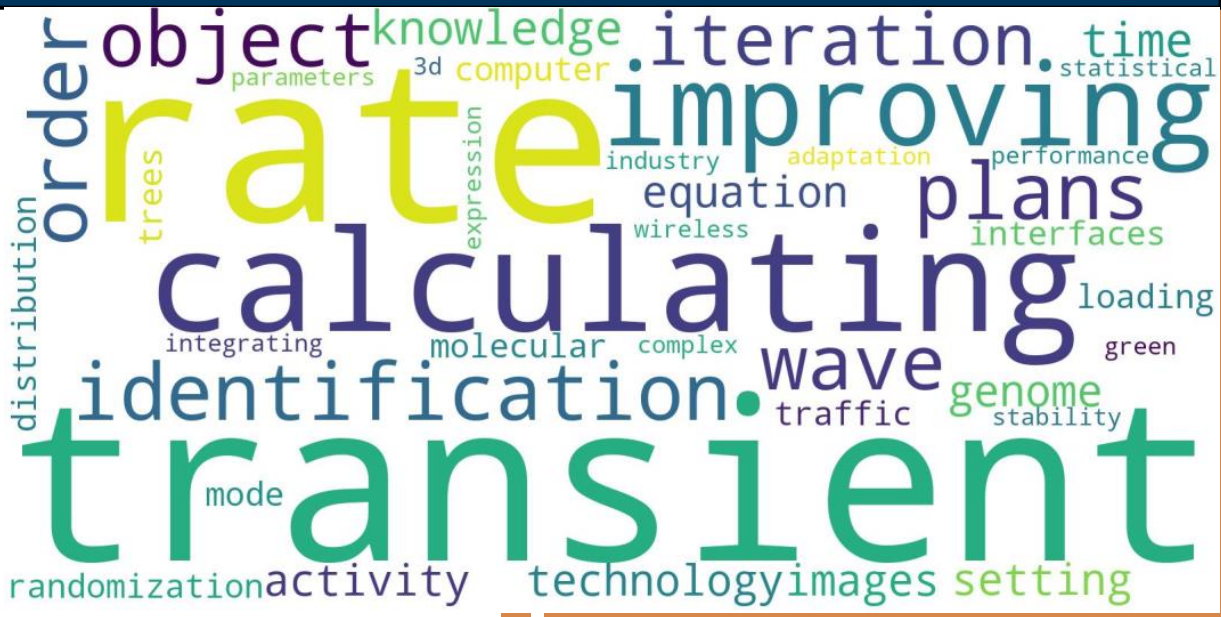
Programa de Engenharia de Sistemas e Computação

PESC Coppe

Relatório Técnico
PESC-782

Explorações em Mineração de Texto



Geraldo Xexéo, D.Sc.

Editor

COPPE UFRJ

UFRJ UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Explorações em Mineração de Texto

Editor:
Geraldo Xexéo

Autores:
Matheus Avellar de Barros, Pedro Boechat,
Carlos Cardoso Dias, Felipe Bevilaqua Guimarães,
Pedro Kuchpil, Gabriel S. Luna,
Guilherme C. Mattos, Vitor Paes,
Rodrigo Pagliusi, Bruno D. de Paiva,
Tales Mello Paiva, Rodrigo Peregrino,
João P. Leite Pinho, Glauco Primo,
Herbert Salazar dos Santos, Ana Clara Correa da Silva,
Fernando Costa de Souza, João Pedro G. de Souza,
Eduardo Vieira Marques Pereira do Valle, Victor Hugo Ventura,
Geraldo Xexéo

19 de agosto de 2022

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este relatório técnico apresenta 11 trabalhos em grupo dos alunos da cadeira de Busca e Mineração de Texto no segundo período de 2022, do Programa de Pós-Graduação em Sistemas e Computação da COPPE.

Nestes trabalhos são usadas diferentes técnicas de representação de texto e diferentes algoritmos de processamento, mineração e visualização de texto, de forma a obter resultados a perguntas que variam desde como tem funcionado a Câmara de Deputados até a Classificação Indicativa Automatizada.

Entre os tópicos abordados estão: *Tag Clouds*, Classificação de Textos, Análise de Sentimentos, Transformadores, Similaridade Textual, Sumarização, Extração de Palavras Chave, Modelos de Linguagem e Análise Comparativa de Algoritmos de Mineração de Texto.

