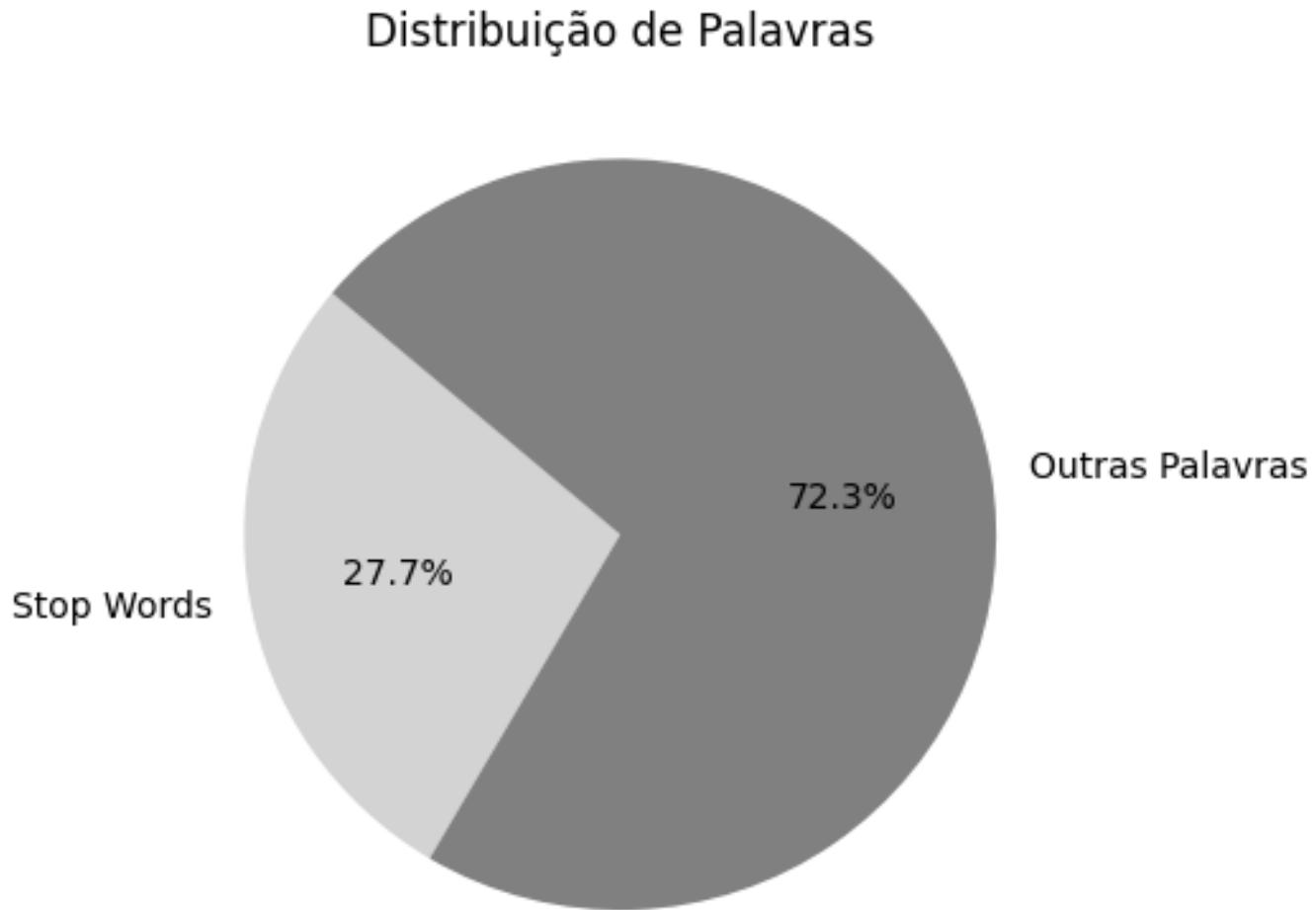


Fonte: Material produzido pelos autores (2024)

Além disso, também foi possível plotar um gráfico para mostrar a quantidade de stop-words que existem em relação ao total de palavras (tal gráfico pode ser visto na Figura 9). A partir deste, tornou-se clara a necessidade de **remover as stop-words**, pois, dado que representam quase 30% das palavras na base de dados, sua presença pode enviesar consideravelmente os resultados do modelo utilizado no momento (Bag of Words).

No entanto, para a realização desta etapa também faz-se necessário o uso da **tokenização**, processo que separa o texto em tokens, ou seja, palavras, as quais podem então ser verificadas como sendo ou não stop-words.

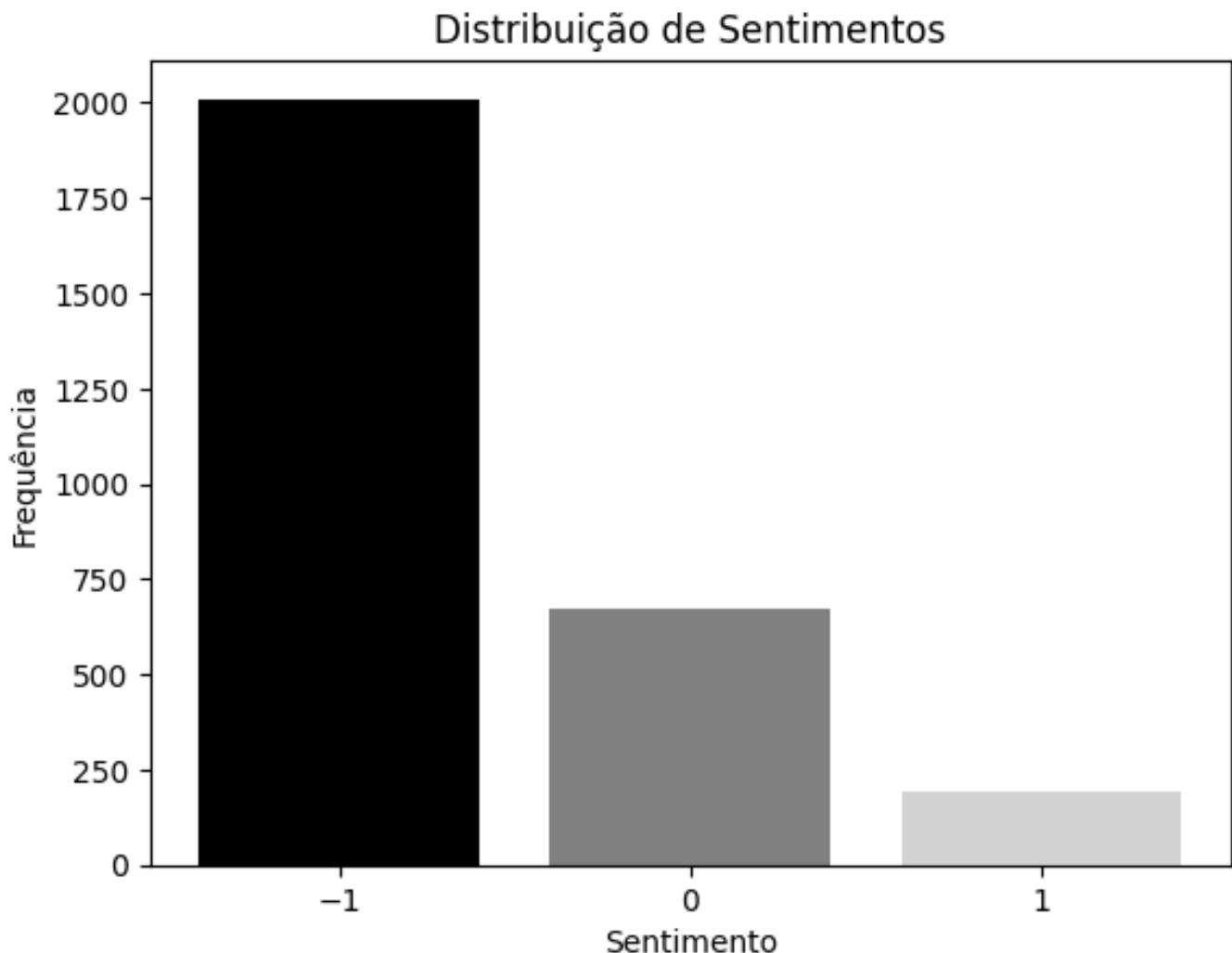
Figura 9 - Distribuição das Palavras



Fonte: Material produzido pelos autores (2024)

Outro ponto interessante de entender é o balanceamento da base de dados. Como os tweets estão classificados em positivo, negativo ou neutro, é bom entender o quanto de cada sentimento existe, para que a melhor estratégia seja descoberta e utilizada. A distribuição dos sentimentos pode ser vista na Figura 10 a seguir, onde os sentimentos representados com '-1' são negativos, os com '0' são neutros, e '1' são positivos.

Figura 10 - Distribuição de Sentimentos



Fonte: Material produzido pelos autores (2024)

Essa análise descritiva é fundamental para compreender melhor a base de dados em questão. Ela permite a identificação de padrões, tendências e possíveis problemas nos dados, além de ajudar na seleção das técnicas de pré-processamento e modelagem mais adequadas. Trata-se de um processo iterativo, em que novas análises podem ser necessárias para refinar o entendimento e aprimorar os modelos ao longo do projeto. Assim, essa análise contínua serve como uma ferramenta valiosa para orientar as decisões e maximizar o valor do projeto.

7. Documentação do Pré-Processamento

7.1. Segmentação

A segmentação é uma etapa fundamental no pré-processamento de dados para projetos de processamento de linguagem natural (PLN). Este processo divide um texto em unidades menores, normalmente sentenças, facilitando análises posteriores. Tal subdivisão é crucial para que o modelo de PLN possa tratar cada sentença individualmente, melhorando tanto a compreensão do contexto quanto a precisão na interpretação do conteúdo.

Neste projeto, foi utilizada uma abordagem sofisticada de segmentação para garantir que cada sentença seja isolada corretamente, o que é particularmente valioso em tarefas como a análise de sentimentos. Essa técnica permite identificar transições temáticas e mudanças de tom dentro do texto, que poderiam permanecer obscuras em um fluxo contínuo de texto. A precisão na segmentação é alcançada através de modelos avançados de PLN que compreendem as regras linguísticas e de pontuação específicas do idioma do texto processado.

A utilização de ferramentas de PLN como o spaCy para executar a segmentação facilita a identificação efetiva dos limites das sentenças, garantindo que cada unidade textual seja processada como uma entidade separada. Este cuidado inicial no processamento de texto é essencial para assegurar que as análises subsequentes, como tokenização e análise semântica, sejam realizadas com a máxima eficácia.

Na imagem abaixo, foi efetuada a segmentação. A função `segmentacao` implementa a segmentação de texto em frases utilizando o modelo de processamento de linguagem spaCy. Primeiramente, o texto é processado através do modelo spaCy fornecido como argumento para a função. Em seguida, o texto processado é iterado sobre suas sentenças, e o texto de cada sentença é extraído e armazenado em uma lista de frases segmentadas. Essa abordagem permite a identificação efetiva dos limites das sentenças, garantindo que cada unidade textual seja tratada como uma entidade separada.

Figura 11 - Etapa de pré-processamento: Segmentação



```
1 def segmentacao(texto, nlp):
2     """
3         Segmenta um texto em frases usando um modelo de processamento de linguagem.
4
5     Args:
6         texto: str - O texto a ser segmentado em frases.
7
8     Returns:
9         list: Uma lista das frases segmentadas.
10    """
11
12
13    # Processa o texto usando o modelo SpaCy
14    doc = nlp(texto)
15
16    # Extrai cada frase do documento e armazena na lista
17    frases_segmentadas = [sent.text for sent in doc.sents]
18
19    # Retorna a lista de frases segmentadas
20    return frases_segmentadas
```

Fonte: Material produzido pelos autores (2024)

Portanto, a segmentação não apenas prepara o texto para uma análise mais detalhada e específica, mas também otimiza o desempenho dos modelos de PLN ao focar nos elementos significativos do texto,

contribuindo para a obtenção de insights mais precisos e a implementação de soluções eficazes baseadas em dados.

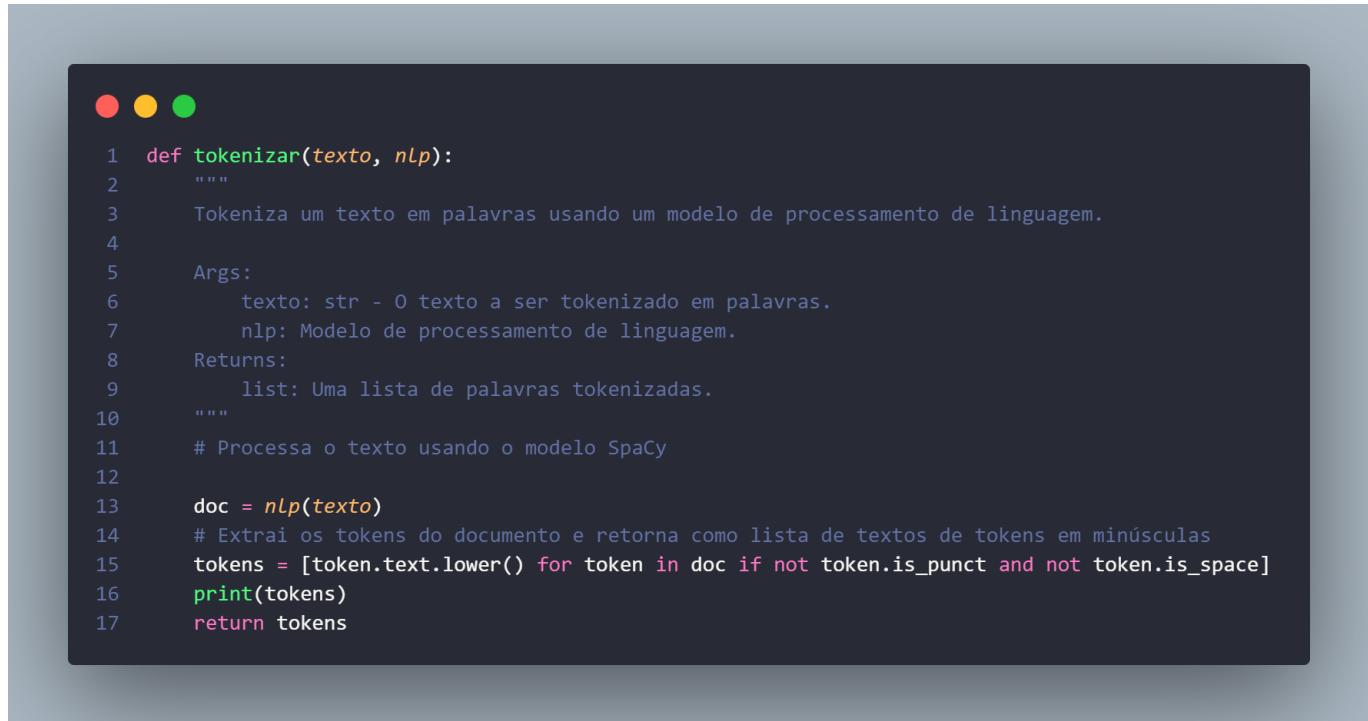
7.2. Tokenização

A tokenização é uma etapa fundamental de pré-processamento em projetos de PLN. Essencialmente, esse processo envolve a divisão de textos extensos em unidades menores, conhecidas como 'tokens', que podem ser palavras, frases ou símbolos. Esta etapa é crucial para simplificar a análise subsequente e para a aplicação de outras técnicas de PLN, como lematização ou análise de sentimentos.

O grupo *Nexus* escolheu a tokenização como método de pré-processamento baseando-se na análise descritiva dos dados, que indicou a presença significativa de 'URLs' e variações verbais nos comentários coletados. Essa análise destacou a necessidade de isolar cada palavra individualmente para uma limpeza eficaz e precisa dos dados, garantindo que elementos não essenciais, como pontuações e espaços em branco, fossem removidos antes de proceder a análises mais complexas.

