Um Aplicativo para Ranqueamento de vídeos do YouTube baseado em Mineração de Dados das emoções encontradas nos comentários

Brenno Resende Martins

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Monte Carmelo - MG
2025

Brenno Resende Martins

Um Aplicativo para Ranqueamento de vídeos do YouTube baseado em Mineração de Dados das emoções encontradas nos comentários

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Professor Carlos Cesar Mansur Tuma

Monte Carmelo - MG
2025

Resumo

O objetivo deste trabalho foi desenvolver um aplicativo que ranqueia os vídeos em alta do YouTube com base na mineração de dados das emoções encontradas nos comentários dos usuários para filtrar os vídeos mais bem qualificados pelos espectadores. Para o desenvolvimento do trabalho, foram usadas APIs que retornam informações públicas dos vídeos do YouTube - como seus títulos e lista de comentários; e um modelo de linguagem chamado EmoRoBERTa para classificar as emoções encontradas no comentários, e com isso finalmente classificar os vídeos de acordo com as emoções positivas e negativas que os usuários expressaram ao assistir um vídeo. Para testar a eficiência destas classificações, foi desenvolvida uma interface onde o usuário pode navegar por uma lista de emoções e ver quais vídeos em alta no YouTube possuem mais comentário refletindo a emoção escolhida. Com testes feitos com estudantes da Universidade Federal de Uberlândia, concluiu-se que esta aplicação foi bem recebida, tendo em maioria *feedbacks* positivos.

Palavras-chave: Mineração de Dados, YouTube, EmoRoBERTa, LLM.

Lista de ilustrações

Lista de tabelas

Lista de siglas

API Interface de Programação de Aplicações - *Aplication Programming Interface*

BoW Sacola de Palavras - *Bag of Words*

BERT Representações de codificadores bidirecionais de transformadores - *Bidirecional Encoder Representations from Transformers*

CRISP-DM Processo Padrão Inter-Indústrias para Mineração de Dados - *Cross-Industry Standard Process for Data Mining*

EMTk *Emotion Mining Toolkit*

KDD Descoberta de Conhecimento em Bancos de Dados - *Knowledge-Discovery in Databases*

PLN Processamento de Linguagem Natural

SEMMA Amostra, Explorar, Modificar, Modelar e Avaliar - Sample, Explore, Modify, Model, Assess

SVM Máquina de vetores de suporte - Support Vector Machine

TF-IDF Termo Frequência-Inverso Frequêcia dos Documentos - *Term Frequency-Inverse Document Frequency*

Sumário

Capítulo 1

Introdução

Redes Sociais nasceram com a proposta de oferecer socialização e entretenimento a longa distância por meio da internet. De acordo com (ZENHA, 2018), "entende-se como Rede Social online, o ambiente digital organizado por meio de uma interface virtual própria que se organiza agregando perfis humanos que possuam afinidades, pensamentos e maneiras de expressão semelhantes e interesse sobre um tema comum".

Estas redes podem funcionar de diversas maneiras. O Facebook, que atualmente é a rede com mais contas cadastradas, possui cerca de 3,07 bilhões de usuários (SAGE, 2025), permite que as pessoas compartilhem textos, imagens, vídeos e interações entre si. Outras redes como Instagram e X (anteriormente Twitter) seguem padrões parecidos, ambas são focadas em compartilhar fotos e vódeos curtos, porém também existem algumas com propostas diferentes, como o YouTube. A proposta principal do YouTube é compartilhar vídeos longos enquanto mantém os aspectos sociais importantes, como comentários em vídeos e postagens em comunidade. A Figura 1 apresenta um gráfico que ilustra o índice de usuários que usam os aplicativos de redes sociais em seus smartphones. Na imagem é possível notar que o YouTube aparece no topo, presente em todos os dispositivos, uma vez que este aplicativo vem instalado de fábrica na maioria dos *smartphones*.

Com o passar dos anos, as redes sociais sofreram algumas mudanças que transformaram o seu foco em veículos informativos, onde os usuários não vão apenas para se divertir, mas também para se atualizar sobre o que está acontecendo no mundo. O uso de redes sociais cresce a cada dia, especialmente após a pandemia da COVID-19, que trouxe uma necessidade maior ainda de conexão à internet. O uso da internet é tão grande que, no mês de fevereiro de 2025, cerca de 63,9% de toda a população mundial acessou alguma rede social (STATISTA, 2025).

1.1 Motivação

Embora tenham seus benefícios, as redes sociais têm a capacidade de influenciar o comportamento de seus usuários. A plataforma de vídeos YouTube, em específico, oferece

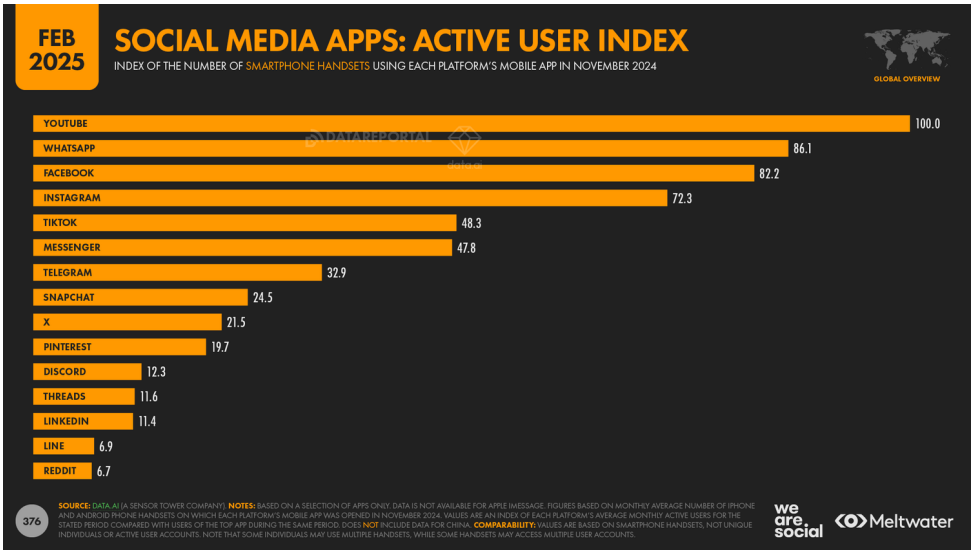


Figura 1 – Aplicativos de redes sociais mais presentes nos smartphones ao redor do mundo. Fonte: DataReportal (2025)

conteúdo variado para todos os públicos, sendo um dos aplicativos mais populares no Brasil, com cerca de 144 milhões de usuários em janeiro de 2024 (STATISTA, 2024).

Um dos problemas mais comuns desta rede é o seu algoritmo de recomendação, que muitas vezes recomenda conteúdos duvidosos, como por exemplo, os chamados *Clickbaits* - Iscas de Cliques, em português, que consistem em títulos ou capas que vendem um assunto chamativo mas que não está presente no vídeo de fato. Em seu estudo (ZANNETTOU et al., 2018) desenvolveu um modelo que analisou títulos, capas e comentários de mais de 200 mil vídeos, para identificar aqueles que se encaixam nos *clickbaits*. Chegou-se a conclusão de que o algoritmo do YouTube até então não tinha nenhum tratamento para evitar que vídeos enganosos fossem recomendados, e que 41% dos vídeos incluídos no trabalho foram identificados como *clickbaits*.

Além disso, procurando trabalhos relacionados ao algoritmo de recomendações do YouTube, notou-se falta de trabalhos que foquem em melhorar este algoritmo. Assim, a motivação maior desta monografia é encontrar alguma forma de melhorar as recomendações do YouTube, usando como base os comentários dos próprios usuários.

1.2 Problema

Algoritmo do YouTube baseia suas recomendações, de forma geral, no número de visualizações e no texto de *clickbait* do título do vídeo, não levando em consideração a satisfação dos usuários apresentada nos comentários.

1.3 Objetivo Geral

Criar aplicativo que analise os comentários de vídeos do YouTube e crie um ranqueamento para mostrar quais os vídeos em que os usuários expressam maior satisfação em relação a uma emoção específica.

1.4 Objetivos Específicos

1. Estudar e selecionar uma API para extrair os links de vídeos do YouTube.
2. Estudar e selecionar uma API para extração de comentários de vídeos do YouTube.
3. Estudar e selecionar uma API de pré-processamento de linguagem natural que consiga também encontrar e classificar emoções em texto.
4. Extrair os links dos vídeos selecionados com a API do objetivo um, e deles extrair os comentários com a API do objetivo dois.
5. Classificar os comentários obtidos em termos de emoções para cada vídeo selecionado.
6. Agrupar as emoções em subgrupos expressivos.
7. Ranquear os vídeos para cada subgrupo.
8. Criar uma interface para que o usuário possa selecionar uma emoção e em seguida ver quais os vídeos foram mais bem classificados.
9. Criar um questionário para colher impressões e sugestões dos usuários.

1.5 Contribuições

Por meio da construção de um aplicativo, melhorar a recomendação entre os vídeos em alta do YouTube com relação às emoções percebidas nos comentários dos usuários.

1.6 Organização da Monografia

O primeiro capítulo introduz a motivação para esta monografia, discutindo sobre o uso crescente de redes sociais, o problema das recomendações ruins no YouTube e apresentando os objetivos do trabalho.

No segundo capítulo é mostrada toda a fundamentação teórica necessária para o desenvolvimento do trabalho. Os conceitos discutidos englobam assuntos da área da programação voltada para tratamento de texto, como mineração de dados e processamento

de linguagem natural; além de englobar também alguns assuntos abstratos da área de psicologia, como sentimentos e opiniões, que são de grande importância para os objetivos descritos no capítulo um.

O capítulo três descreve detalhadamente todas as etapas realizadas para a execução do trabalho proposto no capítulo um, desde a obtenção dos comentários em vídeos do YouTube, seu processamento até as avaliações dos usuários que testaram a aplicação pronta e as conclusões que foram obtidas com o feedback recebido.

No capítulo quatro são apresentadas as conclusões e sugestões para trabalhos futuros. Por fim, logo em seguida, os apêndices mostram os códigos usados para desenvolvimento deste trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados conceitos relacionados às áreas mineração de dados, organização de informação, processamento de linguagem natural e análise de sentimentos, que foram usados no desenvolvimento deste trabalho.

2.1 Mineração de Dados e KDD

Mineração de dados diz respeito a processar conjuntos de dados em busca de padrões não perceptíveis explicitamente. Esta área de pesquisa está ligada diretamente ao processo conhecido como *Knowledge-Discovery in Databases*, ou Descoberta de Conhecimento em Bancos de Dados (KDD) em português. De acordo com Frawley, Piatetsky-Shapiro e Matheus (1992) o processo de KDD é a extração de informações implícitas e desconhecidas em bases de dados. Apesar de similares, a mineração é uma das etapas do processo de KDD.

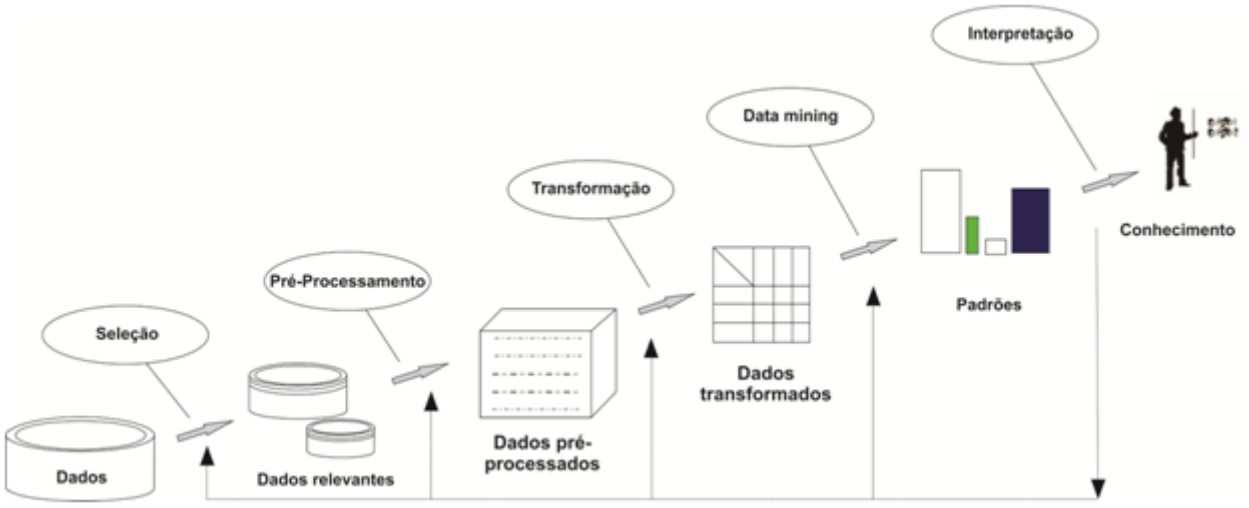


Figura 2 – Etapas do KDD. Fonte: Fayyad et al. (1996), retirado do site DevMedia (2024)

Estudando metodologias de mineração de dados como o Processo Padrão Inter-Indústrias

para Mineração de Dados (CRISP-DM) e o Amostra, Explorar, Modificar, Modelar e Avaliar (SEMMA) Braga (2005) definiu algumas etapas presentes em ambos os modelos:

2.1.1 Etapa de definição do problema

Como o nome sugere, Braga (2005) diz que esta etapa consiste em identificar o problema do usuário e o que se quer encontrar nos dados a serem minerados. Este passo envolve entrevistas com o usuário, estudar os dados e identificar ferramentas à disposição.