Abstract

This technical report presents 11 group works by students of the Text Search and Mining course in the second period of 2022, at the Postgraduate Program in Systems and Computing at COPPE.

In these works, different text representation techniques and different text processing, mining and visualization algorithms are used, in order to obtain results for questions ranging from how the Brazilian House of Representatives has worked to an Automated Movie Rating System.

Among the topics covered are: Tag Clouds, Text Classification, Sentiment Analysis, Transformers, Textual Similarity, Summarization, Keyword Extraction, Language Models and Comparative Analysis of Text Mining Algorithms.

1 Introdução

Geraldo Xexéo
Programa de Engenharia de Sistemas e Computação - COPPE

Esse relatório técnico apresenta os trabalhos em grupo dos alunos da cadeira de Busca e Mineração de Texto no segundo período de 2022, do Programa de Pós-Graduação em Sistemas e Computação da COPPE.

Como professor da cadeira, fiquei muito satisfeito com a qualidade do trabalho entregue em prazo exíguo. Todos os trabalhos demonstram um esforço em seguir o ciclo completo de ter uma pergunta interessante, obter os dados, tratá-los e analisá-los por meio de pacotes de visualização e mineração de texto.

Decidi então fazer o relatório técnico para deixar registradas todas essas contribuições. Segundo a regra da cadeira, os trabalhos apresentados deveriam ter o tamanho de um Poster ou *Short Paper* tipicamente encontrados em congressos da SBC ou da IEEE, isso significa algo ao redor de 5 páginas em duas colunas e corpo 11. Aqui eles aparecem em uma formato que ocupa mais espaço, para facilitar o leitor, porém foram revistos apenas sua apresentação e português (ou inglês, em alguns casos).

Todos os trabalhos apresentam alguma originalidade em computação aplicada, seja na pergunta que é feita, seja na aplicação imediata, seja no uso de algum dado específico. O leitor fica avisado, porém, que nenhum algoritmo novo é apresentado, já que essa não era a finalidade da cadeira. Todos eles são também possíveis sementes para dissertações de mestrado e teses de doutorado.

Ressalto a presença de alunos de graduação do curso de Engenharia de Computação e Informação na turma, que apresentaram trabalhos no mesmo padrão que os alunos da pós-graduação.

1.1 Apresentação dos Trabalhos

Os quatro primeiros trabalhos apresentados usam *Word Clouds* ou *Tag Clouds*. Esse uso foi influenciado por um trabalho anterior meu, com Fernando Morgado e Patrícia Fiúza (Xexéo, F. Morgado e Fiuza, 2009).

O trabalho apresentado, no Capítulo 2, é "Análise de Semântica Latente (LSA) Aplicada a Projeto de Leis", de Carlos Cardoso Dias e Tales Mello Paiva, apresenta uma análise visual, na forma de *Word Clouds*, de projetos de lei da Câmara de Deputados, sendo que essa análise visual é criada por meio de um algoritmo de LSA. Essa é uma forma sofisticada de fazer a visualização por *Word Clouds*, sendo aplicada a dados interessantes. Essa linha de trabalho, a análise de comportamento ou desempenho de políticos, parece muito promissora.

No Capítulo 3, Bruno D. de Paiva e João P. Leite Pinho, alunos de graduação do curso de Engenharia de Computação e Informação, apresentam outra aplicação de Word Clouds, que mostra quais os cômodos mais comentados em relação aos aluguéis no *Airbnb*. Nesse caso, fazem também uma análise de sentimentos, permitindo a separação dos cômodos mais elogiados dos mais criticados. Essa tipo de trabalho tem grande interesse da indústria e os alunos souberam lidar bem, em relação ao curto período de tempo, com as bases de dados reais.

Já Pedro Boechat e Pedro Kuchpil, também alunos de graduação, no Capítulo 4, apresentaram um trabalho usado um conceito criado por mim (Xexéo, F. Morgado e Fiuza, 2009), *Differential Tag Clouds*, em uma abordagem para a análise das publicações de Computação e como elas vêm se alterando ao longo do tempo na UFRJ.

Devido a ocorrerem em paralelo, esses alunos, porém, não usaram uma contribuição feita por Guilherme Mattos, que criou a biblioteca de Python GTAGLIB, que cria Tag Clouds diferenciais, no Capítulo 5, já que ela só ficou disponível no final do período. Seria interessante ver, no futuro, essa biblioteca melhorada e usada em aplicações como as anteriores.