Funcionalmente, a tokenização no contexto de PLN é realizada através de um modelo computacional que processa o texto. O código abaixo exemplifica a implementação de tokenização utilizando a biblioteca spaCy, um popular framework de PLN:

Figura 12 - Etapa de pré-processamento: Tokenização



```
● ● ●
1 def tokenizar(texto, nlp):
2     """
3         Tokeniza um texto em palavras usando um modelo de processamento de linguagem.
4
5     Args:
6         texto: str - O texto a ser tokenizado em palavras.
7         nlp: Modelo de processamento de linguagem.
8
9     Returns:
10        list: Uma lista de palavras tokenizadas.
11
12    """
13    # Processa o texto usando o modelo SpaCy
14
15    doc = nlp(texto)
16    # Extrai os tokens do documento e retorna como lista de textos de tokens em minúsculas
17    tokens = [token.text.lower() for token in doc if not token.is_punct and not token.is_space]
18    print(tokens)
19
20    return tokens
```

Fonte: Material produzido pelos autores (2024)

Este código processa o texto inserido, utilizando o modelo de linguagem fornecido pelo spaCy para desmembrá-lo em 'tokens', excluindo pontuações e espaços, o que simplifica a análise subsequente e melhora a acurácia dos modelos de PLN.

Portanto, a tokenização desempenha um papel indispensável no pré-processamento de textos para PLN, ao segmentar textos em unidades manejáveis que são essenciais para a aplicação eficiente de técnicas de análise. Para o projeto de análise de sentimentos do grupo *Nexus*, a tokenização não apenas facilita a

remoção de ruídos nos dados, como também prepara o terreno para uma análise mais precisa e relevante dos sentimentos expressos nos comentários, contribuindo significativamente para a obtenção de insights mais detalhados e acionáveis sobre as percepções dos usuários em relação à Uber.

7.3. Stop-Words

A remoção de stop-words é a etapa do pré-processamento que serve para melhorar o funcionamento do modelo. Stop-words são palavras comuns que geralmente não contribuem para o significado de um texto e, portanto, podem ser removidas sem prejudicar a compreensão geral da mensagem. Exemplos de stop-words na língua inglesa são as palavras "the", "is", "and", entre outras, as quais são extremamente comuns, porém não agregam nada. Para se entender a escala disso, 27.7% das palavras na base de dados deste projeto são stop-words, ou seja, mais de um quarto da atual base de dados é composta por tais palavras, que podem atrapalhar a análise do modelo.

A remoção de stop-words oferece várias vantagens, sendo um deles a redução de ruídos, já que, devido à frequência com que aparecem, podem enviesar modelos, e ao serem removidas, o modelo consegue dar foco às palavras mais importantes. Com isso, há uma melhora nos resultados das análises, visto que o modelo agora consegue observar os termos mais importantes, além de um aumento de desempenho, já que o modelo analisa uma quantidade menor de palavras, assim realizando os processos de forma mais rápida e econômica.

A função `remove_stop_words` definida no código abaixo tem o propósito específico de efetuar a remoção de stop-words como um pré-processamento de texto. Ela aceita dois parâmetros: `tokens`, uma lista de palavras ou termos, e `nlp`, um objeto de modelo de linguagem natural, como os da biblioteca spaCy. Dentro da função, o conjunto de stop-words é carregado através de `nlp.Defaults.stop_words`. O objetivo é retornar uma nova lista que contenha apenas aqueles tokens que não são entendidos como stop-words, usando uma compreensão de lista para filtrar os tokens que não devem continuar na análise.

Figura 13 - Etapa de pré-processamento: Stop-Words

```
1 #função de stop words
2 def remove_stop_words(tokens, nlp):
3     """
4         Remove as stop words de uma lista de tokens.
5
6     Args:
7         tokens: list - Uma lista de tokens.
8
9     Returns:
10        list: Uma lista de tokens sem as stop words.
11    """
12    # Carrega as stop words do modelo SpaCy
13    stop_words = nlp.Defaults.stop_words
14
15    # Remove as stop words dos tokens
16    return [token for token in tokens if token not in stop_words]
```

Fonte: Material produzido pelos autores (2024)

A remoção de stop-words é uma etapa crucial durante o pré-processamento de dados, já que contribui significativamente para a qualidade e eficiência das análises de texto. Ao eliminar termos que não agregam valor semântico, é possível direcionar melhor os esforços de análise para os aspectos mais relevantes do texto, resultando em "insights" mais precisos e modelos mais eficazes. Essa prática não apenas reduz o ruído nos dados, mas também otimiza o desempenho dos modelos, permitindo uma interpretação mais precisa e uma tomada de decisão mais informada.

7.4. Lematização

A lematização é uma técnica de pré-processamento de texto usada em PLN para reduzir palavras ao seu lema. Essencialmente, ela transforma as palavras em sua forma base, ajudando a padronizar variações da mesma palavra para uma só representação. Isso é útil em tarefas como análise de sentimentos, pois permite que o modelo trate diferentes formas de uma palavra (como "running", "ran", "runs") como uma entidade única ("run"), simplificando a análise e melhorando a eficiência dos modelos de PLN.

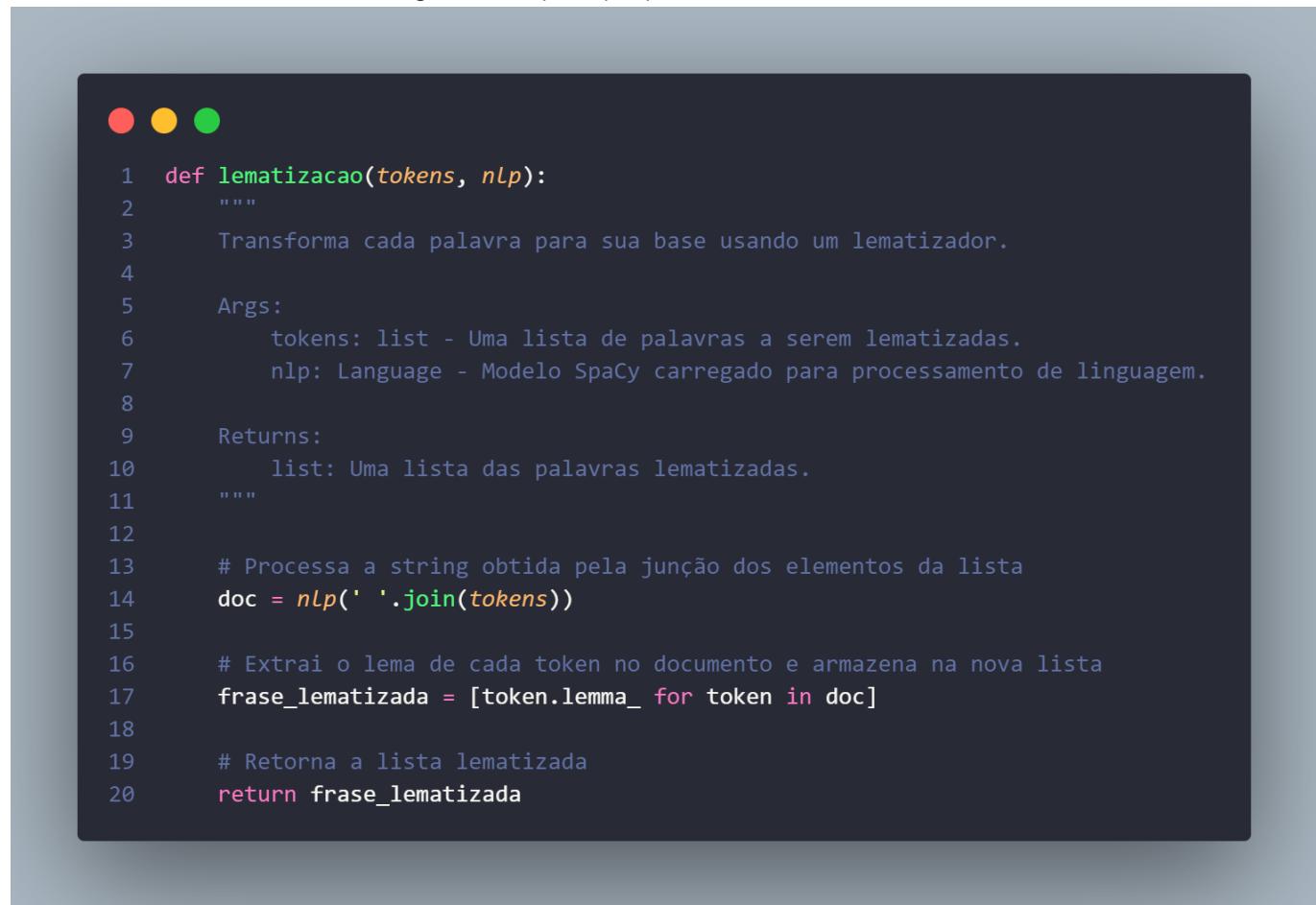
A lematização foi escolhida como uma das técnicas de pré-processamento neste projeto por sua capacidade de reduzir variações de palavras a uma forma base, aumentando a eficácia na análise de dados. Na Análise Descritiva apresentada acima, pode-se ver a recorrência de termos como "lobby", "lobbied" e "lobbying". Isso destaca a necessidade de consolidar essas variações em uma única representação: "lobby". Essa padronização simplifica a interpretação das palavras e auxilia na análise ao focar no significado essencial, reduzindo a redundância dos dados e permitindo que o modelo de PLN opere de maneira mais eficiente e precisa.

Além de ajudar no vocabulário, a lematização é crucial para a precisão na análise de sentimentos, especialmente considerando a predominância de sentimentos negativos na base de dados. Ao padronizar as formas verbais e simplificar a estrutura do texto, esta técnica garante que o modelo identifique e interprete corretamente os sentimentos expressados, refletindo de maneira mais acurada e eficaz as opiniões dos usuários.

Na prática, a lematização é um processo que comprehende o contexto e aplica regras morfológicas às palavras, levando em conta sua função no texto. Diferente do stemming, que corta os sufixos das palavras, a lematização faz uma análise completa da utilização da palavra, o que permite obter resultados mais precisos e alinhados ao significado no contexto.

Para realizar a lematização computacionalmente, foi utilizada uma biblioteca de processamento de textos, como a spaCy. Um método lematizacao recebe uma lista de palavras, tokens, e o modelo de linguagem já carregado. Este modelo processa as palavras, criando um documento (`doc`). Cada palavra do mesmo é tratada como um token. Utilizamos o método `token.lemma_` em cada token, afim de extrair o lema da palavra. O lema de cada palavra é, então, armazenado em uma nova lista, `frase_lematizada`, formando uma frase lematizada a partir das palavras originais, como pode ser visto na Figura 14 a seguir.

Figura 14 - Etapa de pré-processamento: Lematização



```
1 def lematizacao(tokens, nlp):
2     """
3         Transforma cada palavra para sua base usando um lematizador.
4
5     Args:
6         tokens: list - Uma lista de palavras a serem lematizadas.
7         nlp: Language - Modelo SpaCy carregado para processamento de linguagem.
8
9     Returns:
10        list: Uma lista das palavras lematizadas.
11    """
12
13    # Processa a string obtida pela junção dos elementos da lista
14    doc = nlp(' '.join(tokens))
15
16    # Extrai o lema de cada token no documento e armazena na nova lista
17    frase_lematizada = [token.lemma_ for token in doc]
18
19    # Retorna a lista lematizada
20    return frase_lematizada
```

Fonte: Material produzido pelos autores (2024)

Portanto, a lematização é uma ferramenta valiosa no pré-processamento de textos para PLN, simplificando a estrutura linguística do texto e permitindo que algoritmos de aprendizado de máquina trabalhem de maneira mais eficaz. Especialmente para a *Nexus*, em um projeto de análise de sentimentos,

reduzir palavras a uma forma base pode ajudar o modelo a identificar e analisar as opiniões expressas nos comentários dos clientes, trazendo resultados mais confiáveis para a Uber em relação as suas percepções e sentimentos.

7.5. NER

Uma das últimas etapas que a *Nexus* utilizou para efetuar o pré-processamento foi o NER, ou Reconhecimento de Entidades Nomeadas (Named Entity Recognition, em inglês). Essa é uma técnica de PLN que visa identificar e classificar entidades nomeadas em um texto em categorias pré-definidas, como organizações, locais, datas, etc. Por exemplo, existem diversos comentários de usuários da Uber que citam Emmanuel Macron, atual presidente da França, de formas diferentes: "emmanuelmacron", "Macron", "EmmanuelMacron". Ao processar os comentários pelo NER, todas essas variações "entram" na categoria "PESSOA".

É importante explicar como o NER funciona, entrando já em termos um pouco mais técnicos. É criada uma função, que, nesse caso, foi chamada de `NER`. Primeiramente, a função recebe o texto a ser processado e um objeto `nlp`, que é uma instância de um modelo de processamento de linguagem natural. No exemplo fornecido, a biblioteca spaCy é usada para esse fim.

Ao chamar a função NER com um texto específico, o spaCy processa esse texto e o transforma em um objeto chamado `doc`, que representa o documento analisado. Em seguida, a função itera sobre as entidades identificadas neste documento usando a propriedade `ents` do objeto `doc`.

Cada entidade identificada é representada por um objeto `ent`, que contém duas propriedades principais: `ent.text`, que representa o texto da entidade identificada, e `ent.label_`, que representa o rótulo atribuído a essa entidade pelo modelo. Por exemplo, uma entidade pode ser rotulada como "PESSOA" se representar o nome de uma pessoa, ou "LOCAL" se representar um local geográfico. A seguir pode-se ver como foi feita a estruturação dessa etapa.

Figura 15 - Etapa de pré-processamento: NER



```
1 def NER(texto, nlp):
2     """
3     Categoriza as palavras como entidades e tipos.
4
5     Args:
6         texto: str - O texto a ser processado.
7
8     Returns:
9
10    """
11
12    doc = nlp(texto)
13
14    print([(ent.text, ent.label_) for ent in doc.ents])
15
16    return [(ent.text, ent.label_) for ent in doc.ents]
```

Fonte: Material produzido pelos autores (2024)

A função NER retorna, por fim, uma lista de tuplas, onde cada tupla contém o texto da entidade e o rótulo correspondente. Isso permite que os usuários obtenham as entidades identificadas e seus tipos associados para análise posterior ou tomada de decisão.

Essa etapa de pré-processamento não poderia ser deixada de lado, pois é essencial ao ajudar a extrair informações relevantes, a entender o contexto de um texto e a melhorar o desempenho de várias tarefas de processamento de linguagem natural. Como citado no exemplo acima, palavras como "EmanuelMacron" e "Macron" podem ser vistas de formas distintas pelo modelo utilizado. Ao implementar o NER como uma etapa de pré-processamento, essas palavras passam a pertencer a um mesmo conjunto, o que facilita a compreensão semântica.

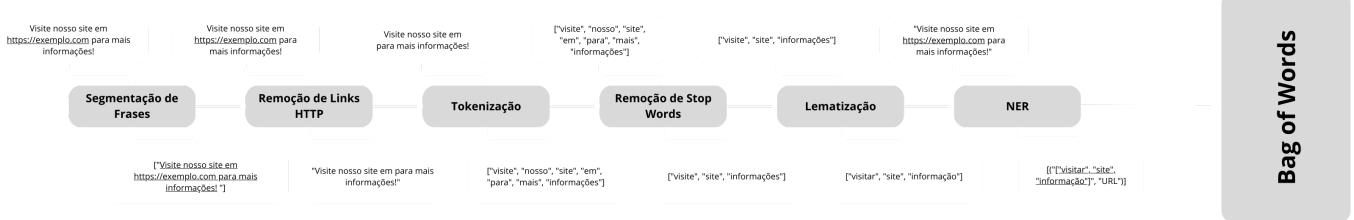
7.6. Pipeline

Em suma, o *pipeline* é a junção de todos os processos definidos. Para isso é necessário ter-se apontado todos os processos os quais os dados irão passar. Considerando o projeto do presente documento, as etapas de pré processamento já foram definidas, então basta apenas definir uma ordem para que essas ocorram.