2.1.2 Etapa de aquisição e avaliação dos dados

Etapa mais importante e que trata de encontrar e fazer o pré-processamento dos dados. O pré-processamento consiste em transformar dados não padronizados em dados formatados para serem estudados Braga (2005). Algumas técnicas de formatação podem ser limpeza e transformação.

- Limpeza - Técnicas de limpeza consistem em tratar dados incompletos ou com ruídos.
  - Registros incompletos são aqueles em que um ou mais atributos vem faltando, e estes podem ser, por exemplo, substituídos pela média ou mediana do atributo; ou então simplesmente removidos do conjunto.
  - Registros com ruídos são aqueles em que o valor de um determinado atributo não faz sentido no contexto desejado, neste caso são aplicadas, por exemplo, funções de regressão e agrupamento, para tentar suavizá-los.
- Transformação - Transforma o formato dos dados originais para um novo formato mais fácil de interpretar na mineração. Alguns exemplos de técnicas de transformação são normalização, seleção e discretização.
  - Normalização consiste em escalar os valores para um novo intervalo. Por exemplo, colocar todos os valores entre 0 e 1.
  - Seleção pega dois ou mais atributos e os transforma em um novo atributo que é a junção de ambos.
  - Discretização é a transformação de variáveis contínuas em variáveis discretas.

  Esta etapa envolve também validação dos dados obtidos, seleção de ferramentas adequadas para processamento e criação de amostras para testes.
- Prototipagem e desenvolvimento do modelo - Trata da criação do modelo de dados, ou seja, a escolha das informações que vão passar por algoritmos de mineração em busca de padrões.

- ❑ Avaliação, Implantação e *Feedback* - Avaliação da eficiência do modelo ao ser colocado em prática, obtendo retorno para fazer as alterações necessárias.

2.1.3 Redes Neurais Artificiais

De acordo com Kukreja et al. (2016), uma Rede Neural Artificial é uma tentativa de imitar o comportamento do cérebro humano com relação à tomada de decisões sobre dados subjetivos. Ainda de acordo com o autor, as redes neurais consistem em neurônios artificiais que copiam o comportamento de neurônios reais. Vários neurônios artificiais são colocados juntos para treinamento em tomada de decisão, formando assim as Redes Neurais.

As Redes Neurais estão muito relacionadas com inteligência artificial e processos de classificação - como os que serão usados neste trabalho para identificar emoções em texto. A Figura 3 mostra de forma simplificada o funcionamento básico de uma rede neural.

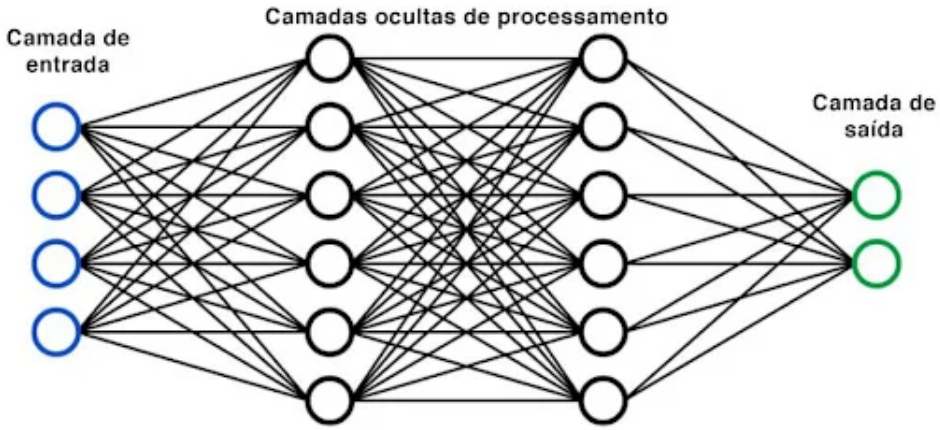


Figura 3 – Exemplificação simplificada do funcionamento das Redes Neurais Artificiais, retirado do site Opencadd (2022)

2.2 PLN - Processamento de Linguagem Natural

O campo de pesquisa do Processamento de Linguagem Natural (PLN) estuda textos publicados na internet em busca de padrões e significados. De acordo com Singh e Weber (2021), a entrada para sistemas de PLN não precisa ser apenas texto, mas também fala e até mesmo gestos. Ainda de acordo com o autor, ´´a saída pode não ser linguística, mas a maioria dos sistemas precisa dar algum tipo de *feedback* ao usuário".

Métodos de PLN são baseados em receber um determinado texto, extrair dele as palavras encontradas, e a partir daí procurar padrões relevantes para o contexto. Estes métodos estão ligados diretamente aos conceitos de Mineração de Dados, mencionados anteriormente, e que no contexto da PLN são chamados de Mineração de Texto.

A Figura 4 ilustra os possíveis usos variados do PLN. Pode ser usado para tradução e auto-correção de textos, sumarização e criação de relatórios, desenvolvimento de *Chat Bots* e modelos de linguagem, além de também trabalhar com análise semântica e de sentimentos em texto. Tal análise será usada em grande parte desde trabalho.

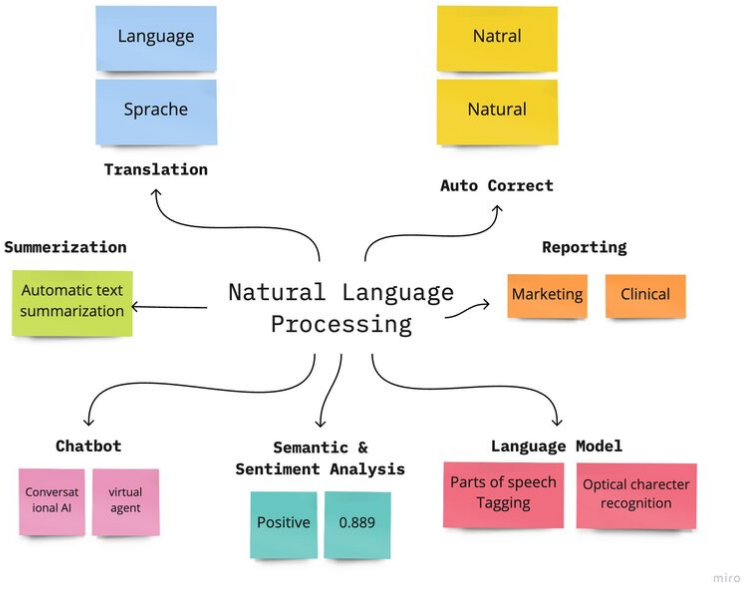


Figura 4 – Exemplo de aplicações para o Processamento de Linguagem Natural. Retirado do trabalho de Hasan (2022).

2.2.1 Pré-Processamento de Texto

Trata-se de todo o conjunto de técnicas, ferramentas e algoritmos usados para separar as palavras importantes no texto. Na análise de PLN, algumas palavras tem mais valor semântico do que outras, e o pré-processamento é a tarefa de limpar o texto, buscando por estas palavras.

2.2.1.1 Corpus

"O corpus é o conjunto dos documentos tidos em conta para serem submetidos aos procedimentos analíticos" Bardin, Reto e Pinheiro (1977). No contexto deste trabalho, o corpus pode ser entendido como todos os textos de entrada para o pré-processamento.

2.2.1.2 Tokenização

De acordo com Singh e Weber (2021), tokenizar é dividir uma sequência de caracteres em tokens individuais e separados por vírgula, dividindo palavras, números e outros sinais gráficos. Por exemplo, a frase "Diga-me o seu nome, por favor." pode ser tokenizada, e gerar o seguinte:

"Diga", "-", "me", "o", "seu", "nome", ",", "por", "favor"

2.2.1.3 *Stopwords*

Os autores Baeza-Yates e Ribeiro-Neto (2013) descrevem as *stopwords* como palavras sem significado na PLN, e que podem ser ignoradas. Apesar de variar com o contexto do trabalho, na língua portuguesa, estas palavras são principalmente os artigos, pronomes, numerais, advérbios e conectores.

Pegando por exemplo a pesquisa na internet "Lojas de roupa e sapatos no Brasil", as palavras "de", "e", "no" podem ser ignoradas e a pesquisa ainda retornará os mesmos resultados.

2.2.1.4 Termos

Termos são as palavras de relevância e alta frequência em um texto e que são usadas no processamento e na classificação. Na língua portuguesa, estas palavras são principalmente os substantivos e os verbos. Retornando ao exemplo anterior da pesquisa na internet "Lojas de roupa e sapatos no Brasil", os termos são "Lojas", "roupa", "sapatos", "Brasil".

2.2.1.5 Estemização e Lematização

Estemização é a técnica de reduzir palavras à sua raiz, removendo flexões e derivações, reunindo palavras de significado semântico parecido. De acordo com Lovins (1968), esta técnica tem como objetivo assegurar que nenhuma variação de palavra foi deixada para trás no pré-processamento.

Um exemplo de estemização seria um dicionário com as palavras "amigo, amigas, amigão". Todas estas palavras têm significados semelhantes, e o resultado da estemização destas palavras resultaria em "amig", que é a raiz que se repete em todas elas.

Porém, esta técnica tem alguns problemas. Por exemplo, a redução à raiz das palavras pode resultar em termos que não possuem significado. No exemplo acima, "amig" não é uma palavra real, e isto poderia causar problemas em alguns algoritmos. Por isso, uma outra alternativa é a técnica de lematização.

Singh e Weber (2021) diz que a lematização é uma maneira de remover terminações flexionais chegando ao lemma - a forma base de uma palavra. É uma técnica mais complexa e mais trabalhosa de se aplicar, pois envolve vocabulário e análise morfológica.

Por exemplo, as palavras "estudioso, estudante, estudado", após a lematização, seriam reduzidas a "estudar", a palavra que deu origem a todas as anteriores, enquanto na estemização, o resultado seria "estud", a parte que se repete.

2.2.1.6 Vocabulário

Vocabulário é a lista final de todos os termos que passaram pelo pré-processamento, já padronizados e com as palavras irrelevantes eliminadas.

2.2.2 *Bag of Words*

Bag of Words (BoW), ou Sacola de Palavras em português, é um método de representação atributo-valor para organização de texto que consiste em criar um vetor com os termos de um texto seguidos de suas ocorrências. Um exemplo de BoW:

[“Água”:5, “azul”:6, “chuva”:2, “céu”:7]

Apesar de simples, a representação da sacola de palavras possui desempenho melhor do que modelos mais elaborados de representação de texto (Apté, Damerau e Weiss (1994); Dumais et al. (1998)).

A Figura 5 apresenta um exemplo simplificado do funcionamento do algoritmo sacola de palavras.