O cinema inspirarou dois trabalhos. Ana Clara Correa da Silva e Gabriel S. Luna, no Capítulo 6, buscaram aprender como fazer a classificação indicativa de filmes por meio de suas sinopses estendidas, enquanto João Pedro G de Souza e Matheus Avellar de Barros fizeram a predição de gênero de filmes a partir das suas legendas, no Capítulo 7. Trabalho semelhante foi feito por Rodrigo Pagliusi e Vitor Paes, no Capítulo 8, que apresentam a classificação de músicas em estilos musicais pelas suas letras.

Eduardo Vieira Marques Pereira do Valle e Herbert Salazar dos Santos, no Capítulo 9, inspirados pelo curso de Prospecção Tecnológica, dado no mesmo período pelo professor Jano de Souza, fazem uma Comparação Automática das Diretrizes Curriculares Nacionais (DCN) e a Classificação Brasileira de Ocupações (CBO).

Rodrigo Peregrino e Victor Hugo Ventura, no Capítulo 10 trataram da tarefa de sumarização de dados, buscando extrair o título de um artigo de seu resumo, usando transformadores (Vaswani et al., 2017).

Alguns alunos buscaram avaliar algoritmos. Felipe Bevilaqua Foldes Guimarães, que também usou transformadores, avaliou o desempenho de modelos de linguagem específicos para Tweeter no Capítulo 11. Já Glauco Primo e Fernando Costa de Souza, no Capítulo 12, avaliaram algoritmos de extração de palavras chave.

Espero que os trabalhos sirvam de inspiração para que esses e outros alunos escolherem suas pesquisas de dissertação e tese.

2

Análise de Semântica Latente (LSA) Aplicada a Projetos de Lei

Carlos Cardoso Dias & Tales Mello Paiva
Programa de Engenharia de Sistemas e Computação - COPPE

Abstract

This work aims to apply Latent Semantic Analysis in bills from the Brazilian House of Representatives to extract their topics and compare them based on date ranges and political parties. The results are presented in a tag cloud format. The analysis concluded that bills are following the country's historical phenomena and that the party with most bills is indeed following its guidelines. The tool developed in this work is publicly available to help citizens and researchers to conduct their own inquires.

Resumo

Este trabalho tem como objetivo utilizar a análise semântica latente em projetos de lei da Câmara dos Deputados no Brasil para extrair tópicos e compará-los com base em intervalos de datas e partidos políticos. Os resultados são apresentados em formato tag cloud. A análise concluiu que os projetos de lei estão acompanhando os fenômenos históricos do país e que o partido com mais projetos está de fato seguindo suas diretrizes. A ferramenta desenvolvida neste trabalho está disponível publicamente para auxiliar cidadãos e pesquisadores a realizarem suas próprias pesquisas.

2.1 Introdução

A Câmara dos Deputados do Brasil, por meio do projeto Dados Abertos, oferece acesso a uma coleção de documentos de cunho legislativo, tais como proposições, frentes parlamentares, deputados, discursos, despesas de parlamentares, dentre outros. Os dados disponíveis são alimentados por sistemas internos da câmara, sendo disponibilizados ao público de forma gratuita com o objetivo de que entidades da sociedade civil possam desenvolver aplicações alimentadas pelos dados disponibilizados (BRASIL, 2022a).

A Análise por Semântica Latente (LSA). é uma técnica de processamento de linguagem natural (NLP) que busca uma relação entre os termos de um documento e os conceitos, ou ideias, por detrás dos mesmos. Essa técnica se baseia no uso da decomposição em valores singulares da matriz termo-documento para redução de dimensionalidade desta matriz, desta forma, aglutinando palavras em conceitos (Xexéo, 2022).

Ao realizar a Análise por Semântica Latente sobre as coleções de documentos disponibilizados pela Câmara sobre projetos de lei, a motivação deste trabalho é capturar as principais ideias que circulam na esfera política por um período de tempo, ou um partido, permitindo a identificação dos principais conceitos e tópicos em pauta, evidenciando tendências e anomalias, e também simplificando o processo de acompanhamento legislativo pelo cidadão.

O trabalho é dividido da seguinte forma: a seção 2.2 traz os objetivos a serem alcançados; a seção 2.3 discorre sobre a metodologia de aquisição e tratamento dos dados; a seção 2.4 apresenta os resultados e os discute; ficando a conclusão do trabalho na seção 2.5.