Para os dados apresentados pela *Uber*, foi necessário aplicar a segmentação das frases, após isso, todos os endereços de sites foram removidos, então o texto foi tokenizado, com isso as *stop words* foram

removidas, dos tokens resultantes houve a lematização e por fim, foi aplicado o NER. Com essas etapas foi possível realizar o pré processamento necessário para lidar da melhor forma com os dados fornecidos.

Figura 16 - Figura ilustrativa do pipeline



Fonte: Material produzido pelos autores (2024)

8. Vetorização

8.1. Bag of Words

8.1.1. Introdução

No artigo "What is Bag of Words?" [4], a IBM explica que o Bag of Words é uma técnica fundamental no campo do Processamento de Linguagem Natural (PLN), usada para converter textos em vetores. Esta etapa é crucial, pois os modelos de machine learning requerem dados numéricos como entrada. Portanto, é essencial representar os textos que queremos analisar numericamente para que os modelos possam processá-los adequadamente.

8.1.2. Método

Neste modelo, primeiro é criado um vocabulário contendo todas as palavras únicas encontradas no texto de entrada. Em seguida, cada frase ou comentário no texto de entrada é comparado com este vocabulário, e o resultado dessa comparação é registrado em uma lista. Dessa forma, o output gerado é uma matriz composta por listas que representam cada uma das frases analisadas.

Com o Bag of Words, as frases são vetorizadas como no exemplo abaixo:

```

# Este é o vocabulário criado a partir do input recebido pelo modelo:
['i', 'hate', 'love', 'driver', 'uber', 'led', 'safely', 'me', 'the']

# E esta é a primeira frase que se deseja vetorizar:

frase = "i love uber"

# Não considerando a remoção de stop-words, o vetor gerado seria:

[1, 0, 1, 0, 1, 0, 0, 0, 0]
  
```

O exemplo acima ilustra o funcionamento do Bag of Words, que executa o mesmo procedimento para todas as frases presentes no input. Os vetores são criados considerando o tamanho do vocabulário, de modo que cada vetor tenha uma lista com o mesmo número de elementos. Posteriormente, verifica-se a presença das palavras do vocabulário na frase em questão. Se uma palavra não estiver presente na frase, é adicionado '0'; se estiver presente, é adicionado o número de vezes que aparece na frase, no mesmo índice em que a palavra está no vocabulário.

Neste projeto, todo o processo de construção do 'Bag of Words' se dá através de uma instância da classe `CountVectorizer()`, pertencente à biblioteca 'Scikit-learn'.

8.1.3. Resultados

Como resultados da aplicação deste modelo, obteve-se uma matriz com 2876 linhas e 5269 colunas, a qual pode ser observada na Figura 17 a seguir.

Figura 17 - Matriz BoW

| id | comment | sentiment | 00 | 000 | 00am | 02 | 039 | 06 | 0600 | ... | t2 | t22 | t25 | t260 | t27 | t30 | t5 | t55 | t70 | t74 |
|-----------|---|------------------|-----|-----|------|-----|-----|-----|------|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|-----|
| 0 1 | That, my friend, is why The Mighty Swift Radio... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 2 | Spent 20 minutes in an Uber listening to what ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 3 | via The Guardian Guardian front page, Monday ... | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 4 | My real job is being my girlfriends personal U... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 5 | i had a bad drive . i want my refund | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2871 3519 | Top story: Uber broke laws, duped police and s... | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2872 3521 | Top story: Uber broke laws, duped police and s... | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2873 3522 | Top story: Uber broke laws, duped police and s... | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2874 3523 | reading about this uber leak reminded me how f... | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2875 3526 | It's clear that Uber broke laws& behaved a... | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Fonte: Material produzido pelos autores (2024)

Esta matriz foi gerada no pipeline, tendo como base a função demonstrada abaixo, na qual foram utilizadas instâncias da biblioteca Scikit-learn. Essa matriz mostra a frequência das palavras do vocabulário nos comentários do twitter. Além disso, também possibilita saber em quantas frases diferentes as mesmas palavras aparecem, o que traz mais facilidade no momento de identificar padrões entre as frases que possuem os mesmo rótulos.

Um dos principais benefícios desta matriz é poder utilizá-la como input para modelos de classificação, o que leva a solução mais próxima ainda de seu objetivo final.

Após compreender a importância e o funcionamento do modelo Bag of Words, é hora de implementá-lo no projeto da *Nexus*. A aplicação desse modelo permitirá a transformação das frases coletadas em vetores numéricos, uma etapa essencial para a construção de modelos de machine learning. Na Figura a seguir, será apresentado o código responsável por criar a representação Bag of Words a partir dos dados coletados.

Figura 18 - Código da matriz BoW

```

1  def bag_of_words():
2
3  ...
4  Função que aplica o modelo bag of words, transforma as frases em vetores que são as linhas de uma matriz
5
6  Args:
7      None
8
9  Returns:
10     DataFrame : dataframe -> dataframe com as palavras e a frequência de cada uma nas frases
11     ...
12
13 #ler df cleaned_data
14 df = pd.read_csv('cleaned_data.csv')
15
16 #montar o bow considerando que cada linha é uma frase e que as palavras são as colunas
17 vectorizer = CountVectorizer()
18 bow = vectorizer.fit_transform(df["0"])
19 bow_df = pd.DataFrame(bow.toarray(), columns=vectorizer.get_feature_names_out())
20
21 return bow_df

```

Fonte: Material produzido pelos autores (2024)

A função `bag_of_words()` apresentada acima oferece uma implementação eficiente do modelo Bag of Words para o PLN. Ao transformar frases em vetores numéricos, essa abordagem permite a extração de informações importantes dos dados textuais, facilitando a análise e a construção de modelos de machine learning. Integrada em um pipeline de processamento de texto, essa função desempenha um papel crucial na preparação dos dados para aplicações futuras, fornecendo uma representação quantitativa das palavras presentes nos textos.

8.1.4 Naive Bayes

Com o intuito de testar a vetorização, foi aplicado o modelo de classificação *Naive Bayes*. Esse calcula a probabilidade de determinadas entradas serem de classes pré definidas a partir da ocorrência de dadas características comuns apresentadas no conjunto de dados de treino. Essas probabilidades são calculadas da seguinte forma

$$P(A|B) = \frac{P(B|A)}{P(A)} P(B)$$

Onde:

- $P(A|B)$ é a probabilidade de A ocorrer dado que B ocorreu (probabilidade posterior).
- $P(B|A)$ é a probabilidade de B ocorrer dado que A ocorreu (probabilidade de evidência).
- $P(A)$ é a probabilidade de A ocorrer (probabilidade priori).
- $P(B)$ é a probabilidade de B ocorrer.

Com isso, dado a ocorrência de certas palavras em cada frase, o modelo conseguiria, utilizando da matriz gerada pelo BoW, classificar frases como positivas, negativas e neutras. Então, após aplicar o modelo, foi

possível alcançar as seguintes métricas

- Acurácia do teste: 0.69
- Matriz de confusão do teste:

| -1 | 0 | 1 |
|-----------|----------|----------|
| 482 | 126 | 10 |
| 66 | 111 | 12 |
| 12 | 40 | 4 |

- f1 score: 0.46
- precision: 0.47
- recall: 0.47

Contudo, por causa de características do conjunto de dados fornecido, como por exemplo o desbalanceamento dos dados, possuindo 3 vezes mais dados negativos que positivos e neutros somados, o modelo acaba por perder sua eficiência e ficando facilmente enviesado. Isso pois se na maior parte das vezes o algoritmo chegar à conclusão que uma classe é negativa, há muitas chances de estar correto, isto por conta do desbalanceamento da base e não devido às características que realmente foram analisadas.

Considerando o fato do desbalanceamento entre as classes, é possível aplicar algumas técnicas para corrigir esse problemas. Primeiramente, foi aplicado uma técnica de "*oversampling*". Essa consiste na criação de novos dados fictícios a partir de uma biblioteca chamada *Smote*. Após aplicar essa técnica no *dataframe* gerado pelo pré processamento, é possível chegar nas seguintes métricas:

- Acurácia do teste: 0.66
- Matriz de confusão do teste:

| -1 | 0 | 1 |
|-----------|----------|----------|
| 458 | 103 | 45 |
| 45 | 194 | 368 |
| 14 | 39 | 544 |

- f1 score: 0.64
- precision: 0.67
- recall: 0.66

A técnica acima resolve o desbalanceamento, mas a criação de dados sintéticos tende a enviesar o modelo. Com isso, foi aplicado tambem a transformação de classes. Isto é, todos os dados referentes a sentimentos neutros, foram convertidos para sentimentos positivos. Então, obteve-se as seguintes métricas:

- Acurácia do teste: 0.7555040556199305
- Matriz de confusão do teste:

| -1 | 1 |
|-----------|----------|
|-----------|----------|

| -1 | 1 |
|-----|-----|
| 469 | 149 |
| 62 | 183 |

- f1 score: 0.72
- precision: 0.71
- recall: 0.75

Com isso, a melhora foi ainda mais significativa. Contudo, observando a matriz de confusão, o modelo ainda comete muitos erros ao classificar sentimentos positivos e também ainda é possível ver um certo desbalanceamento. Então novamente foi aplicado um *oversampling*, o qual obteve as seguintes métricas.

- Acurácia do teste: 0.81
- Matriz de confusão do teste:

| -1 | 1 |
|-----|-----|
| 429 | 179 |
| 47 | 552 |

- f1 score: 0.81
- precision: 0.82
- recall: 0.81

Com esse ultimo ajuste, foi possível ver uma melhora ainda maior. Contudo, é necessário comparar esse modelo com outros, como por exemplo o *Word2Vec*, para que seja possível analisar sua aplicabilidade.

8.1.4. Conclusão

A análise do desenvolvimento realizado revela que o modelo Bag of Words não se mostra ideal para esta solução específica. Essa conclusão emerge da consideração da escalabilidade, onde surgem desafios significativos com o crescente tamanho do vocabulário. Este aumento, por sua vez, resultaria em uma demanda computacional substancialmente maior, o que não se alinha adequadamente com o valor agregado do projeto. Assim, apesar das vantagens iniciais do Bag of Words na simplificação da representação textual, sua aplicação aqui esbarra em limitações práticas que sugerem a busca por alternativas mais eficientes.

Também se faz importante salientar que o Bag of Words, dado que considera apenas a frequência das palavras, não é capaz de entender o contexto em que estas são utilizadas. Isto abre margens para o modelo ter dificuldades em reconhecer fenômenos como a similaridade semântica, ou ainda criar confusões devido à similaridade léxica de algumas palavras.

Devido a tais fatores, esta não será a técnica de vetorização aplicada na versão final do projeto. Portanto, o grupo *Nexus* propõe-se a buscar outras formas de vetorização que atendam às necessidades do projeto.

8.2. Word2Vec

8.2.1. Introdução

O Word2Vec é uma técnica de processamento de linguagem natural que foi desenvolvida em 2013 por pesquisadores da Google. Com base no artigo disponibilizado no site Geeks for Geeks, o Word2Vec serve para transformar cada palavra em um vetor, que será representado em um espaço vetorial de forma que palavras com contextos semelhantes tenham representações vetoriais próximas. [5]

No Word2Vec existem duas arquiteturas principais, a primeira sendo conhecida como Continuous Bag of Words, ou CBOW, e a outra Skip Gram. O CBOW atua prevendo uma palavra com base nas palavras em seu contexto. Por exemplo, se tivermos a frase "O gato está em cima do telhado" e nela a palavra "em" for omitida, o modelo tentaria prevê-la usando as palavras ao redor desta como entrada. Já no Skip Gram, o processo também acontece da forma inversa, onde é usada uma palavra para prever suas palavras de contexto. Utilizando a mesma frase de exemplo, o modelo pegaria as palavras "Gato" e "Telhado" e poderia prever as palavras "está", "em", "cima".

Em conclusão, o Word2Vec serve para representar palavras como vetores que capturam seus significados, o que é muito importante para várias tarefas de processamento de linguagem natural, como a análise de sentimentos realizada neste projeto. Ao transformar palavras em vetores que conseguem explorar o contexto, o Word2Vec facilita a detecção de diferenças semânticas e melhora a precisão em diversas aplicações de processamento de linguagem natural.

8.2.2. Método

Embedding Layer

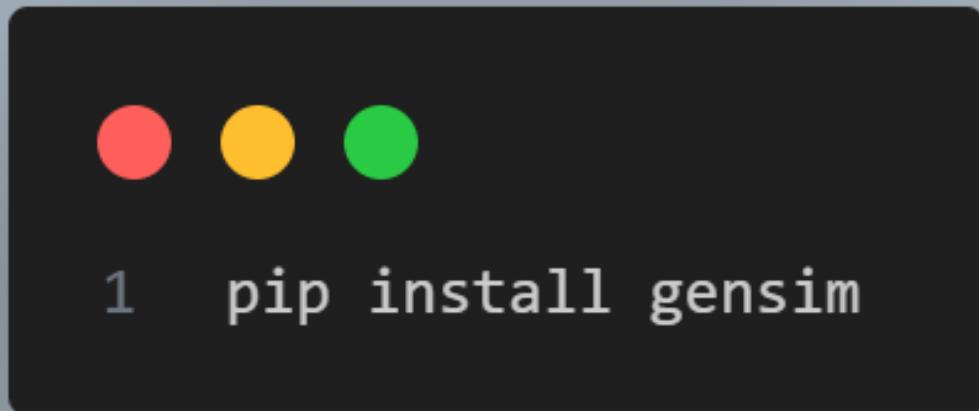
- Entendendo Embedding Layer

Uma camada de Embedding, ou camada de incorporação, é uma representação de dados densa e de baixa dimensionalidade, que é comumente usada em processamento de linguagem natural (NLP) para transformar entradas textuais grandes e esparsas (como índices de palavras) em vetores contínuos e mais compactos. Essa camada aprende a mapear um índice de entrada para um vetor denso que representa aquela entrada em um espaço contínuo. Isso facilita a modelagem de relacionamentos complexos entre palavras em tarefas como classificação de texto, tradução automática e geração de texto.

- Como montar Word2Vec com Embedding Layer

Para montar um modelo Word2Vec utilizando camadas de Embedding em Python, a biblioteca mais comum é o [gensim](#). Primeiro, é necessário instalar o [gensim](#), caso ainda não esteja instalado:

Figura 19 - Código de instalação do gensim



Fonte: Material produzido pelos autores (2024)

Após a instalação, é possível criar um modelo Word2Vec com uma camada de Embedding da seguinte maneira:

Figura 20 - Código para criar um modelo Word2Vec

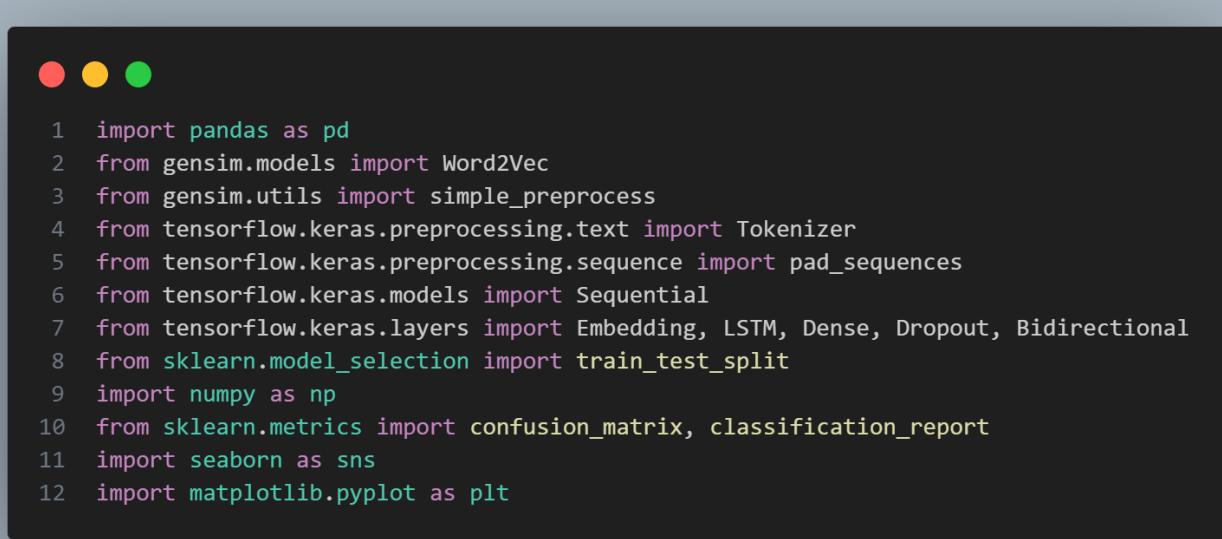
```
1 from gensim.models import Word2Vec
2
3 sentences = [["hello, world"], ["hello, gensim"]]
4 model = Word2Vec(sentences, vector_size=100, windows=5, min_count=1 workers=4)
```

Fonte: Material produzido pelos autores (2024)

Neste exemplo, `vector_size` define a dimensão dos vetores de Embedding, `window` define o tamanho da janela de contexto e `min_count` especifica o número mínimo de ocorrências de uma palavra para que ela seja considerada no treinamento.