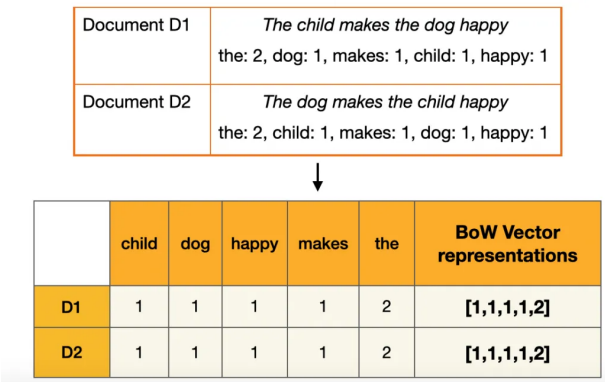


Figura 5 – Exemplo do algoritmo Bag of Words. Retirado do site AML.com (2024).

2.2.3 *Term Frequency-Inverse Document Frequency*

O índice que relaciona a frequência do termo multiplicado pela frequência inversa do termo nos documentos (TF-IDF) é um método de classificação que busca comparar a semelhança entre textos com base em uma dada pesquisa. A classificação é baseada na ocorrência de termos em um texto individual comparando com a ocorrência dos mesmos em um conjunto maior de documentos. É uma maneira eficiente de encontrar quais documentos têm maior correspondência com os termos de busca.

É calculado primeiro obtendo do TF (*Term Frequency*), a frequência do termo no documento, dada pela fórmula:

Tf(t,d) = frequência do termo t no documento d/ total de palavras no documento d

Depois, é calculado o IDF(*Inverse Document Frequency*), a frequência inversa do documento:

IDF(t,D)=log([total de documentos no corpus D/nº de documentos com o termo t])

O cálculo final TF-IDF é feito pela multiplicação de TF por IDF.

2.3 Análise de Sentimentos

Os autores Medhat, Hassan e Korashy (2014) descrevem a Análise de Sentimento como o ramo da computação que estuda as opiniões e atitudes voltadas para uma entidade. O objetivo desta análise é conseguir identificar emoções em textos publicados *online* e classificá-las dentro do contexto em que estão inseridas, e para concretizar este objetivo são usadas as ferramentas de Mineração de Dados e Processamento de Linguagem. Por isso, este campo de estudo também pode ser referido como Mineração de Opinião.

2.3.1 Emoção

O autor Damasio (1994) diz que uma emoção é uma reação biológica passageira. Já o psicólogo Ekman (1999), diz que as emoções são universais para todos os indivíduos.

De forma resumida, uma emoção é uma resposta rápida para um acontecimento do cotidiano - e ela afeta no desenvolvimento dos sentimentos.

2.3.2 Sentimento

De acordo com os trabalhos de Ekman (1999) e Damasio (1994), respectivamente, os sentimentos são pessoais para cada indivíduo e são interpretações de longo prazo que o cérebro faz sobre as emoções vividas no dia a dia.

Em resumo, os sentimentos são mais profundos e duradouros, enquanto as emoções são momentâneas, mas são elas que moldam os sentimentos.

2.3.3 Opinião

De acordo com Silva (2010), opinião é a maneira de ver, sentir e se comportar em relação a uma certa característica ou um determinado objeto; que pode ser, por exemplo, uma pessoa ou um produto. Opiniões podem ser positivas, negativas ou neutras.

Opiniões estão ligadas aos sentimentos e as emoções de um indivíduo diante de determinada situação.

2.3.4 Etapas de busca

Os autores Medhat, Hassan e Korashy (2014) mencionam algumas etapas para identificar aspectos a serem procurados para encontrar sentimentos:

- ❏ Termos e frequências - Após identificar quais aspectos se quer procurar no texto, identificar quais são os termos presentes e quais suas frequências. Podem ser usados os algoritmos de BoW ou TF-IDF, mencionados anteriormente, por exemplo.
- ❏ Partes de fala - Encontrar adjetivos, que são fortes indicadores de opinião.
- ❏ Palavras ou frases de opinião - São palavras ou expressões que caracterizam opiniões, como "bom", "ruim", "caro", "barato", mas nem sempre são explícitas e com sentido claro. Por exemplo, na frase: "Este relógio me custou os olhos da cara", a expressão "os olhos da cara" indica que algo foi muito caro.
- ❏ Negações - São palavras ou expressões com sentido negativo e que mudam a orientação da opinião. Por exemplo, "não é bom" é o equivalente a "ruim".

2.4 Ferramentas e Interfaces

Esta seção apresenta ferramentas e interfaces usadas ao decorrer do desenvolvimento deste trabalho.

2.4.1 Hugging Face

Como pode ser visto na Figura 6, Hugging Face é uma plataforma focada em inteligência artificial que funciona como repositório para modelos de linguagem que trabalham com processamento de linguagem natural. Foi responsável pelo desenvolvimento da biblioteca Transformers para trabalhar com inteligência artificial em Python.

2.4.2 Transformers

A biblioteca Transformers Wolf et al. (2019) é um pacote de código aberto que disponibiliza modelos de aprendizado profundo pré-treinados para tarefas como processamento de linguagem natural e visão computacional, por exemplo.

Esta é a biblioteca principal para treinar modelos de inteligência artificial em Python. Para utilizá-la primeiro cria-se um *pipeline* que recebe uma tarefa a ser feita, como por exemplo, classificação de sentimentos. O pipeline realiza a tarefa desejada e retorna o resultado. O código da Figura 7 demonstra um exemplo de seu funcionamento criando o *pipeline* com a tarefa de análise de sentimentos usando o modelo EmoRoBERTa.

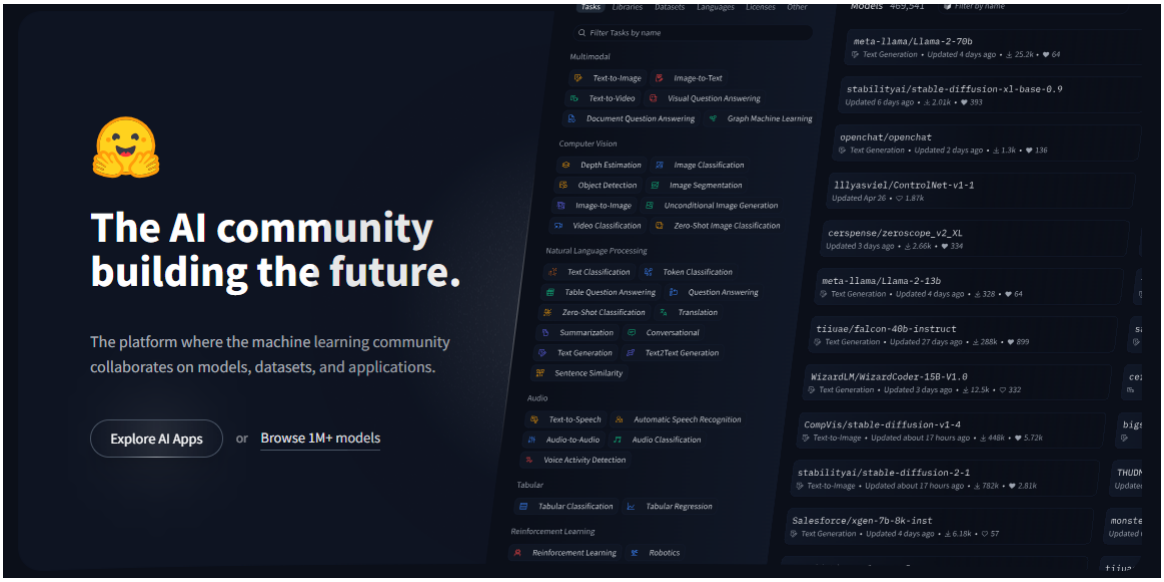


Figura 6 – Repositório Hugging Face. Acesso disponível em: https://huggingface.co/

```python
from transformers import RobertaTokenizerFast, TFRobertaForSequenceClassification, pipeline

tokenizer = RobertaTokenizerFast.from_pretrained("EmoRoBERTa")
model = TFRobertaForSequenceClassification.from_pretrained("EmoRoBERTa")

emotion = pipeline('sentiment-analysis',
                   model='EmoRoBERTa')

emotion_labels = emotion("Thanks for using it.")
print(emotion_labels)
```

Figura 7 – Código usado para testar a biblioteca Transfomers. Cria-se um *pipeline* e atribui a ele a tarefa de análise de sentimentos (*sentiment-analysis)*. Depois, é passado um texto para ser classificado pela análise de sentimentos.

2.4.3 EmoRoBERTa

EmoRoBERTa Ghoshal (2021) é um modelo de linguagem baseado no modelo BERTDevlin et al. (2018) que reconhece e classifica até 28 emoções diferentes em textos.

Este modelo está disponível no repositório HuggingFace para ser clonado localmente. Este modelo é programado na linguagem Python e recebe como parâmetro um texto. A cada texto é atribuída uma emoção e uma nota entre 0 e 1 que mede a similaridade do texto com a emoção atribuída. Até o momento, este modelo classifica apenas textos em inglês.

O código mostrado na Figura 7 da subseção acima trás um exemplo simples do uso deste modelo. O código passa a tarefa de análise de sentimentos para o modelo e ele retorna uma vetor com uma classificação e uma pontuação.

Para a frase usada no exemplo, *"Thanks for using it."* - em português *"Obrigado por usar."*, a resposta será: *'label': 'gratitude', 'score': 0.9964383840560913*, onde *label* é a classificação e *score* é a pontuação. O modelo classificou a frase como tendo o sentimento de gratidão e uma nota bem próxima de 1, o que indica grande identificação com o sentimento.

2.4.4 DeepTranslator

DeepTranslator Baccouri (2020) é uma biblioteca de tradução em Python que usa serviços de tradução comuns na internet, como o Google Tradutor. É uma biblioteca gratuita, mas que impõe limites de solicitação diários.

Esta biblioteca recebe como parâmetro o texto a ser traduzido, o serviço de tradução e o idioma para qual se deseja traduzir, e retorna o texto traduzido. O exemplo da Figura 8 mostra a estrutura do funcionamento da biblioteca DeepTranslator.

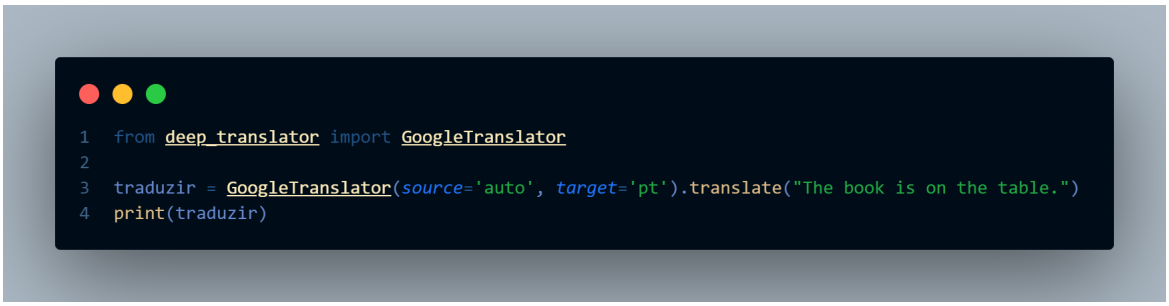


Figura 8 – Código usado para testar a biblioteca Deep Translator.

No exemplo da Figura 8, usa-se a função de tradução do Google Tradutor. O parâmetro *source* define qual o idioma de origem do texto. Neste exemplo, ele está definido como *auto*, para que o idioma de origem seja detectado automaticamente. O outro parâmetro *target* define para qual idioma o texto vai ser traduzido. Neste caso, a função *GoogleTranslate* traduz a frase *"The book is on the table"* para *"O livro está sobre a mesa"*.

2.4.5 API

Application Program Interface (API) é, de acordo com Biehl (2015), um jeito simples de integrar e conectar com um *software* existente. As APIs são usadas para construir *softwares* distribuídos em que os componentes são fracamente acoplados.

Estas interfaces fazem a comunicação entre diferentes aplicações permitindo compartilhamento de dados entre elas. As APIs funcionam como garçons de um restaurante. As aplicações clientes solicitam dados e a interface fica responsável por fazer e entregar os pedidos. Esta analogia está ilustrada na Figura 9.