2.2 Objetivos

Os objetivos desse trabalho foram divididos em um objetivo geral que é atingido por meio de quatro objetivos específicos.

- Objetivo Geral: Identificar e apresentar, numa forma de fácil compreensão, os principais conceitos e ideias associados aos projetos de lei propostos na Câmara dos Deputados do Brasil.
- Objetivos Específicos:
    1. Filtrar por período de tempo, de forma a identificar alterações nas temáticas mais evidentes ao longo dos anos.
    2. Filtrar por partido, de forma a identificar tendências e anomalias sob a legenda de um partido.
    3. Combinar os filtros de partido e período de tempo, de forma a identificar alterações e evoluções nos comportamentos dos partidos ao longo dos anos.

4. Apresentar os filtros de uma maneira fácil de manipulá-los, de forma a possibilitar a experimentação, visando favorecer a descoberta de tendências e anomalias.

2.3 Metodologia

Para compor a base de dados foram utilizados os dados de Proposições, Autores e Temas para todos os anos disponíveis (1934-2022) no site do projeto Dados Abertos da Câmara [1].

A base foi unificada a partir do conjunto de proposições utilizando *idProposicao* para a base de autores e *uriProposicao* para temas.

As proposições foram filtradas para conter apenas:

- Projetos de Lei (*siglaTipo* = "PL")
- propostos por Deputados (*tipoAutor* = "Deputado")

A base final apresenta 124.848 registros com *ano* e *ementa* devidamente preenchidos. Dentre esses, há siglas referentes a 64 partidos ao todo, apesar do Brasil contar hoje com apenas 32 partidos ativos registrados, de acordo com o Tribunal Superior Eleitoral (BRASIL, 2022b). Há registros pertencentes a 32 partidos que foram extintos, incorporados a outro, ou cuja sigla mudou devido a alguma mudança de nome ao longo da história.

Ademais, um total de 11.933 registros, o equivalente a 9,56% da base, não possui registro da sigla do partido do autor da proposição, estando o campo nulo ou com as especificações "PNI" ("Partido Não-Identificado") e "SPART"/"S.PART." ("Sem Partido"). Esses registros foram agrupados sob a sigla "NI" ("Não Identificado"), de forma a facilitar a análise. Algumas outras correções foram realizadas a respeito de nomenclatura das siglas, visando unificar grafias distintas de um mesmo partido. Por fim, as proposições sob a sigla "PRP" foram separadas em 2 partidos: as proposições até 1965 são do "Partido de Representação Popular"e, portanto, foram renomeadas para "PRP*"; já referente às proposições de 1989 em diante, "PRP" é a sigla do "Partido Republicano Progressista", atual "PATRIOTA", e a sigla foi mantida como estava.

Quanto aos temas das proposições, a Câmara dos Deputados conta com 32 temas pré-definidos. A base conta com 50.560 registros cujo tema não se encontra declarado, havendo assim 74.288 registros com tema e ementa devidamente preenchidos, os quais apresentam possibilidades de análises interessantes de serem exploradas. Os registros com tema nulo não foram expurgados da base, uma vez que a ementa encontrava-se presente.

Para o processamento do texto, os campos *ementa*, *ementaDetalhada* e *keywords* foram concatenados, sendo removidos os registros vazios após a concatenação. Os documentos

[1] `https://dadosabertos.camara.leg.br/swagger/api.html#staticfile`

foram pré-processados para a remoção de acentos; transformação de letras para minúsculas; remoção de stopwords padrão (nltk) acrescido de stopwords relevantes encontradas na base (*'lei', 'federal', 'nacional', 'aplica', 'aplicando', 'aplicar', 'aplicado', 'aplicados', 'vigencia', 'programa', 'institui', 'publica', 'publico', 'sobre'*); remoção de palavras não compostas inteiramente por caracteres alfabéticos; e remoção de sufixos utilizando o stemmer RSLP (Orengo e Huyck, 2021).

Após o pré-processamento montou-se a matriz termo-documento da coleção, sendo utilizada a métrica TF-IDF (do inglês, "Term Frequency-Inverse Document Frequency") e em seguida foi feito o SVD (do inglês, "Singular Value Decomposition") desta matriz para obter os termos semânticos ocultos, utilizando o valor de 32 componentes, ou tópicos, para a matriz diagonal de valores singulares, de forma a corresponder à mesma quantidade de temas pré-definidos pela Câmara.