Neste segmento de nossa documentação, apresentamos a configuração inicial das bibliotecas e estruturas essenciais para a incorporação do modelo Word2Vec com camadas de Embedding, como mostrado na Figura abaixo. Este setup é crucial para garantir que as representações vetoriais das palavras sejam aprendidas de forma eficaz e eficiente.

Figura 21 - Instalações Embedding Layer



```
1 import pandas as pd
2 from gensim.models import Word2Vec
3 from gensim.utils import simple_preprocess
4 from tensorflow.keras.preprocessing.text import Tokenizer
5 from tensorflow.keras.preprocessing.sequence import pad_sequences
6 from tensorflow.keras.models import Sequential
7 from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
8 from sklearn.model_selection import train_test_split
9 import numpy as np
10 from sklearn.metrics import confusion_matrix, classification_report
11 import seaborn as sns
12 import matplotlib.pyplot as plt
```

Fonte: Material produzido pelos autores (2024)

A estrutura de código que detalhamos anteriormente nos permite não apenas implementar o Word2Vec com eficiência, mas também explorar profundamente o potencial das camadas de Embedding em tarefas de PLN. Com essas ferramentas, podemos avançar para a fase de modelagem e análise, garantindo análises mais precisas e insights mais profundos derivados dos dados textuais.

- Definir características específicas de funcionamento

As camadas de Embedding, especialmente em modelos como o Word2Vec, possuem características únicas:

- Treinamento Contextual: O Embedding é ajustado de forma que palavras com contextos semelhantes resultem em vetores próximos.
- Redução de Dimensionalidade: Transforma representações esparsas e de alta dimensão em vetores densos e de baixa dimensão.
- Aprendizado Automático: A camada de Embedding é treinada automaticamente junto com o modelo de rede neural para minimizar a função de perda do problema específico, ajustando-se para capturar nuances no uso das palavras.

- Utilizando o Modelo Pré-Treinado do Word2Vec

Uma maneira eficiente de obter vetores de palavras é utilizando um modelo pré-treinado de Word2Vec, como o modelo treinado no corpus do Google News. Este modelo contém vetores para cerca de 3 milhões de palavras e frases e pode ser facilmente carregado utilizando a biblioteca [gensim](#).

Figura 22 - Importação do modelo pré-treinado Google News

```
1 from gensim.models import KeyedVectors
2
3 model_path = './data/GoogleNews-vectors-negative300.bin.gz'
4
5 model = KeyedVectors.load_word2vec_format(model_path, binary=True)
6
7 vector = model['word']
8 print(vector)
```

Fonte: Material produzido pelos autores (2024)

Este código carrega os vetores de palavras que podem ser utilizados para transformar texto em representações vetoriais, facilitando a aplicação em diversas tarefas de PLN, incluindo a análise de sentimentos.

- Como contribui com Word2Vec

Combinar o Word2Vec com a **Embedding Layer** é uma estratégia eficaz para análises de conjuntos de dados de diversos tamanhos. Essa combinação é particularmente útil para capturar nuances semânticas complexas, resultando em modelos de classificação de sentimentos que podem atingir métricas significativas. Os resultados deste projeto, como mostrado a seguir, ilustram a eficácia dessa abordagem:

- Acurácia: 0.76
- f1_score: 0.85 (Negativo), 0.32 (Neutro)
- Precisão: 0.77 (Negativo), 0.67 (Neutro)
- Recall: 0.96 (Negativo), 0.21 (Neutro)
- Média macro: 0.72 (Precisão), 0.59 (Recall), 0.59 (f1_score)
- Média ponderada: 0.74 (Precisão), 0.76 (Recall), 0.71 (f1_score)

Naive Bayes

- Entendendo Naive Bayes

O modelo em questão baseia-se na lógica de que os atributos ou características fornecidos a ele seguem uma distribuição normal. As probabilidades de ocorrência das variáveis presentes no conjunto de dados de treinamento do modelo são calculadas para cada classe.

- Como montar Word2Vec com Naive Bayes

Para construir um modelo de *Gaussian Naive Bayes* com Word2Vec, é necessário seguir alguns passos. Primeiramente, definem-se algumas funções auxiliares para a vetorização de cada texto do conjunto de treino, o qual já deve ter passado pelo *pipeline* de pré-processamento. Para calcular o vetor de cada texto, utiliza-se um modelo de vetorização de palavras do Google com 300 dimensões. A partir disso, calcula-se a média dos vetores para as frases. Após essa etapa, é necessário treinar o modelo com um conjunto de treino que corresponde a 70% do tamanho total do conjunto de dados.

- Definir características específicas de funcionamento

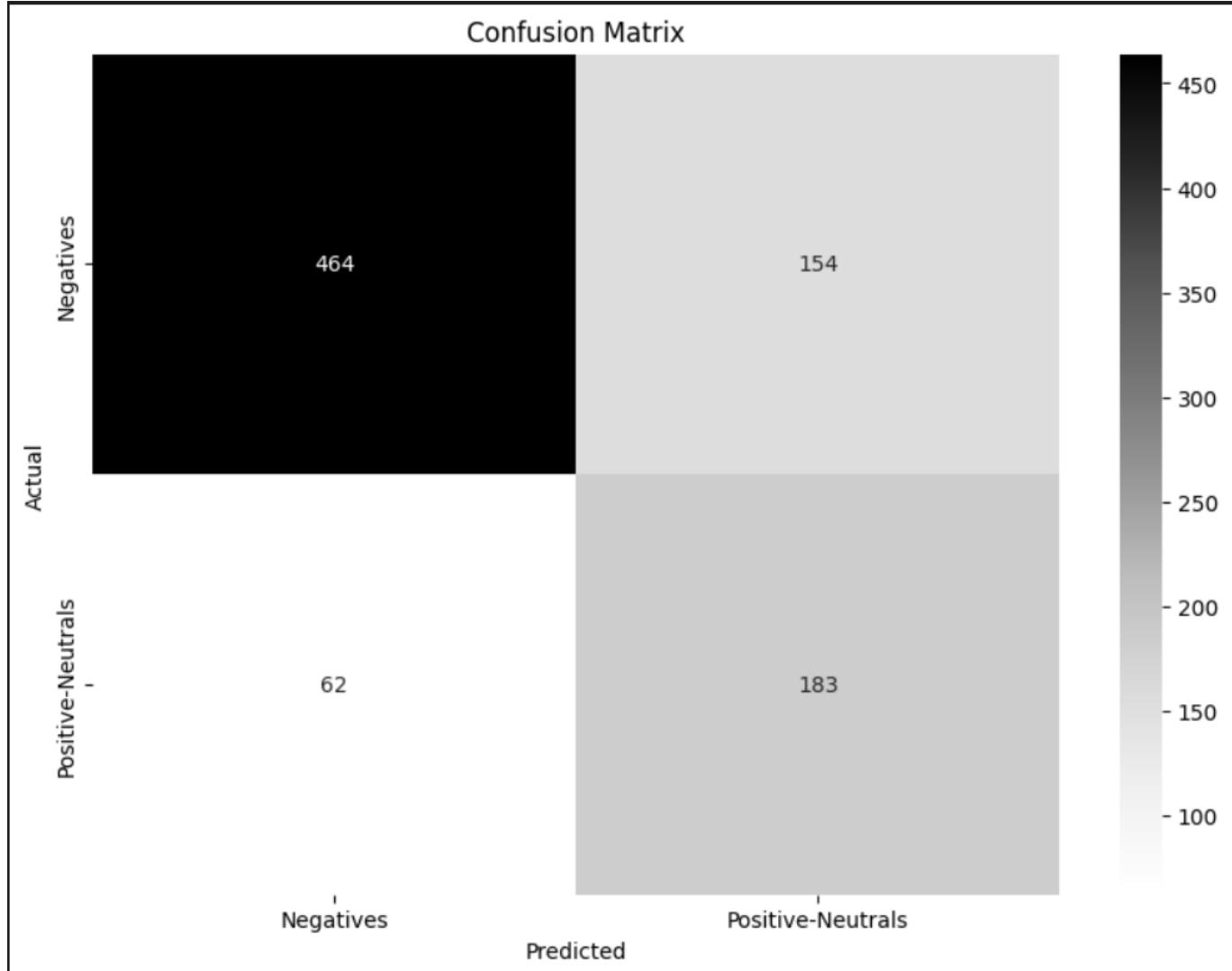
Ressalta-se que o modelo assume que cada característica é independente das demais. Por exemplo, se classificarmos um cachorro com as características de ser grande e latir, para o modelo, o fato de um cachorro latir não tem relação com o seu porte. Essa característica do modelo é conhecida como suposição de independência ou suposição "ingênua".

- Como contribui com Word2Vec

Utilizar do Word2Vec juntamente ao *Gaussian Naive Bayes* pode ser uma boa estratégia para casos de análises com conjuntos de dados não muito grandes e que estejam平衡ados. Ele se torna uma resposta rápida para modelos de classificação de sentimentos e que consegue alcançar métricas como as que foram alcançadas nesse projeto, as quais serão mostradas a seguir:

- **Teste**
 - Acurácia: 0.7497
 - f1_score: 0.7200
 - Precisão: 0.7126
 - Recall: 0.7489
- **Treino**
 - Acurácia: 0.7690
 - f1_score: 0.7487
 - Precisão: 0.7418
 - Recall: 0.7737

Figura 23 - Matriz Confusão do Gaussian Naive Bayes



Fonte: Material produzido pelos autores (2024)

Analizando as métricas alcançadas, é possível chegar a algumas conclusões acerca do modelo. Primeiramente, não parece haver *overfitting*, pois a diferença entre as métricas de treino e teste não é muito grande. Normalmente, quando há *overfitting* ou *underfitting*, o conjunto de treino apresenta uma acurácia muito maior ou menor em comparação com o conjunto de teste. Além disso, o modelo está acertando uma grande parte dos comentários negativos, o que pode ser considerado um sucesso, levando em consideração o foco do projeto.

Olhando para as outras métricas, podemos observar que a **precision** (precisão) e o **recall** estão relativamente equilibrados, indicando que o modelo não está favorecendo demasiadamente nem os falsos positivos nem os falsos negativos. O **F1 score**, que é a média harmônica entre precisão e recall, também está em um valor razoável, indicando um bom equilíbrio entre essas duas métricas. Isso sugere que o modelo está conseguindo lidar de forma adequada com as classes positiva e negativa.

Random Forest

- Entendendo Random Forest

O modelo constrói várias árvores de decisão durante o treinamento e combina suas previsões para obter uma previsão final mais robusta e precisa. Cada árvore de decisão é treinada em uma amostra aleatória dos dados de treinamento e faz previsões independentes. Em seguida, a previsão final é determinada pela votação, no caso de classificação, das previsões individuais das árvores. Ou seja, cada árvore, determinada pelo treino, analisa a entrada do modelo e então toma uma decisão para a classificação, no caso do projeto apresentado, entre "Negativo" e "Positivo-Neutro".

- Como montar Word2Vec com Random Forest

Para construir um modelo de *Random Forest* ligado a um modelo de *Word2Vec* é preciso seguir alguns passos. Primeiramente, deve-se fazer a vetorização dos textos de entrada, como realizado no modelo de *Naive Bayes* da seção 3.2.2. Em seguida, os dados são separados entre conjuntos de teste e treino. Com isso, inicia-se o treino do modelo, para então gerar as "árvores de decisão". Nestas serão registrados os possíveis padrões dos conjuntos de dados que serão utilizados para que cada árvore avalie dados futuros e então realizar novas classificações.

- Definir características específicas de funcionamento

O segredo do poder das Árvores de Random Forest reside em dois pilares:

- **Bagging (Bootstrap Aggregating):** A construção de cada árvore de decisão ocorre de forma independente, utilizando um subconjunto aleatório dos dados de treinamento, com reposição. Isso significa que alguns dados podem ser utilizados em múltiplas árvores, enquanto outros podem não ser utilizados em nenhuma. Essa estratégia garante a diversidade entre as árvores, evitando que se tornem muito semelhantes entre si e aprendam com os mesmos padrões dos dados.
- **Seleção Aleatória de Características:** Em cada nó da árvore de decisão, em vez de considerar todas as variáveis possíveis para a divisão, o algoritmo seleciona apenas um subconjunto aleatório delas. Essa estratégia, conhecida como Feature Randomness, reduz a correlação entre as árvores e evita que o modelo se sobreajuste aos dados de treinamento, o que pode levar a um mau desempenho em dados novos e desconhecidos.

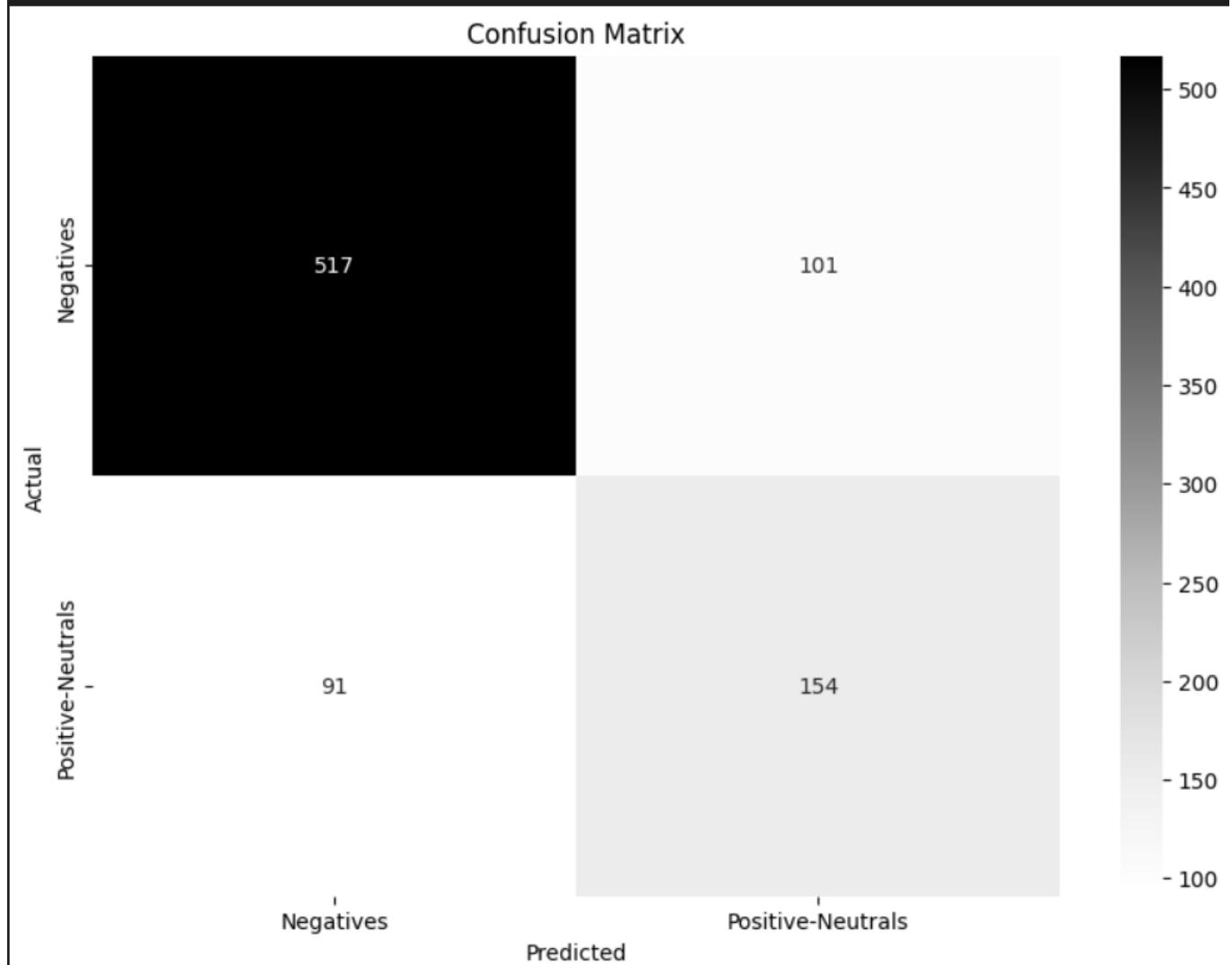
- Como contribui com Word2Vec

O modelo *Random Forest* se adaptaria bem ao fato de termos entradas de dados muito grandes. Por exemplo, no modelo de *Word2Vec* utilizado, são encontrados 300 dimensões de vetores para cada palavra, o *Random Forest* conseguiria analisar diversos conjuntos de vetores a partir de suas diversas árvores de decisão. Com isso, ainda considerando que a ubir terá diversas entradas de comentários todos os dias, o modelo *Random Forest* se adaptaria bem ao fluxo de dados. Mas pode ser bem sensível a conjuntos de dados muito pequenos, como o utilizado no treino, chegando a métricas como as seguintes.