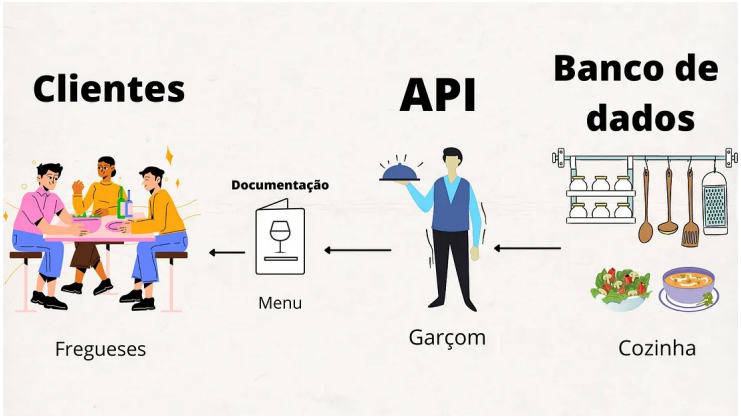


Figura 9 – Analogia das APIs com os garçons.

2.4.5.1 YouTube Trends API

Obtida no site SearchAPI (documentação disponível em *https://www.searchapi.io/docs/youtube-trends* - Esta API retorna os identificadores únicos de cada vídeo da página "Em Alta" do YouTube, que são usados para formar os links dos vídeos.

2.4.5.2 YouTube API

API desenvolvida pelo próprio YouTube (documentação disponível em *https://developers.google.co br* - Recebe como parâmetro os identificadores únicos de cada vídeo, e retorna todas as informações públicas do vídeo, incluindo a lista de comentários de cada vídeo.

2.5 Trabalhos Relacionados

A definição de Emoção é bastante complexa e abstrata. O trabalho de Plutchik (2001) é bastante conhecido no ramo da psicologia por apresentar uma classificação para diferentes emoções usando o modelo mostrado na Figura 10. Este modelo coloca as emoções mais intensas no centro, e a partir delas as emoções vão se ramificando em emoções "mais fracas", por assim dizer, e que possuem intensidade mais baixa.

O estudo de Silva (2010) apresenta uma aplicação de análise de sentimentos baseada em polarização de palavras, seguindo trabalhos propostos anteriormente por Liu et al. (2007) e Ding, Liu e Yu (2008) que classificam os sentimentos em uma sentença encontrando cláusulas negativas e adversativas. O algoritmo de classificação de sentimentos deste trabalho identifica palavras de opinião em uma frase atribuindo a elas uma polaridade positiva, neutra ou negativa, e no final entrega uma pontuação da frase que indica se esta é positiva ou negativa. Exemplo de classificação retirado do próprio trabalho de Silva (2010):

"O Spettus é ótimo e a comida não é cara."

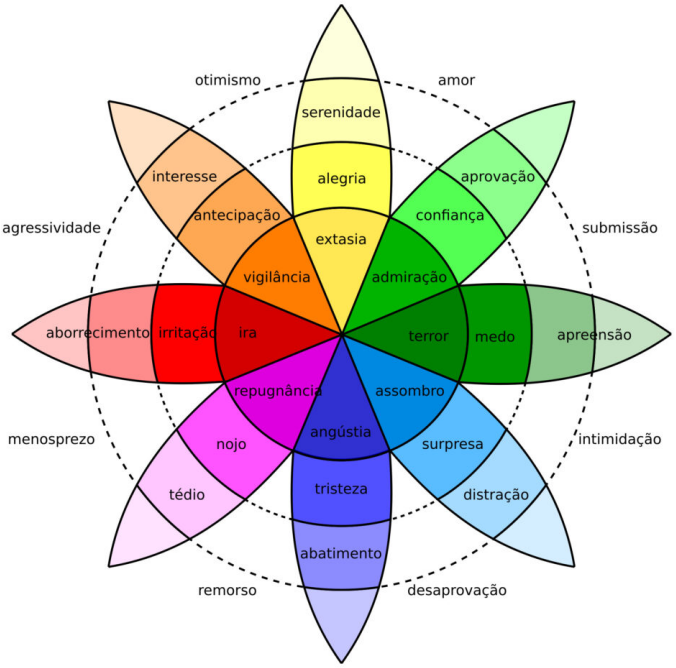


Figura 10 – Roda de Emoções traduzida para o Português. Fonte: Plutchik (2001), retirado do site Vitat (2025).

Nesta frase, as palavras **ótimo** e **cara** são cláusulas positiva e negativa, respectivamente, então uma polaridade é relacionada a elas

"O Spettus é ótimo(+1) e a comida não é cara(-1) ."

Mais um passo foi feito neste exemplo para tratar exceções em expressões negativas, em casos que elas não necessariamente indicam que a expressão é negativa. Neste exemplo a palavra **não** está discordando do adjetivo negativo **cara**, o que indica que esta é uma cláusula positiva, então a polaridade é atualizada.

"O Spettus é ótimo(+1) e a comida não é cara(+1) ."

O último passo mencionado por Silva (2010) é o relacionamento das palavras opinativas com a palavra que ela qualifica. Neste exemplo, **Spettus**, que é um substantivo próprio para o nome de um restaurante, tem a polaridade de **ótimo**, e **comida** tem a polaridade do termo **cara**. No fim, esta frase tem uma pontuação de sentimento positivo.

Em seu trabalho, Medhat, Hassan e Korashy (2014) fazem pesquisas sobre algoritmos para análise e classificação de sentimentos, neste trabalho são apresentadas técnicas de aprendizado de máquina supervisionado e não-supervisionado. Quanto a algoritmos supervisionados, aqueles em que existem bases de dados com itens já previamente treinados e classificados, o trabalho de Medhat, Hassan e Korashy (2014) aborda classificadores probabilísticos como Naive Bayes, Máquina de vetores de suporte (SVM) e árvore de

decisão. Sobre aprendizado não-supervisionado, que é usado quando bases treinadas previamente não existem ou não estão disponíveis, o artigo de Medhat, Hassan e Korashy (2014) explora técnicas de classificação de sentimentos baseadas em dicionários, corpus e semântica. Ainda nas técnicas não-supervisionadas, o trabalho explica sobre o uso de PLN nos métodos de análise, e apresenta trabalhos de referência que fazem uso destas técnicas mencionadas.

Kaur e Sharma (2020) fizeram uma análise por hashtags da rede social Twitter usando algoritmos parecidos com os mencionados nos trabalhos anteriores. Este trabalho coletou comentários da rede social que estavam com as hashtags *#Women* e *#MeToo*, aplicou inicialmente um algoritmo de atribuição de polaridades, semelhante ao trabalho de Silva (2010), identificando comentários positivos, negativos e neutros. Após isto, os comentários passaram por algoritmos de classificação, alguns já mencionados, como Naive Bayes e SVM, mas também outros dois novos - Random Forest e *Logistic Regression*. No fim é feita uma comparação com os quatro métodos com objetivo de encontrar o com maior acurácia.

A ferramenta EMTk, desenvolvida no trabalho de Calefato, Lanubile e Novielli (2017) é um *framework* de análise de texto escrita em Java, Python e R treinada para analisar e identificar seis emoções diferentes em texto - Amor, Alegria, Surpresa, Raiva, Tristeza e Medo. Com modelos treinados por SVM e a interface da linguagem R, os modelos são treinados para identificar a presença ou falta de palavras que indicam uma das seis emoções diferentes usando estruturas de árvores hierárquicas. O trabalho apresentou bons resultados e o *framework*, que está disponível abertamente no GitHub, foi usado em outros trabalhos como suporte para tarefas de classificação de emoções e identificação de polarizações em sentimentos.

O projeto BERT - sigla para *Bidirectional Encoder Representations from Transformers*; é um modelo de representação de linguagem Devlin et al. (2018) utilizado nas pesquisas do navegador Google. Como o nome sugere, o BERT é bidirecional, este algoritmo leva em conta palavras a esquerda e a direita de um termo para aplicar a ele um contexto.

Este algoritmo usa inteligência artificial para analisar o que foi escrito pelo usuário em busca de palavras que indiquem opiniões, sentimentos e emoções, para atribuir à pesquisa algum contexto, e daí então recomendar ao usuário os itens que melhor correspondem aos termos buscados.

A partir do BERT, outros algoritmos foram desenvolvidos usando-o como referência em busca de trazer aprimoramentos. O modelo EmoRoBERTa Ghoshal (2021), por exemplo, pega as estratégias usadas pelo BERT e treina o algoritmo usando quantidades maiores de dados para obter melhores resultados.

Capítulo 3

Desenvolvimento e Análise dos Resultados

Neste capítulo são apresentadas as etapas de realização deste trabalho, começando pela extração de comentários, passando eles para o pré-processamento, identificação e classificação das emoções nos comentários e for fim a exibição dos resultados. Todas as bibliotecas mencionadas neste capítulo são da linguagem Python.

As APIs principais usadas para desenvolver este projeto foram a API do site SearchAPI que monitora a página de vídeos em alta do Youtube - escolhida por monitorar diariamente todas as atualizações a página; e a API própria do YouTube, disponibilizada pelo próprio Google - esta API foi escolhida pois retorna a lista completa de todos os comentários de um vídeo.

Para tratamento dos comentários, foi escolhido o modelo de linguagem EmoRoBERTa, que identifica e classifica emoções em texto. Porém, como este modelo funcione, o texto nos comentários precisa estar em inglês. Para que os comentários possam ser classificados, foi escolhida a biblioteca DeepTranslator, que traduz gratuitamente textos de forma rápida.

3.1 Extração de Comentários

Para realização deste trabalho foram selecionados vídeos da página *"Em Alta"* do YouTube. Como pode ser visto na Figura 11, esta página é dividida em quatro seções - Agora (seção que compila os principais vídeos em alta na plataforma sem separá-los por categoria), Música, Filmes e Jogos. Cada uma destas divisões engloba os vídeos que estão sendo mais assistidos pelos usuários nas últimas 24 horas.

O primeiro passo foi obter os links de cada vídeo desta página. Para fazer esta tarefa foi usada a API YouTube Trends, disponibilizada pelo site Search API. Esta API monitora a aba de tendências no YouTube e retorna todas as informações públicas dos vídeos listados em todas as seções. Assim foi possível obter o identificador único de cada vídeo

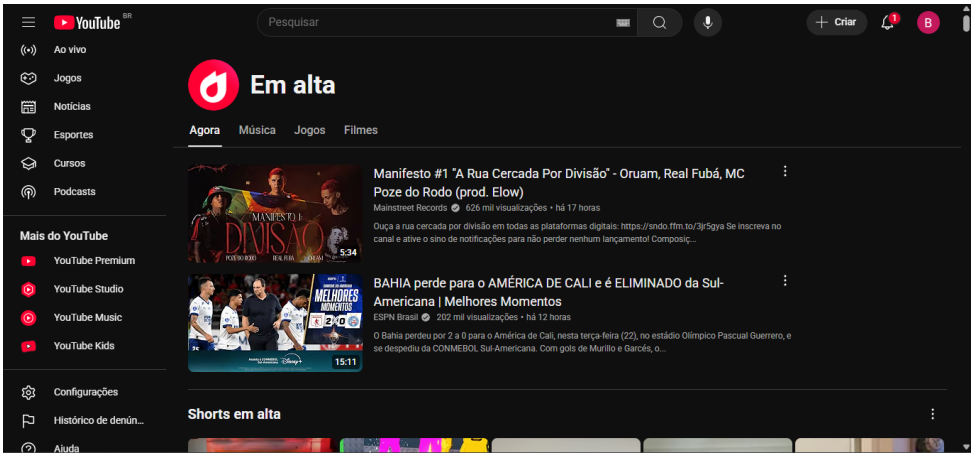


Figura 11 – Página de vídeos em alta no YouTube. Dela foram extraídos os comentários usados neste trabalho.

que compõe o link para encontrá-lo. Com esta API também é possível obter oturas informações, como o título do vídeo e idioma de origem, por exemplo. Com ela foram extraídos todos os identificadores dos vídeos em cada uma das quatro seções da página.