A decomposição em valores singulares da matriz termo-documento, $\boldsymbol{A}$, resulta em três matrizes, onde

$$\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^T \tag{2.1}$$

Sendo $\boldsymbol{A}$ uma matriz com m termos e n documentos, e t o número de tópicos escolhido para a decomposição SVD de $\boldsymbol{A}$, temos que

- $\boldsymbol{U}$ é uma matriz $m \times t$, que pondera os termos em relação aos tópicos
- Σ é uma matriz diagonal $t \times t$ que indica a relevância do i-ésimo tópico em relação à composição da base
- $\boldsymbol{V}^T$ é uma matriz $t \times n$ que pondera os documentos em relação aos tópicos encontrados

Desta forma, um tópico, quando visto sob a ótica de álgebra linear, são os vetores que compõem a base de $\boldsymbol{U}$. Estes vetores atribuem um valor numérico para cada um dos m termos presentes no vocabulário da base. Para a caracterização de um tópico foi tomado um valor de *threshold* e um limitante máximo de 5 termos de forma que para cada vetor da base de $\boldsymbol{U}$, o vetor foi ordenado pelo valor absoluto de suas componentes e foram selecionados os k termos maiores em módulo que o *threshold*, com $k \leq 5$. A relevância, obtida da matriz Σ, para o conjunto de 32 tópicos, obtidos da matriz $\boldsymbol{U}$ conforme descrito, considerando toda a base de dados pode ser vista na figura 2.1, onde os caracteres '+' e '-' no início de cada palavra que compõe o tópico está associado, respectivamente, ao valor positivo e negativo do termo na base de $\boldsymbol{U}$.

Para facilitar a visualização dos termos que compõem um tópico e sua importância, tanto na pertinência ao tópico quanto à caracterização da base, foi utilizada a biblioteca *wordcloud*[2] para a exibição dos dados, tendo seu comportamento estendido para comportar a presença de palavras repetidas com frequências distintas (caso onde uma mesma palavra está presente em múltiplos tópicos). Desta forma, foi utilizado o padrão onde cada cor utilizada na *tag cloud* corresponde a um tópico e o tamanho do termo corresponde a sua influência com relação ao tópico ($\boldsymbol{U}[i]$) ou com relação à base ($\boldsymbol{U}[i]\Sigma[i]$). Para a análise considerando todos os documentos da base, a relevância de

[2] `https://pypi.org/project/wordcloud/`

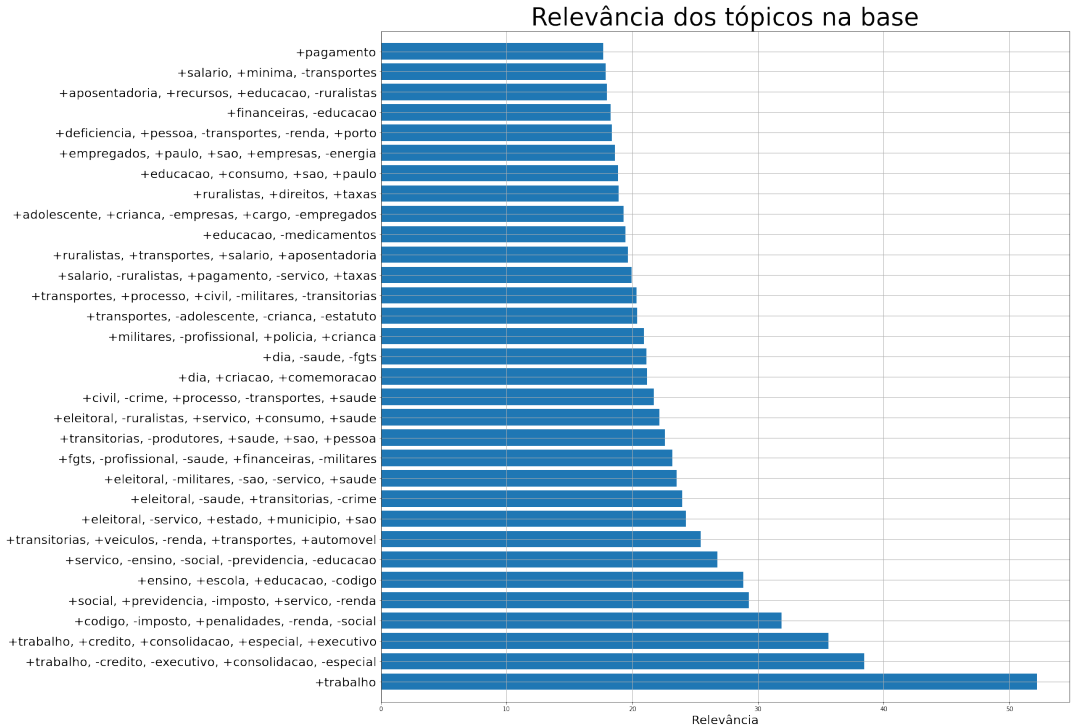


Figura 2.1: Tópicos x Relevância.