- **Teste**
 - Acurácia: 0.7845
 - f1_score: 0.7323
 - Precisão: 0.7348
 - Recall: 0.7300
- **Treino**
 - Acurácia: 0.9930
 - f1_score: 0.9918
 - Precisão: 0.9927

- Recall: 0.9910
- Observação: Considerável overfitting, pois as métricas no treino são significativamente maiores que no teste.

Figura 24 - Matriz Confusão do Modelo Random Forest



Fonte: Material produzido pelos autores (2024)

Agora, a partir de uma análise das métricas apresentadas, é possível chegar a algumas conclusões. Primeiramente, o modelo apresenta métricas de treino muito altas, enquanto as métricas de teste estão mais baixas. Isso mostra um *overfitting* do modelo, pois se as métricas de treino estão altas mas as de teste não, isso significa que o modelo está se ajustando excessivamente aos dados de treino e não generalizando para outras entradas.

Logistic Regression

- Entendendo Logistic Regression

A Regressão Logística é uma técnica de aprendizado de máquina usada para prever a probabilidade de uma observação pertencer a uma determinada categoria. Para isso a Regressão Logística usa uma função

logística para transformar combinações lineares das variáveis explicativas em probabilidades entre 0 e 1. Essas probabilidades são interpretadas como a chance de pertencer a uma classe específica. Então, por último o modelo tenta melhorar suas previsões utilizando da técnica de máxima verossimilhança.

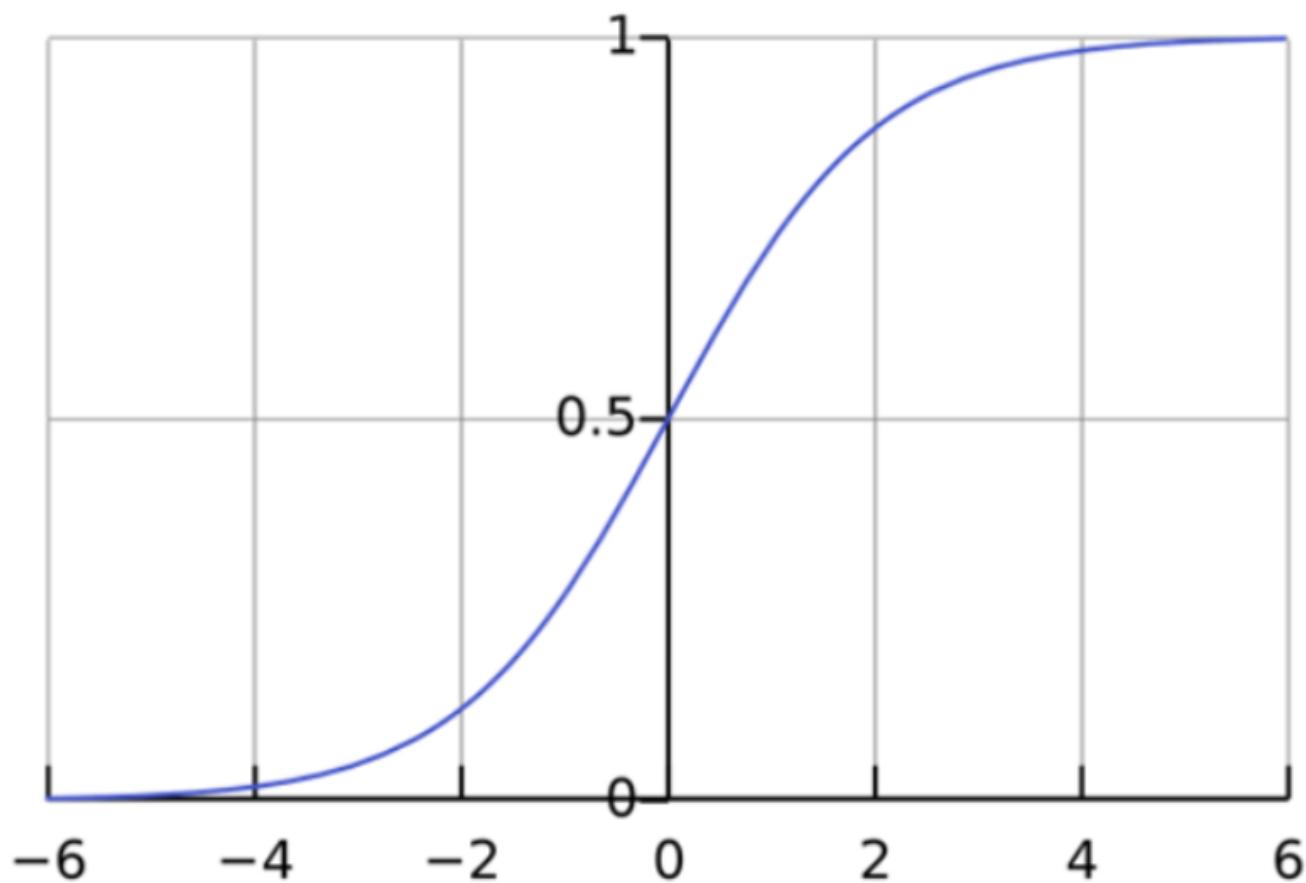
- Como montar Word2Vec com Logistic Regression

Com isso, é possível começar a implementação do modelo com Word2Vec. Primeiramente, utiliza-se as mesmas funções auxiliares citadas na seção 3.2.2 para vetorização de textos. Então, é possível vetorizar o conjunto de dados de entrada para que o modelo possa interpretar os vetores das frases. Após, inicia-se o treino e teste do modelo.

- Definir características específicas de funcionamento

Para entender melhor, a função logística é uma curva em forma de S que mapeia qualquer valor real para o intervalo entre 0 e 1. Essa função é fundamental na Regressão Logística, pois transforma a soma ponderada das variáveis explanatórias (através de uma combinação linear) em uma probabilidade que reflete a chance de pertencer a uma classe ou categoria.

Figura 25 - Curva logística



Fonte: [AWS](#)

- Como contribui com Word2Vec

A aplicação de Regressão Logística em conjunto com modelos de Word2Vec, especialmente aqueles que geram vetores de 300 dimensões, pode ser uma abordagem poderosa para diversas tarefas de

processamento de linguagem natural (PLN) isso pois o Word2Vec transforma palavras em vetores de alta dimensão (300 dimensões, por exemplo) de tal forma que palavras com significados e contextos semelhantes estão próximas no espaço vetorial. Isso captura nuances semânticas de maneira eficiente, permitindo que modelos subsequentes, como a Regressão Logística, se beneficiem de representações de palavras informativas. Por fim, considerando esses fatores, foi possível chegar nas seguintes métricas:

- **Teste**

- Acurácia: 0.7787
- f1_score: 0.7219
- Precisão: 0.7273
- Recall: 0.7174

- **Treino**

- Acurácia: 0.8137
- f1_score: 0.7802
- Precisão: 0.7822
- Recall: 0.7783
- Observação: Diferença menor entre os resultados de treino e teste, indicando um modelo mais equilibrado.

O modelo apresentou bom desempenho, evidenciando uma capacidade de generalização adequada ao evitar o problema de overfitting, ou seja, ele não decorou os dados do conjunto de treino. As métricas de precisão (precision) e recall estão relativamente equilibradas, indicando que o modelo mantém uma boa relação entre falsos positivos e falsos negativos. O F1_score, que é a média harmônica entre precisão e recall, também apresenta um valor satisfatório, demonstrando um bom equilíbrio entre essas duas métricas. Dessa forma, podemos concluir que o modelo está lidando adequadamente com as classes positiva e negativa.

Para testar e treinar os 4 modelos mostrados acima, foram definidas várias métricas. As métricas escolhidas são, como já descrito, **acurácia**, **F1_score**, **precision** (precisão) e **recall**. A seguir, são apresentadas as fórmulas matemáticas de cada métrica e as justificativas para a escolha de cada uma:

Acurácia (Accuracy)

Fórmula:

$$[\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões}}]$$

A acurácia mede a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. É uma métrica básica que fornece uma visão geral da eficácia do modelo, mas pode ser enganosa se os dados estiverem desbalanceados.

F1_score

Fórmula:

$$[f1_score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}]$$

O f1_score é a média harmônica da precisão e do recall, equilibrando os dois. É particularmente útil quando temos uma distribuição desigual entre as classes e queremos assegurar que tanto a precisão quanto o recall sejam considerados.

Precisão (Precision)

Fórmula:

$$[\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}]$$

A precisão mede a proporção de verdadeiros positivos entre todas as instâncias que foram previstas como positivas. Esta métrica é importante quando o custo de um falso positivo é alto.

Recall

Fórmula:

$$[\text{Recall} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}]$$

O Recall mede a proporção de verdadeiros positivos entre todas as instâncias que realmente são positivas. É crucial quando o custo de um falso negativo é alto.

As métricas foram escolhidas para proporcionar uma avaliação abrangente da performance dos modelos de classificação. A **acurácia** oferece uma visão geral do desempenho, enquanto o **f1_score**, **precisão** e **recall** fornecem insights mais detalhados sobre a capacidade do modelo em lidar com classes desbalanceadas e a importância relativa dos falsos positivos e falsos negativos. A escolha dessas métricas permite identificar o melhor modelo para a análise de comentários da Uber, equilibrando a necessidade de previsões precisas com a capacidade de identificar corretamente os comentários negativos, neutros e positivos.

8.2.3. Resultados

Os modelos avaliados são: Gaussian Naive Bayes, Random Forest, Logistic Regression e Embedding Layer. Abaixo, apresentamos uma tabela comparativa das métricas de desempenho para cada modelo.

Tabela 7 - Comparação de Modelos

| Modelo | Acurácia | Precision | Recall | F1-Score | Uso de Memória | Tempo de Execução |
|----------------------|----------|-----------|--------|----------|----------------|-------------------|
| Gaussian Naive Bayes | 0.75 | 0.71 | 0.75 | 0.72 | Baixo | Rápido |
| Random Forest | 0.78 | 0.73 | 0.74 | 0.73 | Alto | Moderado |
| Logistic Regression | 0.78 | 0.73 | 0.72 | 0.72 | Moderado | Rápido |
| Embedding Layer | 0.76 | 0.74 | 0.76 | 0.71 | Muito Alto | Lento |

Fonte: Material produzido pelos autores (2024)

Análise de Métricas

- **Acurácia:** É uma métrica fundamental para avaliar a performance geral dos modelos, sendo particularmente útil em contextos onde as classes são igualmente importantes. Os modelos Gaussian Naive Bayes e Logistic Regression apresentaram valores similares e relativamente altos de acurácia, o que indica um bom desempenho geral.
- **Precision:** É crucial quando o custo de falsos positivos é elevado. Por exemplo, na análise de sentimentos, classificar um comentário negativo como positivo pode levar a ignorar um cliente insatisfeito. O modelo Embedding Layer, apesar de ter um uso de memória e tempo de execução maior, apresenta uma **precision** ligeiramente superior aos demais modelos, tornando-o adequado para cenários que exigem alta precisão.
- **Recall:** É importante quando os falsos negativos representam um risco maior. No contexto da análise de sentimentos para a Uber, perder um comentário negativo pode significar não identificar um problema sério. Nesse aspecto, os modelos Gaussian Naive Bayes e Embedding Layer se destacam.
- **F1-Score:** É útil quando se busca um equilíbrio entre Precision e Recall, especialmente em classes desbalanceadas. Os modelos apresentam valores de F1-Score bastante próximos, indicando um equilíbrio geral entre precision e recall.

Conclusão sobre Métricas

Para a *Nexus*, a escolha do modelo ideal depende do equilíbrio entre acurácia, precisão, recall e eficiência computacional. Considerando o uso de memória e tempo de execução, o modelo Gaussian Naive Bayes oferece uma boa relação custo-benefício. No entanto, para situações que exigem alta precisão e recall, o modelo Embedding Layer pode ser mais adequado, apesar de seu maior consumo de recursos.

(É importante ressaltar que, logo abaixo, no tópico 9., foi feita a escolha do modelo definitivo, pois todos eles foram melhorados com o passar das semanas de desenvolvimento do projeto.)

8.2.4. Conclusão

A partir das análises realizadas acima, a equipe *Nexus* optou por utilizar o modelo Word2Vec em detrimento do Bag of Words (BoW) para a análise de sentimentos dos comentários da Uber. Esta decisão se fundamenta nas vantagens que o Word2Vec oferece em termos de métricas de desempenho e compreensão contextual dos dados.

O modelo Word2Vec, ao contrário do BoW, captura o contexto das palavras dentro de um corpus maior, gerando vetores de palavras que refletem as relações semânticas e sintáticas. Por exemplo, no Word2Vec, palavras com significados ou contextos semelhantes possuem representações vetoriais próximas, o que é uma vantagem significativa para a interpretação de sentimentos, onde as diferenças do contexto podem alterar completamente o sentido de uma avaliação.

Em termos de desempenho, o Word2Vec supera o BoW principalmente na gestão de vocabulários grandes e na capacidade de lidar com palavras novas ou raras sem aumentar muito a dimensionalidade do espaço de características ou até mesmo mantendo-a, o que é uma limitação conhecida do BoW. Além disso, o Word2Vec oferece flexibilidade significativa na incorporação de novas palavras e adaptação a novos contextos.

sem a necessidade de reprocessar todo o corpus, um aspecto crucial para a dinâmica de dados da Uber, onde novos termos e gírias podem surgir continuamente.

Por fim, a utilização da técnica de vetorização Word2Vec proporcionou resultados melhores que a utilização da técnica de Bag of Words. Isto tornou-se notório devido a fatores como a ausência de matrizes esparsas no output do Word2Vec, o que ocorria no Bag of Words e trazia prejuízos no processamento destes dados, especialmente no que diz respeito à sua classificação. Além disso, com o Word2Vec também tornou-se viável a identificação da similaridade semântica, o que contribui para uma melhor análise dos sentimentos apresentados nos comentários, dado que agora faz-se possível compreender palavras que, apesar de diferentes, transmitem o mesmo sentido.