Com os identificadores dos vídeos em alta obtidos, o próximo passo foi extrair os comentários de cada vídeo. O YouTube possui uma API própria disponibilizada pelo Google que permite obter a lista completa de todos os comentários públicos

3.2 Pré-Processamento

Após a extração, os comentários foram traduzidos para inglês com a biblioteca *Deep Translator* da linguagem Python, que faz uso do tradutor de texto do Google. Esta etapa é necessária pois os modelos de classificação e identificação de emoções mais precisos trabalham apenas com o idioma inglês. As traduções em geral são bastante precisas, o único problema é que algumas gírias e expressões regionais que passam sem tradução ou com significado incorreto. Outro problema encontrado nesta etapa de pré-processamento foi o tempo gasto para tradução. A biblioteca de tradução exige um pequeno intervalo de 0,1 segundo entre cada solicitação, mas, como são muitos comentários para muitos vídeos, o tempo total de tradução de todos os comentários pode levar um tempo considerável. Além disso a biblioteca tem um limite de solicitações por dia, fazendo que em alguns casos não fosse possível traduzir todos os comentários em um único dia.

3.3 Classificação

Uma vez traduzidos, os comentários passaram pelo modelo de linguagem EmoRoBerta, que usa o algoritmo BERT(DEVLIN et al., 2018), discutido no capítulo anterior, como base para buscar emoções em texto e classificá-los. O modelo analisa cada comentário

individualmente e atribui uma pontuação e uma emoção a ele. Este algoritmo reconhece vinte e oito emoções diferentes, mas, para este trabalho, as emoções foram condensadas em oito emoções princicais - Alegria, Medo, Raiva, Tristeza, Nojo, Amor, Curiosidade e Surpresa. A Tabela 1 mostra o agrupamento e quais emoções compõem cada grupo de emoções principais.

Emoção Principal	Emoções Englobadas
Alegria	Felicidade, Alívio, Diversão, Gratidão, Orgulho, Otimismo, Realização, Excitação
Medo	Medo, Nervosismo, Excitação, Diversão
Raiva	Raiva, Irritação, Nervosismo, Remorso
Tristeza	Tristeza, Luto, Solidão, Arrependimento, Decepção
Nojo	Nojo, Constrangimento, Remorso, Decepção, Desaprovação
Amor	Amor, Admiração, Aprovação, Cuidado, Desejo
Curiosidade	Curiosidade, Excitação, Diversão
Surpresa	Surpresa, Admiração, Confusão

Tabela 1 – Categorias de emoções principais e suas emoções englobadas.

Esta consolidação foi sugerida pela psicológa Marília Camargo Tuma. É importante notar que algumas emoções englobadas estão presentes em mais de uma emoção principal por conta da subjetividade deste assunto. Além disso, o EmoRoBerta também pode classificar um texto como Neutro, porém, para este trabalho, foram ignorados todos os comentários indicados como neutros.

Após a classificação dos comentários individuais, cada vídeo é pontuado para todas as oito emoções principais e logo em seguida ranqueados, gerando ao todo oito ranqueamentos - um para cada emoção.

3.4 Exibição dos Resultados

Foi desenvolvida uma interface onde o usuário pode selecionar uma das oito emoções principais discutidas anteriormente, conforme pode ser visto na Figura 13.

O usuário pode escolher uma emoção e apertar em ”Pronto”. Logo em seguida, será redirecionado para uma nova página onde são apresentados os cinco vídeos que possuem melhor ranqueamento da emoção selecionada com base na análise dos comentários, como pode ser visto na Figura 13.

Para cada vídeo é montado um bloco onde está a capa do vídeo e um gráfico que mostra a porcentagem encontrada de cada uma das oito emoções. É possível assistir a maioria dos vídeos pela interface, porém, por conta de limitações impostas pelos canais em que o vídeo foi publicado, alguns deles só podem ser assistidos quando se aperta o botão que redireciona para o próprio YouTube, como pode ser visto na Figura 14.

Figura 12 – Primeira página da interface. O texto acima dos botões introduz o usuário ao objetivo do trabalho.

3.5 Experimentos

Quando o usuário termina de assistir os vídeos exibidos, é pedido que ele deixe sua avaliação em um formulário feito para avaliar a experiência tida com o trabalho. O formulário foi hospedado no Google Forms e consiste de três perguntas e uma caixa de texto onde o usuário pode deixar seus comentários e sugestões. As perguntas foram as seguintes:

1. O que você achou da interface?
2. O que você achou das sugestões de vídeos? Considere as pontuações apresentadas.
3. Qual a probabilidade de você usar um aplicativo como este caso ele fosse integrado ao YouTube?

Cada pergunta foi respondida com a possibilidade de dar notas de um até cinco, onde cinco é a melhor nota e indica grande satisfação com o trabalho, como pode ser visto na Figura 15.

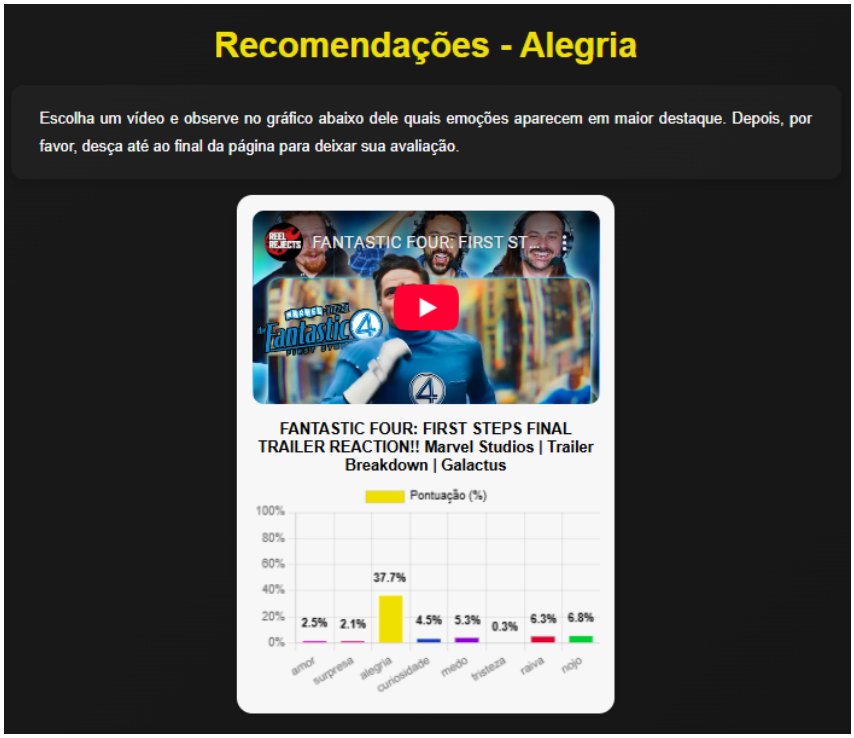


Figura 13 – Primeiro colocado nos vídeos de Alegria.

Trinta e três estudantes da Universidade Federal de Uberlândia se voluntariaram para responder. Logo abaixo, as Figuras 16, 17 e 18 mostram os gráficos que sintetizam as avaliações coletadas.

3.6 Avaliação dos Resultados

Para a questão 1 - “O que você achou da interface?”, é notável que os usuários em grande parte aprovam o que viram. Com uma média de 4,06 estrelas, 33,3% marcaram cinco estrelas, 45,5% marcaram quatro estrelas, 15,2% marcaram três estrelas, 6,1% marcaram duas estrelas e ninguém marcou uma estrela, como mostra o gráfico da Figura 16.

Já na questão 2 - “O que você achou das sugestões de vídeos?”, os resultados indicam ainda maior satisfação com o trabalho. O gráfico da Figura 17 mostra uma média de 4,33 estrelas, onde 54,5% atribuíram nota cinco estrelas, 27,3% marcaram quatro estrelas, 15,2% marcaram três estrelas, 3% marcaram duas estrelas, e novamente, ninguém marcou uma estrela.

A questão 3 pergunta “Qual a probabilidade de você usar um aplicativo como este caso fosse integrado ao YouTube?”, e mais uma vez as avaliações denotam boas impressões quanto a proposta apresentada. De acordo com o gráfico da Figura 18, 51,5% marcaram como ”Muito Alta” a probabilidade de usar uma aplicação como esta caso ela existisse no YouTube oficial, e ninguém marcou como ”Muito Baixa” a probabilidade.

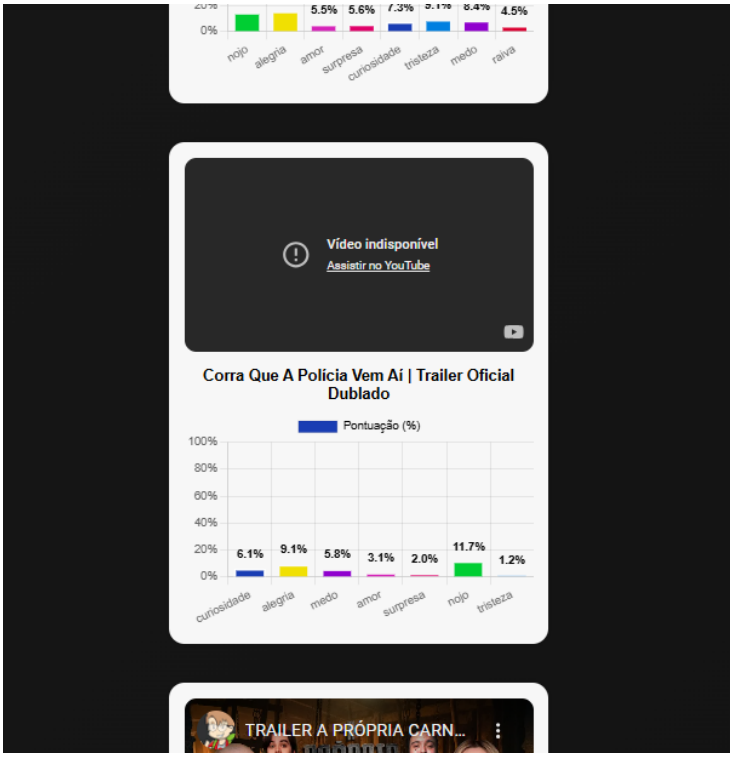


Figura 14 – Quarto colocado nos vídeos de Curiosidade. O canal em que o vídeo foi publicado impede sua exibição em lugares fora do YouTube.

Dos trinta e três participantes, vinte e sete deixaram um comentário avaliando o trabalho. Para sintetizar os resultados, os comentários foram agrupados em quatro categorias:

1. Elogios - Comentários que expressam satisfação quanto ao trabalho, mas não apresentam sugestões de melhorias.

2. Reclamações - Comentários que expressam insatisfação por conta de problemas encontrados na aplicação em sua fase de testes.

3. Sugestão Factível - Comentários que sugerem ideias possíveis de serem implementadas.

4. Sugestão Aleatória - Comentários que sugerem ideias que não podem ser implementadas; devido a falta de conhecimento dos usuários na dificuldade da implementação de tais ideias - ou que fogem da proposta do trabalho.

Os elogios ao trabalho estão logo abaixo. Todo o texto foi mantido da forma como os usuários escreveram sem nenhuma alteração.

- ❏ *Achei a proposta interessante, tendo em vista que muita das vezes clicamos em um vídeo sem saber o que vamos estar assistindo, então ter uma visão sobre o vídeo deixaria essa experiência mais satisfatória.*

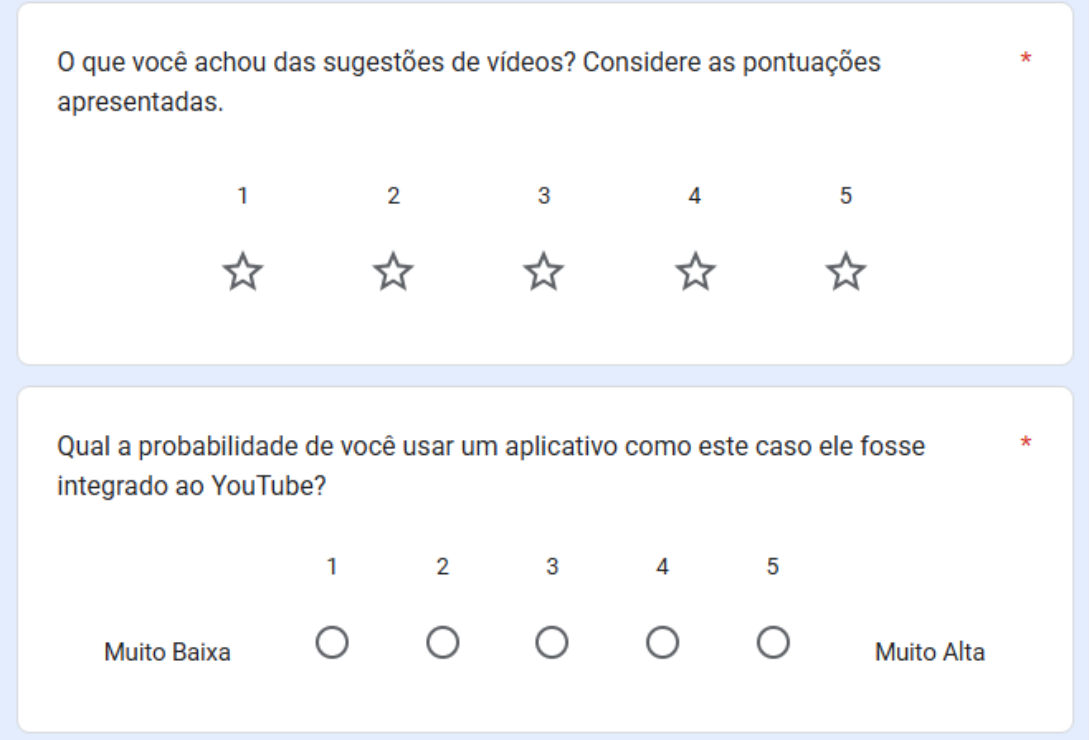


Figura 15 – As duas primeiras perguntas são medidas em estrelas, e a terceira pergunta mede a probabilidade de uso.