cada termo com relação ao seu tópico e de cada termo com relação à base podem ser vistas, respectivamente, nas figuras 2.2 e 2.3.

Figura 2.2: Relevância termo-tópico.

Figura 2.3: Relevância termo-base.

2.4 Resultados

Como resultados primários, obtidos de forma a guiar as demais análises posteriores, temos que 6 temas correspondem a mais da metade das proposições com tema declarado, dentre os 32 temas possíveis:

1. Administração Pública - 12,35%
2. Trabalho e Emprego - 11,83%
3. Direitos Humanos e Minorias - 7,52%
4. Direito Penal e Processual - 6,32%
5. Educação - 6,12%
6. Finanças Públicas e Orçamento - 6,03%

Podemos observar também que apenas 8 partidos, dentre os 64 presentes na base, são responsáveis por mais da metade das proposições:

1. PT - 13,19%
2. PMDB - 9,01%
3. PSDB - 5,91%
4. PSD - 5,43%
5. PTB - 4,80%
6. PDT - 4,21%
7. MDB* - 3,40%
8. PFL - 3,01%

Tal concentração é ainda mais latente quando considera-se que o partido cuja sigla é apresentada como ”MDB*”é, na verdade, o próprio ”PMDB”durante os anos entre 1966 e 1979, o qual em 2017 tornou a denominar-se MDB, constando na base como ”MDB”, de forma a diferenciar os 3 momentos do mesmo partido. Assim, temos que, efetivamente apenas 7 partidos respondem por mais da metade das proposições da base.

Para a verificação da relevância dos Projetos de Lei quando comparados ao cotidiano, foram analisados os períodos de 2000-2019 (figura 2.4) e 2000-2022 (figura 2.5), com especial foco nas mudanças entre as bases para este período.

Na figura 2.5, pode-se notar o surgimento da palavra “pandemia” dentre os termos relevantes para a classificação em mais de um tópico, além do aumento da relevância de “saúde” com relação a base da figura 2.4.

Além disso, quando comparados os tópicos gerados para os dois períodos, os termos “violência” e “mulher” apareciam nos tópicos “violência, mulher, doméstica, rural” e “idoso, violência, mulher” para o período de 2000-2019, o primeiro sugerindo um contexto específico regional de violência doméstica e o segundo sugerindo um contexto genérico aplicado ao grupo. Já para o período de 2000-2022, os mesmos termos aparecem nos tópicos “violência, profissional, mulher”, “violência, mulher, doméstica”, “fgts, violência, mulher”, sugerindo a aparecimento de medidas específicas para o contexto de violência doméstica e também da profissional mulher para o segundo período. Quando pesquisado

Figura 2.4: Relevância termo-base para o período de 2000-2019.

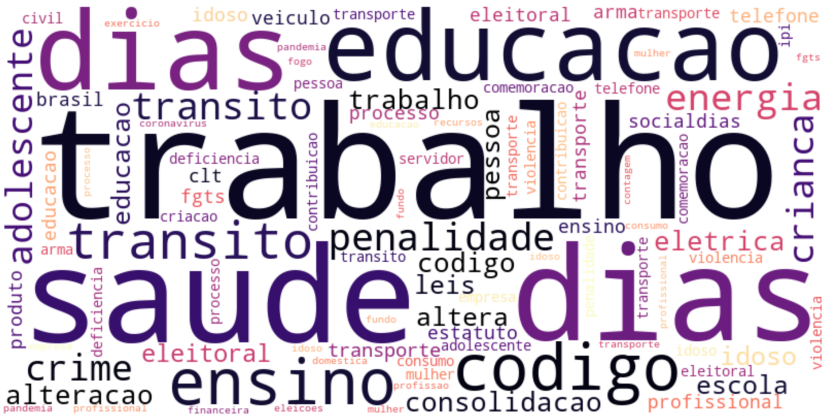


Figura 2.5: Relevância termo-base para o período de 2000-2022.

em sites de notícias[3], verifica-se que há um aumento dos casos de violência contra a mulher durante o período da pandemia.