Portanto, a escolha do Word2Vec é estratégica para o projeto da Uber, permitindo uma análise mais profunda e precisa dos sentimentos expressos pelos usuários, além de proporcionar uma base mais robusta para o desenvolvimento futuro do modelo de análise de sentimentos. A implementação deste modelo visa não apenas melhorar a precisão da classificação dos sentimentos, mas também aprofundar o entendimento da empresa sobre as experiências e percepções dos seus clientes, um fator crítico para o aprimoramento contínuo dos serviços oferecidos pela Uber.

9. Modelo Escolhido

Depois de analisar todos os modelos e suas métricas, foi decidido o melhor deles para se usar no projeto. Porém eles foram sendo aperfeiçoados pelo grupo, o que fez com que as métricas mudassem um pouco. Uma das formas de aprimoramento foi o **Oversampling**.

Oversampling é uma técnica usada em Machine Learning para lidar com o problema de desequilíbrio de classes em bases de dados. No caso do projeto atual, ele se dá na classe de sentimentos negativa da base de dados, onde inicialmente havia 198 dados positivos, 688 neutros, e 2010 negativos. Com uma base de dados majoritariamente negativa, o modelo pode se tornar tendencioso (o que, de fato, ocorreu), resultando em uma pior performance na predição das outras classes de sentimentos. Assim, a solução encontrada pelo grupo *Nexus* para resolver este problema foi adotar o Oversampling, gerando novos comentários nas classes de comentários positivos e neutros, chegando a um número de: 1343 positivos, 988 neutros e os mesmos 2010 comentários negativos, melhorando o ambiente de treinamento do modelo.

A aplicação do Oversampling pode ajudar o modelo classificatório de diversas maneiras. Em primeiro lugar, quando se equilibra a proporção das classes, o modelo consegue entender melhor as características das classes com menos dados, o que melhora a sua capacidade de fazer previsões para essas classes. Além disso, o Oversampling ajuda a reduzir o viés do modelo, visto que após a aplicação, um comentário do qual o modelo não tem certeza, não será caracterizado como um sentimento negativo simplesmente pela adaptação do modelo a grande quantidade de dados negativos.

Após aplicar o Oversampling e melhorar os modelos de outras maneiras, as métricas ficaram como mostra a tabela a seguir:

Tabela 8 - Comparação de Modelos Atualizados

| Modelo | Acurácia | Precision | Recall | F1-Score | Uso de Memória | Tempo de Execução |
|--------|----------|-----------|--------|----------|----------------|-------------------|
|--------|----------|-----------|--------|----------|----------------|-------------------|

| Modelo | Acurácia | Precision | Recall | F1-Score | Uso de Memória | Tempo de Execução |
|----------------------|----------|-----------|--------|----------|------------------|-------------------|
| Gaussian Naive Bayes | 0.83 | 0.83 | 0.82 | 0.82 | Baixo | Alto |
| Random Forest | 0.78 | 0.73 | 0.74 | 0.73 | Alto | Moderado |
| Logistic Regression | 0.86 | 0.86 | 0.86 | 0.86 | Baixo a moderado | Rápido |
| Embedding Layer | 0.83 | 0.83 | 0.82 | 0.82 | Muito Alto | Lento |

Fonte: Material produzido pelos autores (2024)

Com base na análise das métricas apresentadas na tabela acima, o *Nexus* optou por utilizar o modelo de **Regressão Logística** (Logistic Regression). Esta decisão foi tomada após uma cuidadosa consideração das métricas de desempenho e das características computacionais dos modelos. Entre os modelos avaliados, a Regressão Logística apresentou a melhor acurácia (0.86), indicando uma maior precisão nas previsões corretas. Além disso, exibiu uma precisão de 0.86, demonstrando eficácia na identificação correta das instâncias positivas, e um recall de 0.86, evidenciando eficiência na recuperação de todas as instâncias positivas. O F1-Score, também de 0.86, confirmou a robustez do modelo, equilibrando precisão e sensibilidade.

Outro fator decisivo foi a simplicidade e interpretabilidade do modelo. A Regressão Logística é um modelo linear que é fácil de interpretar e explicar, tanto para cientistas de dados quanto para stakeholders não técnicos. A transparência do modelo facilita a compreensão dos fatores que influenciam as previsões.

Considerando essas vantagens, a Regressão Logística se destaca como o modelo mais equilibrado e eficiente, tanto em termos de métricas de desempenho quanto de requisitos computacionais. Assim, a Regressão Logística foi adotada como o modelo preferido para essa aplicação, garantindo uma solução robusta e eficiente para a solução.

10. API

10.1. Rotas de Limpeza

Os Endpoints de limpeza são dois: [/csv](#) (Figura 26) e [/tweet](#) (Figura 27), os quais recebem, respectivamente, uma base de dados em formato `.csv` e uma string. A função de ambos dentro da API é receber os dados crus e tratá-los para que, posteriormente, possam ser transformados em vetores de maneira adequada. Esta limpeza dos dados é realizada através da tokenização dos textos e da posterior remoção de tokens considerados Stop Words - palavras que não agregam valor semântico às frases - como também de links presentes nos textos, caso sejam encontrados. Após este processo, os tokens restantes são novamente elencados de modo a constituir novos textos limpos.

Figura 26 - Rota de Limpeza de Arquivos

```

1  from flask import Flask, Blueprint, jsonify, request, send_file
2  from services.helper_clean_data import carrega_dados, pipeline
3  import io
4  import pandas as pd
5  import spacy
6
7  clean_data_bp = Blueprint('clean_data', __name__)
8
9  @clean_data_bp.route("/csv", methods=['POST'])
10 def clean_csv():
11
12     try:
13         if 'file' not in request.files:
14             return jsonify({'message': 'No file part in the request'}), 400
15
16         file = request.files['file']
17
18         if file.filename == '':
19             return jsonify({'message': 'No selected file'}), 400
20
21         file_stream = io.BytesIO(file.read())
22         file_stream.seek(0)
23
24         data = pd.read_csv(file_stream)
25         nlp = spacy.load("en_core_web_sm")
26         cleaned_data = []
27
28         for row in data.iterrows():
29             cleaned_text = pipeline(nlp, row)
30             cleaned_data.append(cleaned_text)
31
32         df = pd.DataFrame(cleaned_data, columns=['cleaned_text'])
33
34         output = io.BytesIO()
35         df.to_csv(output, index=False)
36         output.seek(0)
37
38         return send_file(output, mimetype='text/csv', as_attachment=True, download_name='cleaned-data.csv')
39
40     except Exception as e:
41         return jsonify({'message': str(e)}), 400

```

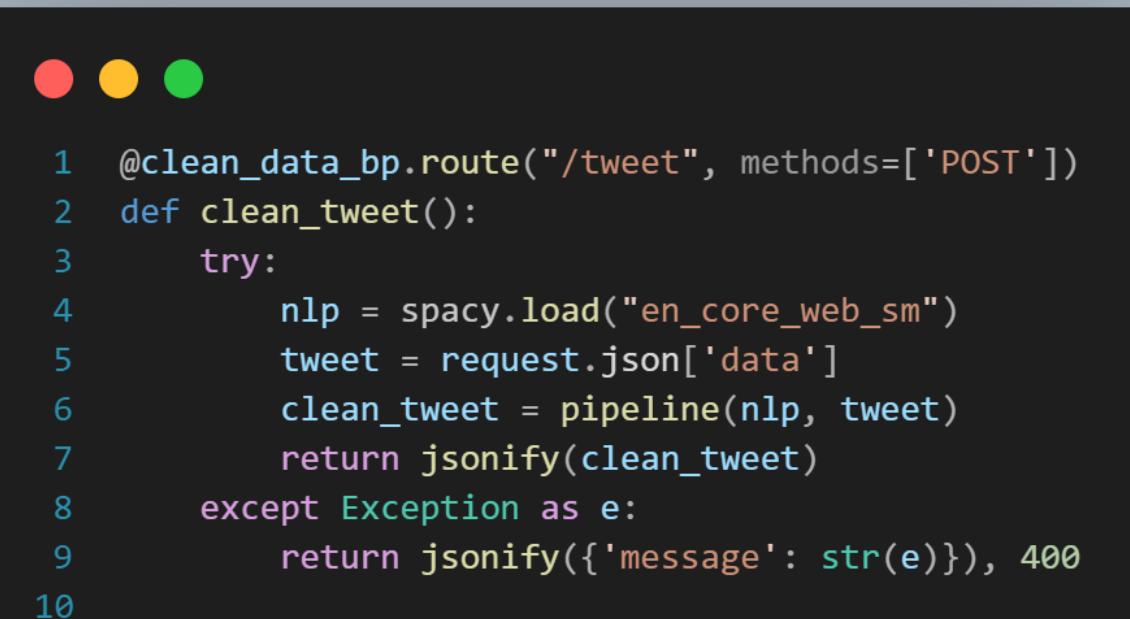
Fonte: Material produzido pelos autores (2024)

Na Figura acima, é possível visualizar a utilização do método `Blueprint()` da biblioteca Flask para a criação de um *Blueprint* chamado `clean_data`, o qual permite a organização do código em componentes modulares. Assim, todas as rotas de limpeza utilizam este mesmo blueprint, dado que compõem o mesmo grupo de Endpoints. Após isso, é definido o Endpoint `/csv` com o método 'POST' e, sempre que este for chamado, a função `clean_csv()`, que recebe um arquivo `.csv` com dados brutos e retorna outro arquivo `.csv` com os dados tratados, será executada. Tal função verifica se houve o input de algum arquivo. Caso não haja ou o nome do arquivo esteja vazio, é retornado um JSON com uma resposta de erro. Após isso, o método `io.BytesIO()` é utilizado para criar um fluxo de bytes em memória, e o arquivo `.csv` recebido é carregado em um DataFrame da biblioteca Pandas.

Posteriormente a este processo, inicia-se de fato o tratamento dos dados. A biblioteca spaCy (para processamento de linguagem natural) é carregada, e a função auxiliar `pipeline()`, importada de outro arquivo através do comando `from services.helper_clean_data import carrega_dados`, pipeline, é

executada para limpar os dados, conforme já referenciado. Os novos textos pré-processados são então armazenados em uma lista que, ao fim das iterações, é transformada em um novo DataFrame. Este, por sua vez, é salvo em um arquivo `.csv`, o qual é enviado de volta na resposta como um anexo. Caso ocorra algum tipo de exceção durante o processo, uma mensagem de erro é retornada como resposta JSON com status HTTP 400.

Figura 27 - Rota de Limpeza de Tweets



```
1  @clean_data_bp.route("/tweet", methods=['POST'])
2  def clean_tweet():
3      try:
4          nlp = spacy.load("en_core_web_sm")
5          tweet = request.json['data']
6          clean_tweet = pipeline(nlp, tweet)
7          return jsonify(clean_tweet)
8      except Exception as e:
9          return jsonify({'message': str(e)}), 400
10
```

Fonte: Material produzido pelos autores (2024)

No que diz respeito à rota `/tweet`, ela compõe um Endpoint que utiliza o método POST e executa a função `clean_tweet()` sempre que é chamado. Este Endpoint age similarmente ao visto na Figura 26. No entanto, ao invés de receber um arquivo `.csv`, ele recebe um JSON com um único texto, que é o tweet a ser tratado. Para tratar o tweet recebido, é executada a função `pipeline()`, a mesma função de limpeza de dados executada e explicada na rota anterior. Após a limpeza do tweet, este é devolvido em um JSON. Caso ocorra algum erro durante este processo, é retornada uma resposta JSON com uma mensagem de exceção e status HTTP 400.

Ambas as rotas de limpeza mostram-se essenciais para que a análise de sentimentos possa ser realizada através da API. Isto porque estas representam uma etapa importantíssima do tratamento dos dados, que é a sua limpeza, a qual possibilita que, posteriormente, sejam gerados vetores mais coerentes e eficientes para classificação de sentimentos.

10.2. Rota de Vetorização

A rota de vetorização é essencial para o sucesso das operações analíticas na API, pois converte textos limpos em representações vetoriais numéricas. Esses vetores são gerados utilizando o modelo Word2Vec, que captura eficazmente os elementos semânticos dos dados em 300 dimensões distintas. Este detalhamento dimensional permite uma análise aprofundada e precisa dos sentimentos.

Figura 28 - Rota de Vetorização



```
1 from flask import Flask, Blueprint, jsonify, request
2 import numpy as np
3
4 vectorize_bp = Blueprint('vectorize_bp', __name__)
5
6 def vectorize_text(text):
7     return np.random.rand(300)
8
9 @vectorize_bp.route('/vectorize', methods=['POST'])
10 def vectorize_text_route():
11     try:
12         text = request.json['text']
13         vector = vectorize_text(text) # Função retorna um numpy array.
14         return jsonify({'vector': vector.tolist()}) # Converta o numpy array para lista, se necessário.
15     except Exception as e:
16         return jsonify({'message': str(e)}), 400
17
18 app.register_blueprint(vectorize_bp, url_prefix='/api')
19
20 if __name__ == '__main__':
21     app.run(debug=True)
```

Fonte: Material produzido pelos autores (2024)

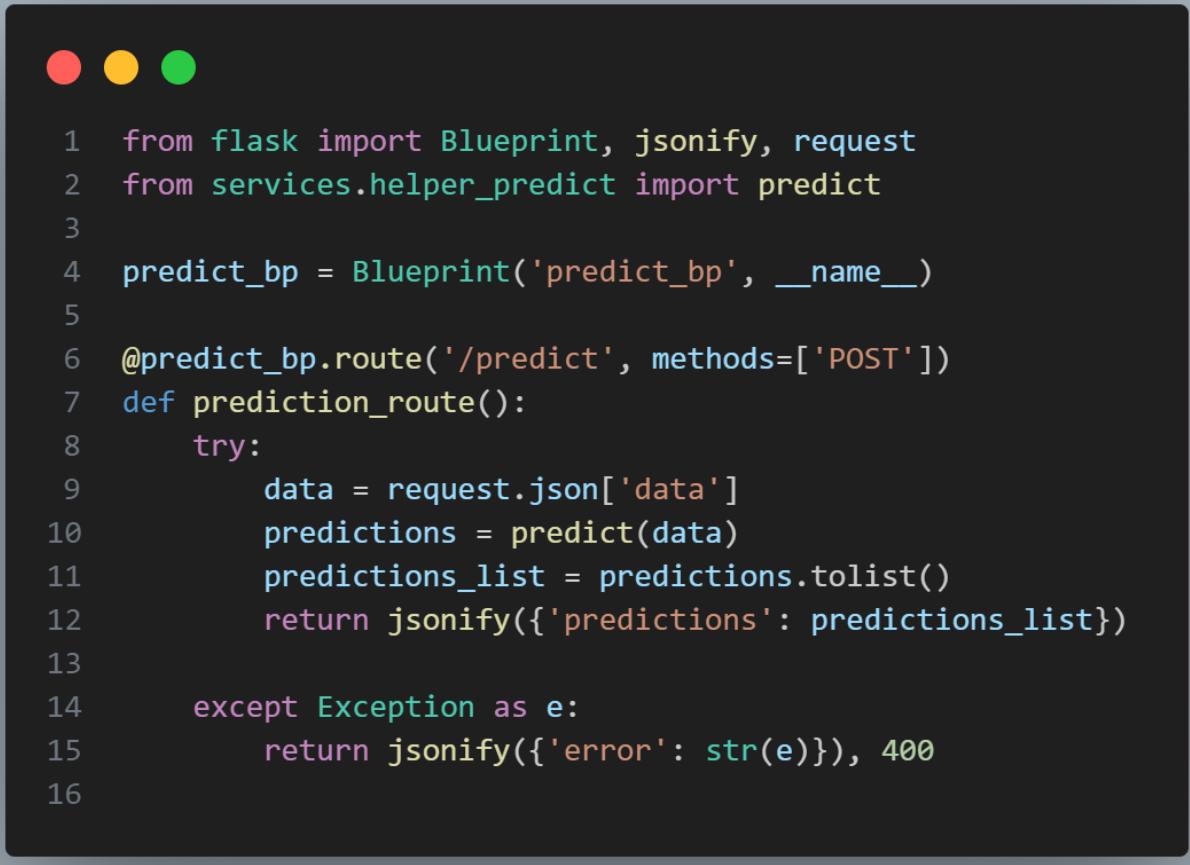
A Figura acima ilustra a configuração do *Blueprint* `vectorize_bp` utilizando a biblioteca Flask, que facilita a organização do código em componentes modulares. O Endpoint `/api/vectorize` aceita requisições POST com textos para vetorização. A função `vectorize_text(text)` é o núcleo desta operação, onde, idealmente, o texto recebido é transformado em um vetor numérico utilizando o modelo Word2Vec.