- ❏ *gostei bastante*
- ❏ *ideia bem maneira, agora que o youtube tirou os dislikes é uma boa ajuda*
- ❏ *muito bom, boa sorte na sua caminhada!*
- ❏ *Do geral, ferramenta interessante, interface intuitiva e com boa proposta.*
- ❏ *Mas parabéns! Muito bom o trabalho.*
- ❏ *Gostei bastante da interface*

Quanto as reclamações:

- ❏ *nao tinha som*
- ❏ *a interface ficou pouco intuitiva, principalmente quando um recurso e selecionado, a pagina volta ao inicio sem motivo aparente*
- ❏ *alguns videos estão aparecendo com outros sentimentos maiores do que o solicitado, por exemplo, o video em primeiro no ranking de nojo tinha mais comentarios refletindo tristeza do que nojo em si.*

A falta de som foi um problema encontrado em alguns computadores das salas aonde os testes foram feitos. A questão do botão de "Pronto" fazer a tela retornar ao início será discutida logo em seguida nas sugestões.

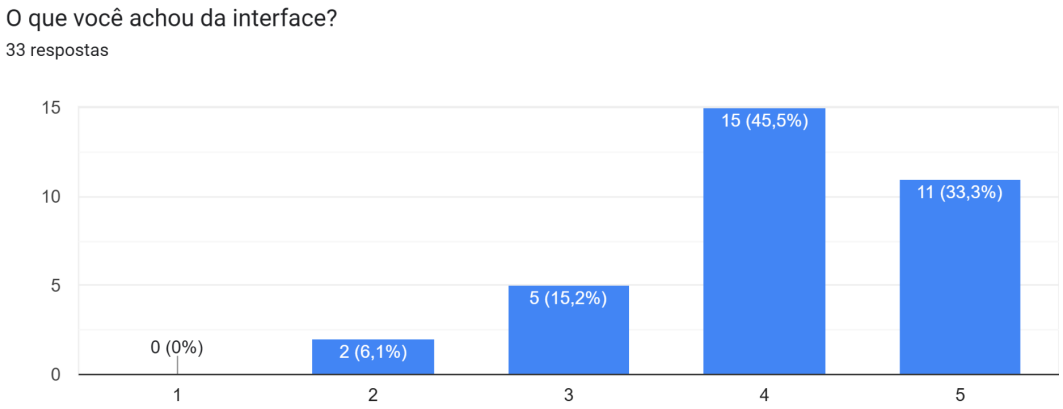


Figura 16 – Gráfico com as respostas da pergunta 1 - O que você achou da interface?.

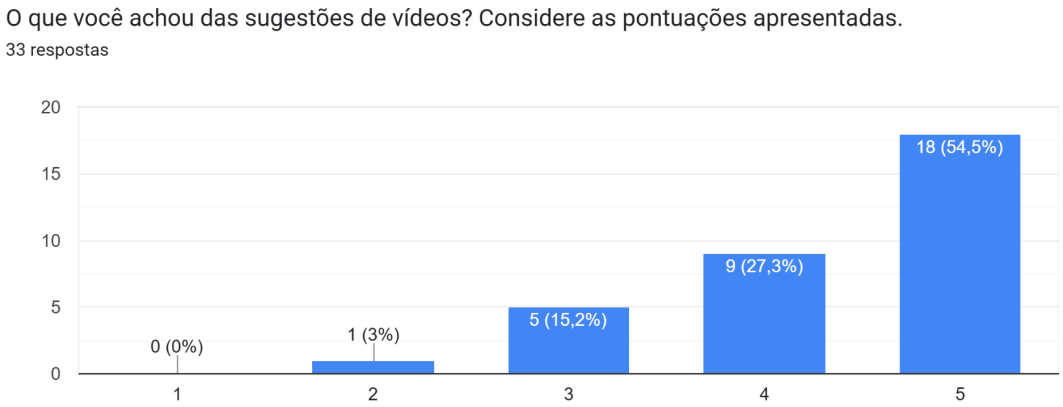


Figura 17 – Gráfico com as respostas da pergunta 2 - O que você achou das sugestões de vídeos? Considere as pontuações apresentadas.

Sobre a presença de outras emoções maiores do que as selecionadas, isto acontece pois o YouTube disponibiliza de trinta a cinquenta vídeos para cada seção de vídeos em alta - uma amostra relativamente pequena e que muda diariamente. Então, mesmo que, por exemplo, os vídeos de nojo apresentem mais comentários de tristeza do que nojo, eles são os vídeos com maior quantidade de comentários expressando nojo em seu determinado dia.

Os comentários com Sugestões Factíveis estão logo a seguir. Todo o texto foi mantido da forma como os usuários escreveram sem nenhuma alteração.

Figura 18 – Gráfico com as respostas da pergunta 3 - Qual a probabilidade de você usar um aplicativo como este caso ele fosse integrado ao YouTube?

- *Acho que o botão "pronto" não faz muito sentido, o usuário poderia apenas clicar na emoção desejada e a aplicação já retornaria o resultado.*
- *Minha sugestão seria, ao selecionar uma emoção, que ela fosse indicada em algum lugar da interface, talvez ao lado do botão pronto, já que quando clicamos em uma emoção a pagina está sendo colocada para cima.*
- *A interface é legal e bem simples, se a ideia foi acessibilidade, muito bom, mas se quiser deixar mais detalhado e menos "forte" os tons, deixa em tons pastéis*
- *Gostei bastante da interface e acredito que poderia ser acrescentado mais algumas emoções para a lista como por exemplo: ansiedade*
- *O design poderia ser um pouco mais lindo, e também ter uma confirmação sobre qual foi a emoção escolhida pelo usuário pois acaba ficando meio confuso como se usa o site*
- *Filtrar o estilo de video também além das emoçoes, tipo filtrar por jogos, filmes, programas de tv, etc...*
- *Ao invés de clicar em um botão para para ir para a aba dos videos, poderia fazer eles aparecerem diretamente quando a emoção for clicada. Poderia tambem adaptar o programa para mostrar videos apenas na linguagem especifica do usuario.*
- *Sobre a interface gráfica eu acho que poderia melhorar um pouco no sentido de usabilidade, pois ao escolher a emoção, já me levou para cima, precisando voltar e ainda clicar no button pronto.*

- *acho que deveria ter um tema especifico, é muito disperso os videos entre as recomendações.*
- *melhorar o tamanho dos botões em relação a interface*
- *poderia ter uma confirmação de qual emoção foi escolhida pelo usuário.*

Algumas sugestões e uma reclamação mencionada anteriomente envolvem o botão de "Pronto" e a ideia de que seria mais intuitivo que ao escolher uma emoção os vídeos apareçam logo abaixo imediatamente. O botão foi colocado pois é uma opção mais simples de desenvolver, já que esta foi apenas uma primeira etapa de testes. Alguns computadores apresentaram um erro com este botão, ao pressioná-lo a página subia um pouco e fazia com que o usuário tivesse que descer novamente até o botão. Este erro acontece em algumas telas com resoluções específicas, mas é facilmente corrigido em código e não afeta na execução do trabalho.

Outros comentários sugerem melhorias visuais específicas, como por exemplo, mais confirmações da emoção selecionada - que embora já exista no trabalho, não é de fácil visualização, de acordo com os usuários. Quando se escolhe uma emoção, o botão de "Pronto" muda de cor para a mesma cor da emoção escolhida, mas, de acordo com o feedback recebido isto não ficou de fácil visibilidade ao usuário. Para melhorar este aspecto poderiam ser adicionados textos na página de exibição dos vídeos que lembrem quem está usando qual foi a emoção selecionada.

Muitas sugestões pedem mais filtros, sejam eles mais emoções, opções de idioma e separação por categorias, como por exemplo filmes, jogos ou música, assim como no YouTube. A escolha das oito emoções principais tem como base trabalhos de psicologia, porém, é sim possível separá-las em mais opções, uma vez que o modelo de linguagem usado no desenvolvimento deste trabalho reconhece vinte e oito sentimentos diferentes. Opções de idioma e categorias podem sim ser implementadas, mas por conta da quantidade pequena de vídeos em alta por dia foi decidido que seria melhor não fazer tais divisões.

Já comentários com sugestões aleatórias, que não podem ser implementadas estão logo a seguir. Todo o texto foi mantido da forma como os usuários escreveram sem nenhuma alteração.

- *seria bom se pegasse todas as recomendacoes do youtube.*
- *podia tirar as opções de sentimentos ruins*
- *pode-se fazer uma análise e procurar comentários "maldosos" para evitar que apareça esse tipo de ódio.*
- *faz pro youtube da tv*
- *adicionar uma opção de colocar link para um video e fazer a mesma análise*

- *seria interessante se pudesse fazer a mesma coisa com as outras recomendaçoes do site*

Alguns sugerem que todas as recomendações que aparecem no YouTube fossem usadas no trabalho, e embora isto seja o ideal, existem questões de privacidade na plataforma que impedem que isso aconteça. O serviço de API do YouTube retorna apenas conteúdo que é público a todos os usuários, então não é possível adaptar este trabalho para analisar as recomendações individuais de cada usuário. Por conta desta limitação também não foi possível adaptar esta proposta para o YouTube de televisões, como alguns também sugeriram.

Remover emoções negativas e desconsiderar comentários maldosos foram algumas propostas. Porém, uma das ideias principais deste trabalho é encontrar os vídeos que estão em alta mas não foram bem recebidos por quem os assistiu, e as emoções negativas e os comentários maldosos são importantes para esta parte.

E por fim, a ideia de receber um link para um vídeo e fazer toda a análise de emoções também foi cogitada, o problema é que, com as biblitecas de tradução gratuitas disponíveis atualmente, o tempo de processamento gasto para traduzir os comentários é muito grande devido a exigência de um pequeno intervalo entre requisições, fazendo com que o tempo de execução total do programa leve um tempo considerável. Existem bibliotecas pagas de tradução que removem o impasse do limite de requisições, porém, para o escopo deste trabalho, não há orçamento. Por conta destes problemas, não é possível fazer a mesma análise para um link passado pelo usuário em um tempo de execução razoável.

Capítulo 4

Conclusão

A proposta original do trabalho de desenvolver um sistema que classifica e ranqueia vídeos do YouTube foi possível de se executar usando duas APIs - A API própria do YouTube desenvolvida pelo Google para extrair comentários dos vídeos da plataforma, e uma API do site SearchAPI desenvolvida para monitorar os vídeos em alta do YouTube e retornar informações sobre eles, como os links para os vídeos e seus identificadores.

Os comentários foram traduzidos para inglês usando a biblioteca Deep Translator e após isso todos eles passaram pelo modelo de classificação de sentimentos EmoRoBERTa. Após isso todos os vídeos foram ranqueados em oito emoções principais.