Quando analisado especificamente o período de 2020-2022, com a escolha de categorização em apenas 8 tópicos, podemos observar se o impacto da pandemia de coronavírus no Brasil teve relevância do ponto de vista legislativo na esfera nacional, como mostra a figura 2.6.

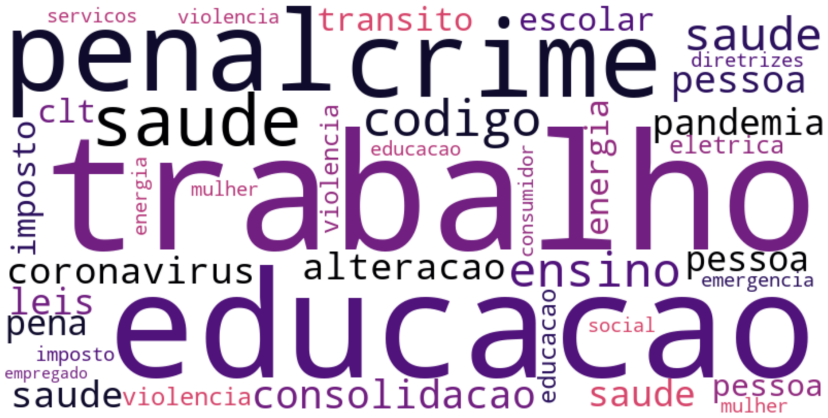


Figura 2.6: Relevância termo-base para o período de 2020-2022.

Por fim, também foi selecionado o partido com maior número de proposições de Projetos de Lei da base ("PT") e feita a análise para a extração de 16 tópicos baseados nestes projetos. Foi analisado os tópicos e termos relevantes para formação do tópico, de acordo com a figura 2.7.

Figura 2.7: Relevância termo-tópico para o PT.

Segundo o site oficial do partido, "o PT surgiu como agente promotor de mudanças na vida de trabalhadores da cidade e do campo, militantes de esquerda, intelectuais e

[3]https://agenciabrasil.ebc.com.br/saude/noticia/2021-08/violencia-contra-mulheres-cresce-em-20-das-ci
staticfile

artistas" Partido dos Trabalhadores, 2022. Pode-se observar que "trabalhador", palavra de destaque na descrição do partido, é também uma palavra com peso relevante e que figura em mais de um tópico. Além disso, para a caracterização de tópicos também figuram palavras como: benefício, social, criança, adolescente, rural, ensino, dentre outras. O gráfico na figura 2.8 mostra os tópicos definidos e sua relevância com relação à base composta por projetos de lei do PT.

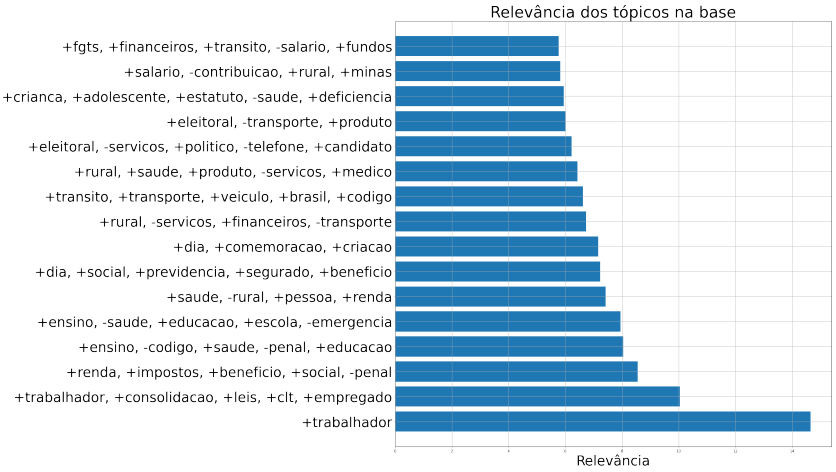


Figura 2.8: Tópico x Relevância para o PT.

2.5 Conclusão

Como visto na seção 2.4, é possível estabelecer uma relação direta de acontecimentos relevantes do cotidiano com a alteração de temáticas definidas pela análise de semântica latente dos projetos de lei. Também é possível, através da delimitação da base de dados pelos filtros criados, avaliar o trabalho de partidos políticos de acordo com a premissa destes partidos através da criação de tópicos da base. Por fim, a representação dos tópicos através de uma *tag cloud* permite a rápida compreensão e visualização das temáticas do governo, facilitando o compartilhamento da informação.