Os vetores resultantes são cruciais para a rota de classificação de sentimentos, onde são analisados para identificar as emoções contidas nos textos. Esta interconexão entre as rotas de limpeza, vetorização e classificação sublinha a importância da vetorização, funcionando como uma ponte entre o texto bruto e as análises sofisticadas realizadas sobre os dados. Cada vetor,meticulamente organizado em 300 dimensões, é formatado em JSON, garantindo compatibilidade com os processos a jusante e proporcionando uma base sólida para decisões baseadas em dados. A rota é estruturada para ser robusta, gerenciando exceções e falhas com eficácia para fornecer respostas consistentes e informativas para os usuários da API.

10.3. Rota de Classificação

O Endpoint de predição (ou classificação) desempenha uma função crítica na API, realizando análises de sentimentos a partir de textos fornecidos. Ele é essencial para determinar a natureza emocional dos comentários, apoando decisões baseadas em feedbacks dos usuários. Este Endpoint, junto ao resto da API, integra-se com a plataforma "Slack" para proporcionar análises vitais para monitorar e responder a sentimentos expressos sobre a Uber nas redes sociais.

Figura 29 - Rota de Classificação



```
 1 from flask import Blueprint, jsonify, request
 2 from services.helper_predict import predict
 3
 4 predict_bp = Blueprint('predict_bp', __name__)
 5
 6 @predict_bp.route('/predict', methods=['POST'])
 7 def prediction_route():
 8     try:
 9         data = request.json['data']
10         predictions = predict(data)
11         predictions_list = predictions.tolist()
12         return jsonify({'predictions': predictions_list})
13
14     except Exception as e:
15         return jsonify({'error': str(e)}), 400
16
```

Fonte: Material produzido pelos autores (2024)

Esse Endpoint é configurado e aceita requisições do tipo POST. É projetado para receber dados em formato JSON (mais especificamente um vetor de 300 dimensões que representa uma frase). Esse vetor é o resultado do processamento feito pela rota anterior com o modelo de Word2Vec, que prepara os dados para a análise de sentimentos.

O código importa o módulo `helper_predict`, que inclui funções essenciais, principalmente a `predict`. Esta função utiliza um modelo de Regressão Logística, carregado de um arquivo `.pk1` através da biblioteca pickle, para prever sentimentos a partir de vetores de características.

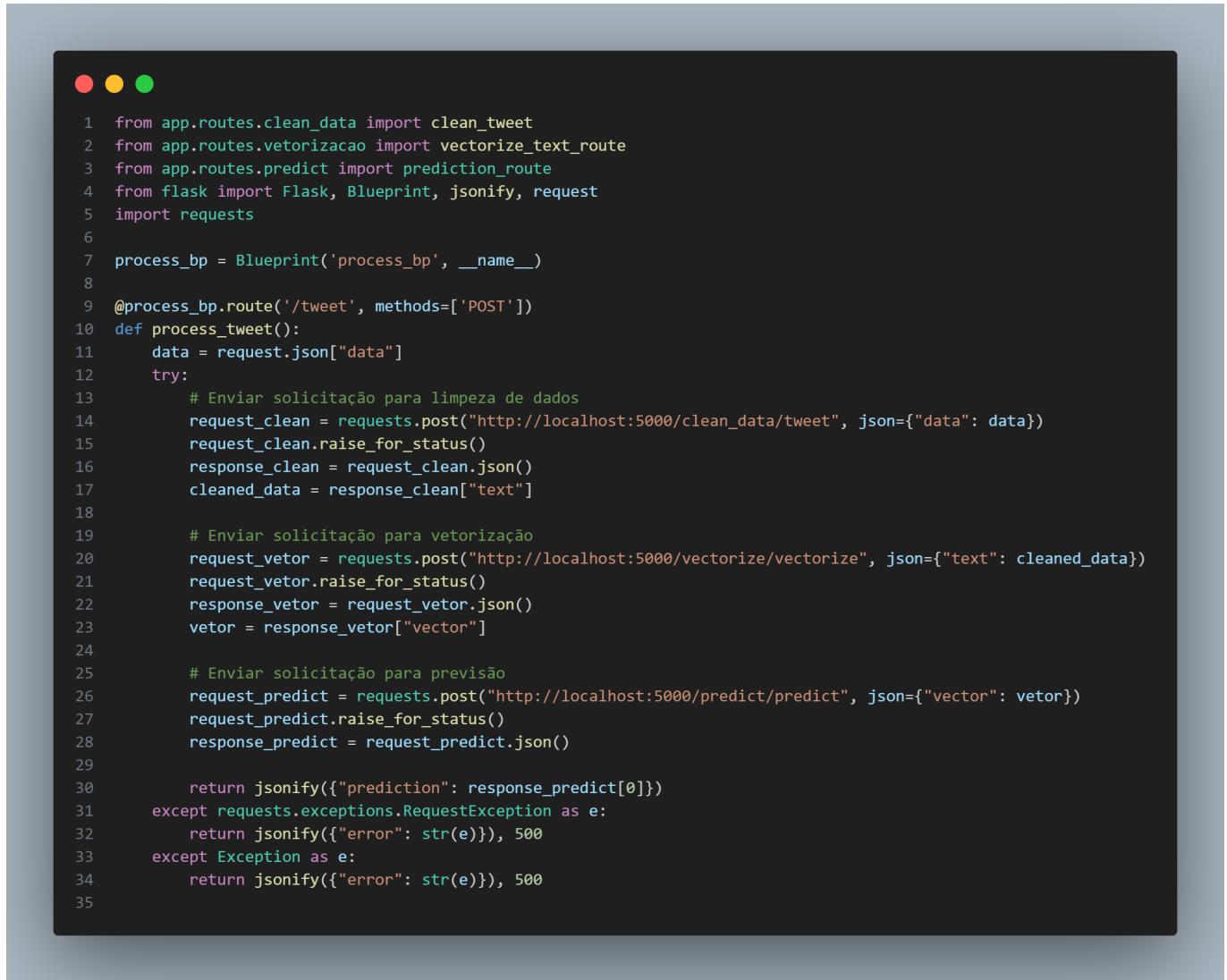
Então é processado o vetor, ajustando os dados para o formato necessário (`np.array(data).reshape(1, -1)`) e utilizando o modelo de Regressão Logística carregado para prever o sentimento, retornando um "array" (0 para negativo, 1 para positivo).

O Endpoint responde com as previsões encapsuladas em JSON. Este formato de resposta é ideal para integração com aplicações cliente, incluindo o Slack, que será usado para proporcionar uma reação eficaz baseada na análise de sentimentos.

10.4. Rota Geral

A rota geral de processamento é vital na API deste projeto, pois ela coordena e executa uma sequência de operações cruciais para o processamento de texto. A finalidade deste Endpoint é integrar diversas funcionalidades — limpeza de dados, vetorização e predição de sentimentos — em uma única chamada de API. Esta rota é essencial para garantir uma análise de sentimentos eficiente e precisa, particularmente na integração com o bot do Slack, onde respostas rápidas e acuradas são cruciais para a gestão de interações com os usuários.

Figura 30 - Rota Geral



```

1  from app.routes.clean_data import clean_tweet
2  from app.routes.vetorizacao import vectorize_text_route
3  from app.routes.predict import prediction_route
4  from flask import Flask, Blueprint, jsonify, request
5  import requests
6
7  process_bp = Blueprint('process_bp', __name__)
8
9  @process_bp.route('/tweet', methods=['POST'])
10 def process_tweet():
11     data = request.json["data"]
12     try:
13         # Enviar solicitação para limpeza de dados
14         request_clean = requests.post("http://localhost:5000/clean_data/tweet", json={"data": data})
15         request_clean.raise_for_status()
16         response_clean = request_clean.json()
17         cleaned_data = response_clean["text"]
18
19         # Enviar solicitação para vetorização
20         request_vetor = requests.post("http://localhost:5000/vectorize/vectorize", json={"text": cleaned_data})
21         request_vetor.raise_for_status()
22         response_vetor = request_vetor.json()
23         vetor = response_vetor["vector"]
24
25         # Enviar solicitação para previsão
26         request_predict = requests.post("http://localhost:5000/predict/predict", json={"vector": vetor})
27         request_predict.raise_for_status()
28         response_predict = request_predict.json()
29
30         return jsonify({"prediction": response_predict[0]})
```

The code is a Python script using the Flask framework. It defines a Blueprint named 'process_bp'. It has a single endpoint '/tweet' that handles POST requests. When a POST request is made to '/tweet', it first calls the 'clean_data' endpoint at 'http://localhost:5000/clean_data/tweet' with the JSON payload containing the 'data' key. It then calls the 'vectorize' endpoint at 'http://localhost:5000/vectorize/vectorize' with the 'text' key set to the cleaned data. Finally, it calls the 'predict' endpoint at 'http://localhost:5000/predict/predict' with the 'vector' key set to the resulting vector. The script then returns the prediction result as JSON.

Fonte: Material produzido pelos autores (2024)

O Endpoint `/tweet` começa com uma requisição POST que recebe um texto bruto. Este texto é primeiramente submetido à rota de limpeza de dados, onde é purificado de ruídos e formatações desnecessárias, preparando-o para a vetorização. Após a limpeza, o texto é enviado para a rota de vetorização, onde é convertido em um vetor de 300 dimensões usando um modelo de Word2Vec pré-treinado, essencial para a análise semântica do texto.

Seguindo a vetorização, o vetor resultante é encaminhado para a rota de predição. Aqui, um modelo de Regressão Logística previamente treinado avalia o vetor e classifica o sentimento do texto como positivo ou negativo. O resultado da predição é então encapsulado em um objeto JSON e enviado de volta como resposta da API. Este processo integrado não só agiliza a análise de sentimentos, mas também fornece uma

base sólida para respostas automatizadas e interativas via Slack, permitindo uma reação contextual e informada aos feedbacks dos usuários em Tweets a respeito da Uber.

10.5 Testes das rotas

10.5.1 Teste 1

Testes da rota /clean_data/tweet

Corpo da requisição incorreto:

Tabela 9 - Teste da rota /clean_data/tweet -- corpo incorreto

| Pré Condição | Teste | Pós Condição |
|--------------------------|--|---|
| Servidor sendo executado | Tentar acessar a rota /clean_data/tweet e realizar a limpeza dos dados com a seguinte entrada {"text":"uber is so good"} | deve ser retornado o json {"message": "data"}, informando que a chave de entrada correta do json é "data" |

Fonte: Material produzido pelos autores (2024)

Figura 31 - Teste Rota de limpeza

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- Request URL:** http://127.0.0.1:5000/clean_data/tweet
- Body Content:**

```

1 {
2   "text": "uber is so good"
3 }
```
- Response Status:** 400 BAD REQUEST
- Response Body:**

```

1 {
2   "message": "data"
3 }
```

Fonte: Material produzido pelos autores (2024)

Na Figura acima há um exemplo de execução desse teste. Como é possível ver, a resposta da API é um json com a informação de qual chave deve ser utilizada para a rota de limpeza de dados.

Corpo da requisição correto:

Tabela 10 - Teste da rota /clean_data/tweet -- corpo correto

| Pré Condição | Teste | Pós Condição |
|-------------------------------|--|--|
| Servidor está sendo executado | Tentar acessar a rota /clean_data/tweet com o seguinte corpo de requisição <code>{"data": "Example text!"}</code> | Deve ser retornado o texto após o pré processamento no seguinte formato <code>{"text": "example text"}</code> |

Fonte: Material produzido pelos autores (2024)

Figura 32 - Teste rota de limpeza dos dados

The screenshot shows the Postman interface. At the top, there's a navigation bar with 'Overview', 'Getting started', 'GET https://gateway.api...', 'POST http://127.0.0.1:500...', and 'No environment'. Below the bar, the URL 'http://127.0.0.1:5000/clean_data/tweet' is entered. A 'Send' button is visible. The main area shows a POST request to 'http://127.0.0.1:5000/clean_data/tweet'. The 'Body' tab is selected, showing the raw JSON input:

```
1 {
2   "data": "uber is so good"
3 }
```

The response section shows a 200 OK status with a response time of 24 ms and a body size of 187 B. The response JSON is:

```
1 {
2   "text": "uber good"
3 }
```

Fonte: Material produzido pelos autores (2024)

Na imagem acima há um exemplo de execução desse teste. Como é possível ver, a resposta da API é um json com o texto inserido anteriormente, porém após passar pelo pré processamento, estando padronizado

para que o modelo possa ser executado.

10.5.2 Teste 2

Testes da rota /vectorize/vectorize

Corpo da requisição incorreto:

Tabela 11 - Teste da rota /vectorize/vectorize -- corpo incorreto

| Pré Condição | Teste | Pós Condição |
|-------------------------------|---|---|
| Servidor está sendo executado | Tentar acessar a rota /vectorize/vectorize com o seguinte corpo de requisição {"data": "Example text!"} | deve ser retornado o json {"message": "text"}, informando que a chave de entrada correta do json é "text" |

Fonte: Material produzido pelos autores (2024)

Figura 33 - Teste rota de limpeza dos dados

The screenshot shows the Postman interface with the following details:

- URL:** http://127.0.0.1:5000/vectorize/vectorize
- Method:** POST
- Body (raw JSON):**

```

1 {
2   "data": "uber is so good"
3 }
```
- Response Headers:**
 - 400 BAD REQUEST
 - 9 ms
 - 196 B
- Response Body:**

```

1 {
2   "message": "'text'"
3 }
```

Fonte: Material produzido pelos autores (2024)

Na imagem acima há um exemplo de execução desse teste. Como é possível ver, a resposta da API é um json com a informação de qual chave deve ser utilizada para a rota de vetorização de um texto.

Corpo da requisição correto:

Tabela 12 - Teste da rota /vectorize/vectorize -- corpo correto

| Pré Condição | Teste | Pós Condição |
|-------------------------------|---|---|
| Servidor está sendo executado | Tentar acessar a rota <code>/vectorize/vectorize</code> com o seguinte corpo de requisição <code>{"text": "example text!"}</code> | Deve ser retornado o vetor que representa a frase enviada na requisição`` |

Fonte: Material produzido pelos autores (2024)

Figura 34 - Teste rota de limpeza dos dados

The screenshot shows the Postman interface. The URL in the header is `http://127.0.0.1:5000/vectorize/vectorize`. The method is set to `POST`. In the `Body` tab, the `raw` option is selected, and the JSON payload is:

```

1 {
2   "text": "uber is so good"
3 }

```

The response at the bottom shows a `200 OK` status with a response time of `9 ms` and a size of `5.3 KB`. The response body is a JSON object with a single key `"vector"` containing a list of numbers representing a vector.

Fonte: Material produzido pelos autores (2024)

Na imagem acima há um exemplo de execução desse teste. Como é possível ver, a resposta da API é um json com uma lista de vetores, no caso, como houve o processamento de apenas uma frase, há apenas um vetor.