Foi feita uma interface que permite ao usuário escolher uma das oito emoções, e mostra para ele os cinco vídeos mais bem colocados para aquela emoção. Embaixo de cada vídeo é exibido um gráfico mostrando sua classificação em todas as oito emoções, com destaque para a selecionada. Por fim é oferecido ao usuário um formulário de avaliação onde pode-se dar uma nota para a aplicação e enviar comentários com sugestões e críticas.

Como visto no capítulo anterior, o aplicativo foi bem avaliado, portanto cumprindo o objetivo deste trabalho. As sugestões e críticas estão tratadas na seção seguinte de trabalhos futuros.

4.1 Trabalhos Futuros

1. Estudar bibliotecas pagas de tradução e análise de sentimentos que não possuam limitações de solicitações para que se torne possível implementar a ideia do usuário entrar com links para fazer a análise de sentimentos em vídeos escolhidos por ele em tempo real.

2. Estudar uma melhor forma de fazer a seleção da emoção na tela inicial. Como por exemplo, remover o botão de “Pronto”, ou adicionar indicadores da emoção escolhida.

3. Adicionar mais filtros na seleção, como por exemplo, opções de idioma e de divisões por categorias.

4. Adicionar mais confirmações visuais da opção escolhida para que o usuário se lembre do quê foi selecionado na página de ranqueamento.

5. Estudar a possibilidade de viabilizar a implantação deste aplicativo em outros dispositivos.

6. Adaptar a aplicação para o formato atual de vídeos Em Alta introduzido no YouTube em julho de 2025.

Referências

AML.COM. **What are the advantages and disadvantages of Bag-of-Words model?** 2024. Disponível em: <https://aiml.com/what-are-the-advantages-and-disadvantages-of-bag-of-words-model/>. Acesso em: 07 july 2025. Citado 2 vezes nas páginas 3 e 19.

APTÉ, C.; DAMERAU, F.; WEISS, S. M. Automated learning of decision rules for text categorization. **ACM Transactions on Information Systems (TOIS)**, ACM New York, NY, USA, v. 12, n. 3, p. 233–251, 1994. Citado na página 19.

BACCOURI, N. **Deep Translator**. 2020. Disponível em: <https://pypi.org/project/deep-translator/>. Acesso em: 02 september 2025. Citado na página 23.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca**. [S.l.]: Bookman Editora, 2013. Citado na página 18.

BARDIN, L.; RETO, L.; PINHEIRO, A. **Análise de conteúdo**. Edições 70, 1977. ISBN 9789724408989. Disponível em: <https://books.google.com.br/books?id=K-9vAAAACAAJ>. Citado na página 17.

BIEHL, M. **API architecture**. [S.l.]: API-University Press, 2015. v. 2. Citado na página 23.

BRAGA, L. P. V. B. **Introdução à Mineração de Dados-2a edição: Edição ampliada e revisada**. [S.l.]: Editora E-papers, 2005. Citado na página 15.

CALEFATO, F.; LANUBILE, F.; NOVIELLI, N. Emotxt: a toolkit for emotion recognition from text. In: IEEE. **2017 seventh international conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)**. [S.l.], 2017. p. 79–80. Citado na página 26.

DAMASIO, A. Descartes' error: Emotion, rationality and the human brain. **New York: Putnam**, v. 352, 1994. Citado na página 20.

DATAREPORTAL. **Digital 2025: top social platforms in 2025**. 2025. Disponível em: <https://datareportal.com/reports/digital-2025-sub-section-top-social-platforms.> Acesso em: 8 july 2025. Citado 2 vezes nas páginas 3 e 11.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: . [s.n.], 2018. Disponível em: <https://arxiv.org/abs/1810.04805>. Citado 3 vezes nas páginas 22, 26 e 28.

DEVMEDIA. **Aspectos teóricos da mineração de dados e aplicação das regras de classificação para apoiar o comércio**. 2024. Disponível em: <https://www.devmedia.com.br/aspectos-teoricos-da-mineracao-de-dados-e-aplicacao-das-regras-de-classificacao-para-apoiar-o-come 25429>. Acesso em: 16 april 2024. Citado 2 vezes nas páginas 3 e 14.

DING, X.; LIU, B.; YU, P. S. A holistic lexicon-based approach to opinion mining. In: **Proceedings of the 2008 international conference on web search and data mining**. [S.l.: s.n.], 2008. p. 231–240. Citado na página 24.

DUMAIS, S. et al. Inductive learning algorithms and representations for text categorization. In: **Proceedings of the seventh international conference on Information and knowledge management**. [S.l.: s.n.], 1998. p. 148–155. Citado na página 19.

EKMAN, P. Basic emotions, vol. 476 of handbook of cognition and emotion. t. dalgleish and m. **Power, John Wiley & Sons Ltd. Sussex UK**, 1999. Citado na página 20.

FAYYAD, U. M. et al. Knowledge discovery and data mining: Towards a unifying framework. In: **KDD**. [S.l.: s.n.], 1996. v. 96, p. 82–88. Citado 2 vezes nas páginas 3 e 14.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge discovery in databases: An overview. **AI magazine**, v. 13, n. 3, p. 57–57, 1992. Citado na página 14.

GHOSHAL, A. **EmoRoBERTa**. 2021. Disponível em: <https://huggingface.co/arpanghoshal/EmoRoBERTa>. Acesso em: 27 january 2025. Citado 2 vezes nas páginas 22 e 26.

HASAN, M. **Transformers in Natural Language Processing**. 2022. Citado 2 vezes nas páginas 3 e 17.

KAUR, C.; SHARMA, A. Social issues sentiment analysis using python. In: IEEE. **2020 5th international conference on computing, communication and security (ICCCS)**. [S.l.], 2020. p. 1–6. Citado na página 26.

KUKREJA, H. et al. An introduction to artificial neural network. **Int J Adv Res Innov Ideas Educ**, v. 1, n. 5, p. 27–30, 2016. Citado na página 16.

LIU, B. et al. **Web data mining: exploring hyperlinks, contents, and usage data**. [S.l.]: Springer, 2007. v. 1. Citado na página 24.

LOVINS, J. B. Development of a stemming algorithm. **Mech. Transl. Comput. Linguistics**, v. 11, n. 1-2, p. 22–31, 1968. Citado na página 18.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. **Ain Shams engineering journal**, Elsevier, v. 5, n. 4, p. 1093–1113, 2014. Citado 4 vezes nas páginas 20, 21, 25 e 26.

OPENCADD. **Redes neurais e avanços tecnológicos: o que têm em comum?** 2022. Disponível em: <https://www.opencadd.com.br/blog/redes-neurais-e-avancos-tecnologicos>. Acesso em: 07 july 2025. Citado 2 vezes nas páginas 3 e 16.

PLUTCHIK, R. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. **American scientist**, JSTOR, v. 89, n. 4, p. 344–350, 2001. Citado 3 vezes nas páginas 3, 24 e 25.

SAGE, D. **Facebook Users Statistics (2025) — Worldwide Data**. 2025. Disponível em: <https://www.demandsage.com/facebook-statistics/#:~:text=As%20of%202025%2C%20Facebook%20has,its%20monthly%20active%20user%20base.> Acesso em: 20 march 2025. Citado na página 10.

SILVA, N. G. R. da. **Bestchoice: Classificação de sentimento em ferramentas de expressão de opinião**. Tese (Doutorado) — Tese de graduação, Universidade Federal de Pernambuco, Recife, 2010. 7, 17, 2010. Citado 4 vezes nas páginas 20, 24, 25 e 26.

SINGH, A.; WEBER, A. **Processamento de linguagem natural com Python: Simplesmente em profundidade**. Babelcube Incorporated, 2021. ISBN 9781667404660. Disponível em: <https://books.google.com.br/books?id=Z4s9EAAAQBAJ>. Citado 3 vezes nas páginas 16, 17 e 18.

STATISTA. **Leading countries based on YouTube audience size as of January 2024**. 2024. Disponível em: <https://www.statista.com/statistics/280685/number-of-monthly-unique-youtube-users/#:~:text=As%20of%20January%202024%2C%20India,users%20watching%20content%20on%20YouTube>. Acesso em: 13 april 2024. Citado na página 11.

_____. **Number of internet and social media users worldwide as of February 2025**. 2025. Disponível em: <https://www.statista.com/statistics/617136/digital-population-worldwide/>. Acesso em: 20 march 2025. Citado na página 10.

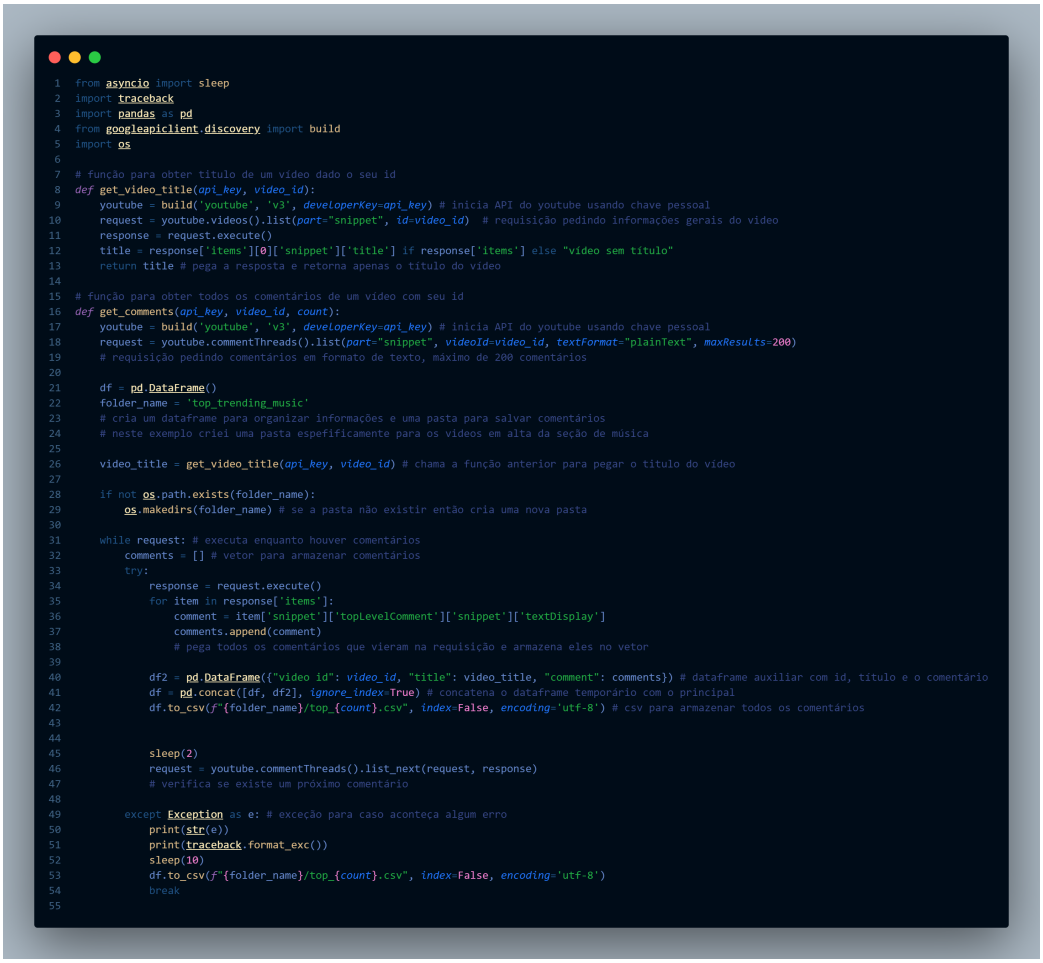
VITAT. **Roda das emoções: Saiba como identificar os sentimentos**. 2025. Disponível em: <https://vitat.com.br/roda-das-emocoes/>. Acesso em: 06 february 2025. Citado 2 vezes nas páginas 3 e 25.

WOLF, T. et al. Huggingface's transformers: State-of-the-art natural language processing. **CoRR**, abs/1910.03771, 2019. Disponível em: <http://arxiv.org/abs/1910.03771>. Citado na página 21.

ZANNETTOU, S. et al. The good, the bad and the bait: Detecting and characterizing clickbait on youtube. In: **2018 IEEE Security and Privacy Workshops (SPW)**. [S.l.: s.n.], 2018. p. 63–69. Citado na página 11.