10.5.3 Teste 3

Testes da rota `/predict/predict`

Corpo da requisição incorreto:

Tabela 13 - Teste da rota /predict/predict -- corpo incorreto

| Pré Condição | Teste | Pós Condição |
|-------------------------------|--|--|
| Servidor está sendo executado | Tentar acessar a rota <code>/predict/predict</code> com o seguinte corpo de requisição <code>{"text": [[123, 134, 123]]}</code> , sendo a lista de numeros, apenas um exemplo de vetor | deve ser retornado o json <code>{"message": "vector"}</code> , informando que a chave de entrada correta do json é <code>"vector"</code> |

Fonte: Material produzido pelos autores (2024)

Figura 35 - Teste rota de limpeza dos dados

Fonte: Material produzido pelos autores (2024)

Na imagem acima há um exemplo de execução desse teste. Como é possível ver, a resposta da API é um json com a informação de que o para realizar uma predição é necessário que seja enviado em um json com uma chave `vector` uma lista de vetores.

Corpo da requisição correto:

Tabela 14 - Teste da rota /predict/predict -- corpo correto

Fonte: Material produzido pelos autores (2024)

Figura 36 - Teste rota de predição

Fonte: Material produzido pelos autores (2024)

Acima, a imagem documentada. Como é possível ver, a resposta da API é um json com uma lista de predições para as frases, ou vetores, enviados no corpo da requisição à API.

10.5.4 Teste 4

Testes da rota /process/tweet

Corpo da requisição incorreto:

Tabela 15 - Teste da rota /process/tweet -- corpo incorreto

| Pré Condição | Teste | Pós Condição |
|-------------------------------|--|--|
| Servidor está sendo executado | Tentar acessar a rota <code>/process/tweet</code> com o seguinte corpo de requisição <code>{"text": "Example text"}</code> | É retornado um erro http 500, informando que com aquela entrada há um erro interno no servidor |

Fonte: Material produzido pelos autores (2024)

Figura 37 - Teste rota geral

The screenshot shows the Postman interface. At the top, it says "Overview", "Getting started", "GET https://gateway.api", "POST http://127.0.0.1:5000/process/tweet", "No environment". Below that, the "Body" tab is selected, showing a raw JSON payload:

```

1 {
2   "text": "uber is so good"
3 }

```

At the bottom, the response is displayed as "Pretty" HTML:

```

1 <!doctype html>
2 <html lang=en>
3 <title>500 Internal Server Error</title>
4 <h1>Internal Server Error</h1>
5 <p>The server encountered an internal error and was unable to complete your request. Either the server is
   overloaded or
6    there is an error in the application.</p>

```

Fonte: Material produzido pelos autores (2024)

Na imagem acima há um exemplo de execução desse teste. Como é possível ver, a resposta da API é informar que a partir daquela entrada ocorreu um erro.

Corpo da requisição correto:

Tabela 16 - Teste da rota /process/tweet -- corpo correto

| Pré Condição | Teste | Pós Condição |
|--------------|-------|--------------|
|--------------|-------|--------------|

| Pré Condição | Teste | Pós Condição |
|-------------------------------|--|---|
| Servidor está sendo executado | Tentar acessar a rota <code>/process/tweet</code> com o seguinte corpo de requisição <code>{"data": "Example text"}</code> | Deve ser retornado a predição do modelo para o texto enviado. |

Fonte: Material produzido pelos autores (2024)

Figura 38 - Teste rota de predição

The screenshot shows the Postman interface. At the top, there are tabs for Overview, Getting started, and a selected tab for https://gateway.api. Below that, a sub-tab for POST http://127.0.0.1:5000/process/tweet is selected. The main area shows a POST request with the URL http://127.0.0.1:5000/process/tweet. The Body tab is active, showing the raw JSON input:

```

1 {
2   "data": "uber is so good"
3 }

```

Below the request, the response is shown with a status of 200 OK, a duration of 6.26 s, and a size of 185 B. The response body is also JSON:

```

1 {
2   "prediction": 1.0
3 }

```

Fonte: Material produzido pelos autores (2024)

Acima, a imagem documentada. Como é possível ver, a resposta da API é um json com uma lista de predições para as frases, ou vetores, enviados no corpo da requisição à API.

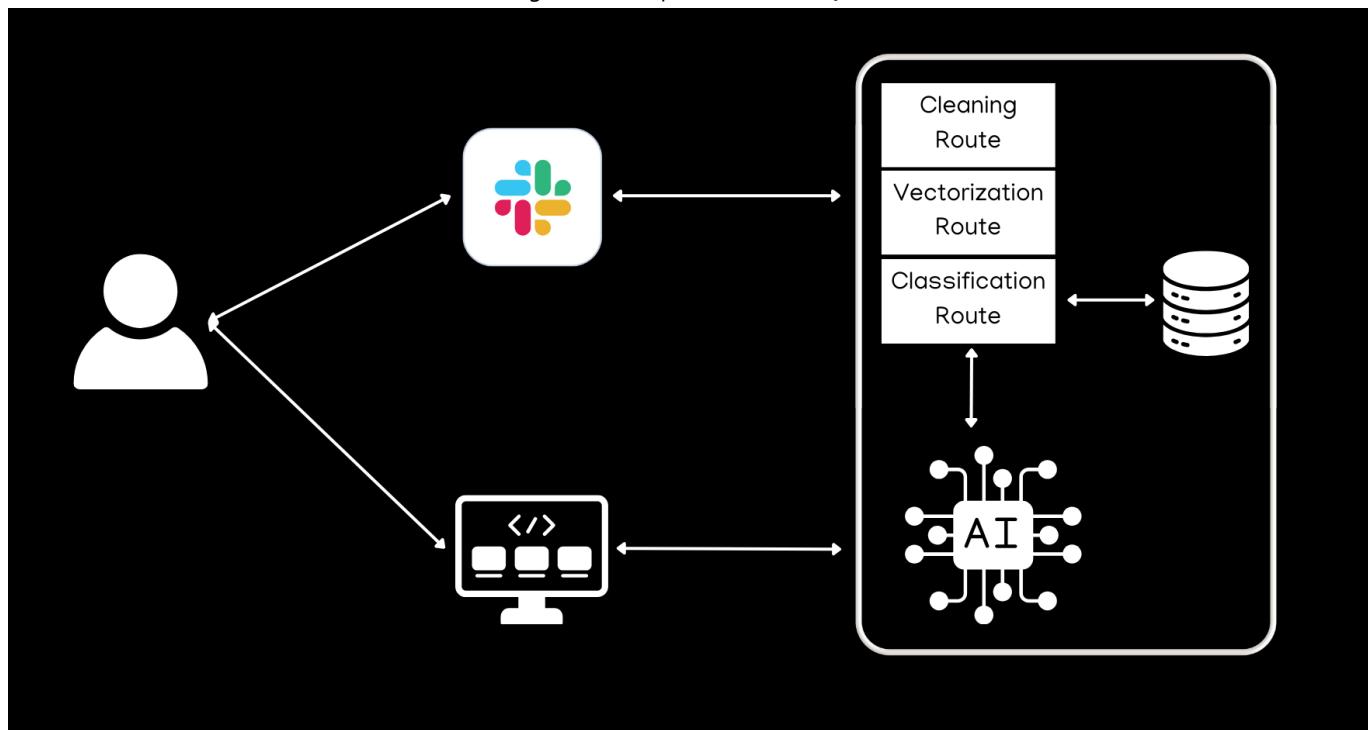
10.6. Conclusão

As rotas apresentadas nesta API são componentes interligados que desempenham papéis fundamentais na análise de sentimentos de textos. Desde a limpeza dos dados até a classificação dos sentimentos expressos, cada etapa é cuidadosamente projetada para garantir precisão e eficiência. A capacidade de integrar essas funcionalidades em uma chamada única de API demonstra a robustez e a versatilidade da solução proposta, especialmente no contexto de interações em tempo real, como as proporcionadas pelo Slack. Essa API não apenas simplifica o processo de análise de sentimentos, mas também abre portas para uma compreensão mais profunda e contextualizada das emoções expressas nos textos, contribuindo significativamente para uma tomada de decisão informada e reativa.

11. Arquitetura Macro

A arquitetura da solução tem como objetivo final representar em alto nível como a solução foi projetada, demonstrando sua estrutura. Dessa forma, ela mostra como o sistema funciona, desde a maneira como o usuário interage com ele até como os elementos existentes neste se comunicam.

Figura 39 - Arquitetura da Solução



Fonte: Material produzido pelos autores (2024)

Como visto na arquitetura acima, o usuário pode interagir com a solução tanto através do Slack quanto do Front-end. Caso o usuário prefira utilizar o Slack, ele enviará suas requisições através dessa plataforma, as quais serão então enviadas para o Backend, que por sua vez, retornará as respostas ao usuário através do Slack.

Dentro do Backend, que está representado no retângulo demarcado na imagem, existem três rotas: "Cleaning Route", "Classification Route" e "Vectorization Route". Essas rotas servem, respectivamente, para a limpeza, vetorização e classificação dos dados inseridos pelos usuários. A primeira rota recebe os dados brutos, executa a limpeza neles e então os redireciona para a rota de vetorização, onde todos os textos recebidos são transformados em vetores. A rota de vetorização envia os dados para a rota de classificação, a qual utiliza um modelo de processamento de linguagem natural para classificar o sentimento transmitido por cada tweet. Posteriormente, a rota de classificação calcula a porcentagem de dados com sentimentos negativos na base, armazena essa informação no banco de dados e registra a data em que tal classificação foi realizada. O resultado deste processo é então retornado para o usuário através do Slack na forma de alerta, além de ser apresentado no front-end.

Além disso, o usuário pode realizar requisições através do front-end, que as envia ao back-end e realiza o processo já descrito. É válido ressaltar que, sempre que o usuário realiza uma requisição, independentemente

de ser pelo front-end ou pelo Slack, a resposta vinda do backend retornará nas duas plataformas, garantindo que ambas permaneçam constantemente atualizadas.

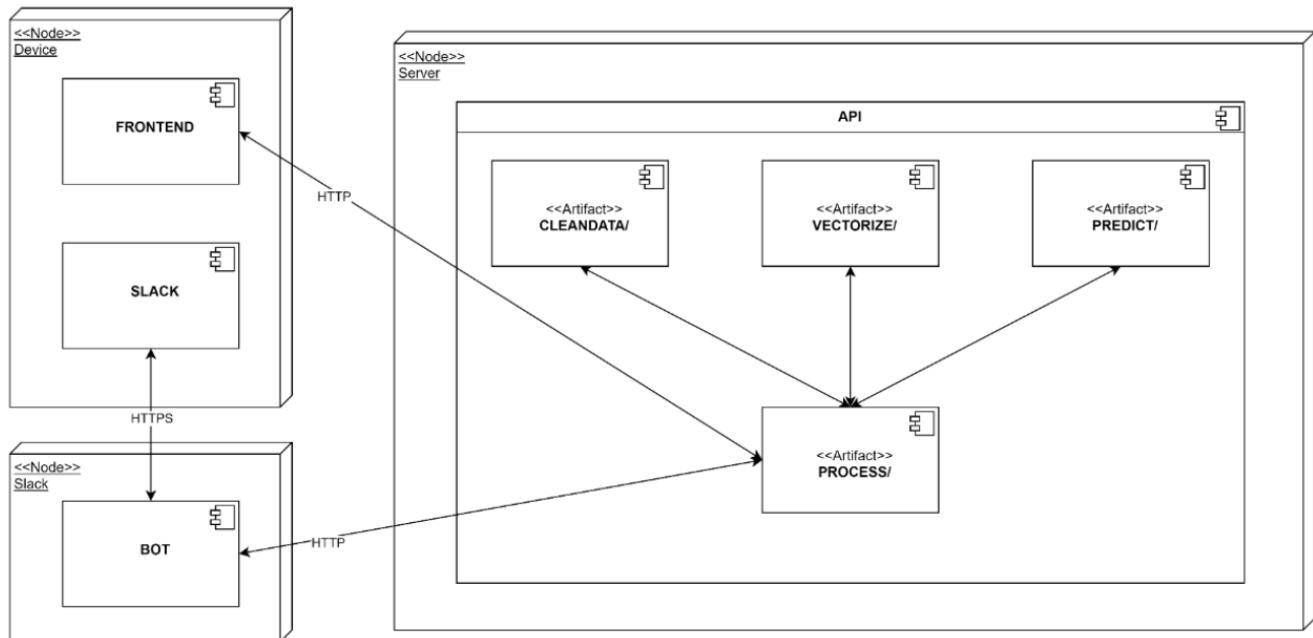
Em suma, a arquitetura da solução permite uma comunicação eficiente entre o usuário e o sistema, seja pelo Slack ou pelo front-end, e assegura que ambas as plataformas estejam sempre sincronizadas com os resultados das análises realizadas.

12. Diagrama de Implantação UML

Um diagrama de implantação UML (Unified Modeling Language) é uma ferramenta visual utilizada para representar a arquitetura física de um sistema. Este tipo de diagrama ilustra a configuração dos componentes de hardware e software de um sistema, mostrando como diferentes elementos se relacionam e interagem entre si em um ambiente de execução. Componentes, nós e suas conexões são representados graficamente para facilitar a compreensão de como o sistema será implementado e operado.

A Figura abaixo mostra o diagrama de implantação UML da *Nexus*.

Figura 40 - Diagrama de Implantação UML



Fonte: Material produzido pelos autores (2024)

Existem três "Nodes" (nós) diferentes: **Device**, **Slack**, e **Server**.

Node: Device

Este nó representa o dispositivo que contém dois artefatos:

- **FRONTEND:** Este artefato representa a interface de usuário da aplicação. Ele está conectado ao servidor via protocolo HTTP, indicando que ele faz solicitações ao servidor.

- **SLACK:** Este artefato representa a integração com a plataforma Slack. Ele se comunica com o artefato BOT via HTTPS, assegurando uma conexão segura.

Node: Slack

Este nó representa a plataforma Slack e contém um artefato:

- **BOT:** Este artefato representa um bot integrado ao Slack. Ele se comunica com o SLACK via HTTPS e com o FRONTEND via HTTP.

Node: Server

Este nó representa o servidor da aplicação, que hospeda vários artefatos sob a camada de API:

- **CLEANDATA/:** Este artefato é responsável pela limpeza dos dados. Ele se comunica com o artefato PROCESS/.
- **VECTORIZER/:** Este artefato é responsável pela vetorização dos dados. Ele também se comunica com o artefato PROCESS/.
- **PREDICT/:** Este artefato realiza as previsões baseadas nos dados processados. Ele se comunica com o artefato PROCESS/.
- **PROCESS/:** Este artefato centraliza o processamento dos dados, recebendo inputs dos artefatos CLEANDATA/, VECTORIZER/ e enviando resultados para o PREDICT/.

Uma observação importante: todos os artefatos do nó Server possuem duas rotas diferentes: /tweet e /csv. Ou seja, os artefatos CLEANDATA/, VECTORIZER/, PREDICT/ e PROCESS/ podem receber as rotas /tweet ou /csv. Caso recebam a rota /tweet, apenas um comentário foi dado como input. Já a rota /csv é a que ocorre quando o input é um data frame com mais de um comentário.

Conexões

- **HTTP:**
 - Utilizado pelo FRONTEND para se comunicar com o servidor.
 - Utilizado pelo BOT para se comunicar com o FRONTEND.
- **HTTPS:**
 - Utilizado pelo SLACK para se comunicar com o BOT, garantindo segurança na transferência de dados.

Referências

[1] Value Proposition Canvas: o que é e como funciona essa metodologia: G4 Educação. Disponível em: <https://g4educacao.com/portal/value-proposition-canvas>. Acesso em: 23 abr. 2024.

[2] User Story - Saiba o que é | Glossário de produto da PM3. Disponível em: <https://www.cursospm3.com.br/glossario/user-story/>. Acesso em: 25 abr. 2024.

[3] Análise descritiva: o que é e como interpretar os dados. Disponível em:
<https://www.fiveacts.com.br/analise-descritiva>. Acesso em: 7 mai. 2024.

[4] What is Bag of Words? | IBM. Disponível em: <https://www.ibm.com/topics/bag-of-words>. Acesso em: 7 mai. 2024.

[5] Word Embedding using Word2Vec | Geeks for Geeks Disponível em:
<https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/>. Acesso em 16 de mai. de 2024