ZENHA, L. Redes sociais online: o que são as redes sociais e como se organizam? **Caderno de Educação**, n. 49, p. 19–42, 2018. Citado na página 10.

Apêndices



```python
from asyncio import sleep
import traceback
import pandas as pd
from googleapiclient.discovery import build
import os

# função para obter titulo de um video dado o seu id
def get_video_title(api_key, video_id):
    youtube = build('youtube', 'v3', developerKey=api_key) # inicia API do youtube usando chave pessoal
    request = youtube.videos().list(part="snippet", id=video_id)  # requisição pedindo informações gerais do video
    response = request.execute()
    title = response['items'][0]['snippet']['title'] if response['items'] else "video sem titulo"
    return title # pega a resposta e retorna apenas o titulo do video

# função para obter todos os comentários de um video com seu id
def get_comments(api_key, video_id, count):
    youtube = build('youtube', 'v3', developerKey=api_key) # inicia API do youtube usando chave pessoal
    request = youtube.commentThreads().list(part="snippet", videoId=video_id, textFormat="plainText", maxResults=200)
    # requisição pedindo comentários em formato de texto, máximo de 200 comentários

    df = pd.DataFrame()
    folder_name = 'top_trending_music'
    # cria um dataframe para organizar informações e uma pasta para salvar comentários
    # neste exemplo criei uma pasta espefificamente para os videos em alta da seção de música

    video_title = get_video_title(api_key, video_id) # chama a função anterior para pegar o titulo do video

    if not os.path.exists(folder_name):
        os.makedirs(folder_name) # se a pasta não existir então cria uma nova pasta

    while request: # executa enquanto houver comentários
        comments = [] # vetor para armazenar comentários
        try:
            response = request.execute()
            for item in response['items']:
                comment = item['snippet']['topLevelComment']['snippet']['textDisplay']
                comments.append(comment)
                # pega todos os comentários que vieram na requisição e armazena eles no vetor

            df2 = pd.DataFrame({"video id": video_id, "title": video_title, "comment": comments}) # dataframe auxiliar com id, titulo e o comentário
            df = pd.concat([df, df2], ignore_index=True) # concatena o dataframe temporário com o principal
            df.to_csv(f"{folder_name}/top_{count}.csv", index=False, encoding='utf-8') # csv para armazenar todos os comentários


            sleep(2)
            request = youtube.commentThreads().list_next(request, response)
            # verifica se existe um próximo comentário

        except Exception as e: # exceção para caso aconteça algum erro
            print(str(e))
            print(traceback.format_exc())
            sleep(10)
            df.to_csv(f"{folder_name}/top_{count}.csv", index=False, encoding='utf-8')
            break

```

Figura 19 – Funções usadas para obter título e comentários de um vídeo passando seu identificador.

```python
import json
import requests
import comments_trends as ct
# comments_trends é o nome do arquivo com as funções para
# extrair comentários e título do vídeo

url = "https://www.searchapi.io/api/v1/search" # link do SearchAPI

params = { # parâmetros usados para usar a API YouTube Trends
  "engine": "youtube_trends", #SearchAPI fornece várias APIs diferentes
  # este parâmetro define qual delas usar
  "gl": "br", # define qual o país da página de vídeos em alta
  "bp": "music", # define qual divisão da página, pode ser now, music, films ou gaming
  "api_key": "chave Search API" # chave pessoal do SearchAPI
}

ids = [] # vetor para guardar identificadores e outras informações dos vídeos
yt_api_key = "chave YouTube API" # chave pessoal do YouTube API

response = requests.get(url, params = params)
data = response.json() # faz uma requisição para a api

trends = data.get('trending', []) # pega a resposta da requisição

for item in trends:
    dict = {"Title": item['title'],"Position": item['position'], "Video_id":item['id']}
    ids.append(dict) # extrai da resposta recebida o titulo, posição no ranking e id do video

with open("trending_videos_music.json", "w", encoding="utf-8") as final:
  json.dump(ids, final,ensure_ascii=False, indent=4)

with open('trending_videos_music.json', 'r', encoding='utf-8') as f:
    dados = json.load(f)

    # escreve e salva num arquivo json tudo que foi recebido

count = 0
for video in dados: # chama a função de extrair comentarios para todos os videos
     count = count + 1 # contador usado na função
     ct.get_comments(yt_api_key,video['Video_id'], count)
```

Figura 20 – Código usado para obter os identificadores dos vídeos da página **Em Alta** do YouTube.

```python
from deep_translator import GoogleTranslator
import csv
import time
import os


folder_name = "top_trending_gaming" # pasta principal com todas as traduções
folder_name_translate = 'translate_gaming' # subpasta especifica para videos de jogos
arquivos = os.listdir(folder_name)  # lista de todos os arquivos na pasta principal
count = 0

if not os.path.exists(folder_name_translate):
    os.makedirs(folder_name_translate) # se a subpasta não existir cria ela


for arq in arquivos:
    if os.path.exists(f'{folder_name}/{arq}'):
        count += 1
        arq2 = f'{folder_name_translate}/top_{count}_translate.csv'
        # cria arquivos para cada video em alta do primeiro até o último

        with open(f'{folder_name}/{arq}', 'r', encoding='utf-8') as leitura, open(arq2, 'w', newline='', encoding='utf-8') as escrita:
            reader = csv.reader(leitura)
            writer = csv.writer(escrita)
            count_linhas = 0 # conta quantas linhas tem no comentário


            for linha in reader:
                if count_linhas >= 100:
                    break # comentários com muitas linhas travam o tradutor

                if(len(linha) >= 2): # toda linha do csv tem que ter os 3 atributos
                    video_id = linha[0]
                    title = linha[1]
                    comment = linha[2]

                    try:
                        translated = GoogleTranslator(source='auto', target='en').translate(comment)
                        writer.writerow([video_id, title, translated])
                        # detecta idioma automaticamente, traduz e escreve

                        time.sleep(0.1)
                        count_linhas += 1

                    except Exception as e:
                        print(e)

    else:
        print(f"Arquivo {arq} não encontrado")


```

Figura 21 – Código usado para traduzir os comentários.

```python

from transformers import RobertaTokenizerFast, TFRobertaForSequenceClassification, pipeline
import os, csv

DATA_DIR = 'data'
respostas = []
tokenizer = RobertaTokenizerFast.from_pretrained("EmoRoBERTa") # tokenizador do modelo emoroberta
model = TFRobertaForSequenceClassification.from_pretrained("EmoRoBERTa") # carrega o modelo de linguagem

emotion = pipeline('sentiment-analysis',
                   model='EmoRoBERTa') # cria o pipeline para classificar sentimentos

folder_name_translate = 'translate_films' # pasta aonde estão os comentários da aba de filmes traduzidos
folder_name_emotion = 'resultado' # nova pasta para guardar as classificações
arquivos = os.listdir(folder_name_translate) # lista todos os arquivos dentro da pasta
count = 0

if not os.path.exists(folder_name_emotion):
    os.makedirs(folder_name_emotion) # se a pasta não existe cria ela

for arq in arquivos:
    if os.path.exists(f'{folder_name_translate}/{arq}'):
        count += 1
        arq2 = f'{folder_name_emotion}/films_top_{count}.csv'

        with open(f'{folder_name_translate}/{arq}', 'r', encoding='utf-8') as leitura, open(arq2, 'w', newline='', encoding='utf-8') as escrita:
            reader = csv.reader(leitura)
            writer = csv.writer(escrita)

            writer.writerow(['video_id', 'title', 'comment', 'emotion_label', 'emotion_score'])
            # escreve tudo que já tem nos comentários + classificação da emoção e pontuação

            respostas = []
            count_linhas = 0
            for linha in reader:
                if len(linha) >= 2: # linhas precisam ter os 3 atributos
                    video_id = linha[0]
                    title = linha[1]
                    comment = linha[2]

                    try:
                        emotion_labels = emotion(comment) # função que classifica emoções em texto

                        if emotion_labels and isinstance(emotion_labels, list):
                            emotion_label = emotion_labels[0]['label'] # nome da emoção
                            emotion_score = emotion_labels[0]['score'] # pontuação
                        else:
                            emotion_label = ''
                            emotion_score = ''

                        writer.writerow([video_id, title, comment, emotion_label, emotion_score])




                        count_linhas += 1
                    except Exception as e:
                        print({e})
                else:
                    print(f"Linha inválida ou vazia ignorada - {linha}")

    else:
        print(f"Arquivo {arq} não encontrado.")

```

Figura 22 – Código usado para identificar e pontuar a emoção principal encontrada em cada comentário.

```python
import os
import pandas as pd
from collections import defaultdict
import json

emotion_sets = {
    "tristeza": ["sadness", "grief", "loneliness", "regret", "disappointment"],
    "amor": ["love", "admiration", "approval", "caring", "desire"],
    "alegria": ["joy", "relief", "amusement", "gratitude", "pride", "optimism", "realization", "excitement"],
    "curiosidade": ["curiosity", "excitement", "amusement"],
    "medo": ["fear", "nervousness", "excitement", "amusement"],
    "nojo": ["disgust", "embarrassment", "remorse", "disappointment", "disapproval"],
    "raiva": ["anger", "annoyance", "nervousness", "remorse"],
    "surpresa": ["surprise", "admiration", "confusion"]
} # conjuntos de emoções agrupando em oito principais

pasta = 'resultado/'
arquivos_csv = [f for f in os.listdir(pasta) if f.endswith('.csv')] # verifica se todos os arquivos são csv

dfs = []
for arquivo in arquivos_csv:
    caminho = os.path.join(pasta, arquivo)
    df = pd.read_csv(caminho, header=None, names=["video_id", "title", "comment", "emotion_label", "emotion_score"])
    df = df[df['emotion_score'] != 'emotion_score']
    df['emotion_score'] = df['emotion_score'].astype(float)
    dfs.append(df) # pega apenas a pontuação, converte ela para tipo float e coloca em um dataframe

df_total = pd.concat(dfs, ignore_index=True)
total_por_emocao = df_total.groupby('emotion_label')['emotion_score'].sum().to_dict()
# soma as pontuações dos comentários de um vídeo para cada emoção e calcula qual a nota

df_total['normalized_score'] = df_total.apply(
    lambda row: row['emotion_score'] / total_por_emocao[row['emotion_label']]
    if row['emotion_label'] in total_por_emocao else 0,
    axis=1
) # calcula pontuação normalizada

scores = defaultdict(lambda: defaultdict(float)) # dicionario no formato 'id_video: pontuação (tipo float)'
titulos = {}

for _, row in df_total.iterrows():
    video_id = row['video_id']
    titulo = row['title']
    emotion = row['emotion_label']
    score = row['normalized_score']

    titulos[video_id] = titulo

    for emocao_principal, subemocoes in emotion_sets.items():
        if emotion in subemocoes:
            scores[video_id][emocao_principal] += score
            # calcula a pontuação final para todas as oito emoções principais
            # somando as pontuações de cada uma das sub-emoções dentro do respectivo
            # conjunto de emoções


resultados = {}
for video_id, emocao_scores in scores.items():
    resultados[video_id] = {
        'title': titulos[video_id],
        'scores': emocao_scores
    } # cria um novo vetor com titulos e pontuações

with open('pontos.json', 'w', encoding='utf-8') as f:
    json.dump(resultados, f, indent=4, ensure_ascii=False)
    # salva em json as pontuações para as 8 emoções de todos os videos

emotions = list(emotion_sets.keys())
top_videos = {}

for emotion in emotions:
    videos_scores = []
    for video_id, data in resultados.items():
        score = data['scores'].get(emotion, 0.0)
        videos_scores.append({
            "video_id": video_id,
            "title": data['title'],
            "score": score
        })

    top_5 = sorted(videos_scores, key=lambda x: x["score"], reverse=True)[:5]
    top_videos[emotion] = top_5

# todo o bloco acima cria um novo arquivo com os 5 videos mais bem qualificados
# para cada uma das 8 emoções principais

with open('top_5_por_emocao.json', 'w', encoding='utf-8') as f:
    json.dump(top_videos, f, indent=4, ensure_ascii=False)

# salva o arquivo em json
```

Figura 23 – Código usado para calcular os cinco vídeos mais bem qualificados para cada uma das emoções principais.