Além disso, todo o código desenvolvido neste trabalho está disponibilizado em [4] como uma ferramenta que conta com campos de input disponíveis da plataforma que permitem a customização das análises por período, partido, número de tópicos e demais opções do algoritmo de acordo com a necessidade do usuário.

Como sugestão de trabalhos futuros, visualiza-se a criação de uma interface mais amigável ao uso, e a disponibilização dessa ferramenta em um ambiente público e de fácil acesso. Além disso, pode-se enriquecer a escolha das siglas a partir de um mapeamento

[4] `https://colab.research.google.com/drive/1pKLW2xDtWffogS3lopReLua3XMLZuQcF?usp=sharing`

da evolução dos partidos, a partir das suas fusões, incorporações e mudanças de nomenclatura. Na mesma linha, seria interessante uma forma de visualizar o efeito que essas fusões e incorporações de partidos tem nas novas legendas resultantes, de forma a visualizar a interferência ideológica antes e depois de tais eventos. Além disso, a análise aplicada neste trabalho pode ser estendida a outros documentos da mesma base, como a descrição detalhada dos projetos submetidos ou até mesmo o conteúdo dos discursos políticos, o que abriria espaço para novas análises de congruência.

3

Most Relevant Rooms in an Airbnb Accommodation: A Quantitative Analyses on Reviews

Bruno D. de Paiva e João P. Leite Pinho
Engenharia de Computação e Informação - Escola Politécnica

Abstract

This paper describes a project that gets Airbnb reviews and clusters them to get the rooms more commented on, positively or negatively. The idea is that the person who announces its houses can focus on improving those rooms, aiming for better reviews on the platform.

Resumo

Este artigo descreve um projeto que recebe avaliações do Airbnb e as agrupa para obter os cômodos mais comentados, positiva ou negativamente. A ideia é que a pessoa que anuncia suas casas possa focar na melhoria dessas salas visando as melhores avaliações da plataforma.

3.1 Introduction

Airbnb is an application that works as if it were a social network that aims to unite people who want to share their accommodation with those who want some accommodation to stay, uniting hosts and travelers who aim to book unique accommodations anywhere in the world.(Airbnb, 2022)

Bearing in mind the importance of the platform in serving people anywhere in the world, always offering a place to rent, having a lodging evaluation policy makes it necessary for them to be able to measure the quality of the environment.

In addition, because the platform does not specify some information, it is necessary to understand what efforts the hosts must have to please their customers in the best possible way. With that in mind, we chose to build a solution that facilitates the planning of the person who is offering accommodation by discovering which room in the house will bring a better evaluation from the traveler, making the facility better evaluated and, consequently, increasing your search.

For this, we use a dataset containing about 9 million reviews to classify them into good or bad reviews and then extract which rooms are included in these reviews.

Research has already been done to analyze Airbnb reviews qualitatively("Me senti em casa: análise das revisões de experiências de hospedagem colaborativa no site Airbnb sob o prisma da confiança" 2021). However, our work intends to extract information from reviews through a quantitative analysis based on tools available in python language libraries.

3.2 Methodology

Considering the size of the dataset, we decided to reduce it to test the entire procedure, which will be better addressed in the following subsections.

3.2.1 Understanding the dataset

The dataset used, as previously mentioned, contains 9 million reviews about Airbnb's accommodations worldwide, written in more than ten languages.

First, we chose only English language reviews for this work. Moreover, we decided not to translate other languages to English since this could make sentiment analysis more challenging.

In addition, some reviews had only numerals as information. Since they would not add any value, we removed them from the dataset.

This dataset has six attributes: the hosting id, the date of the review, who made the review, and the actual review, which is, for this project, the most important data, and the only one used.

It is essential to point out that the same review can have multiple sentences, each with its own positive and negative sentiment. Thus, we chose to separate each review into its constituent sentences to have a better sentiment analysis of their content.

3.2.2 Data Treatment

Having a large amount of data, we chose to normalize all reviews made by users. Bearing in mind the previous point, that a user in the same accommodation can make more than one review.

Thus, we use the simple_preprocess tool from the gensim library, which aims to convert a document into a list of tokens, having passed as parameters the minimum size of the tokens that will remain in this list. In our case, we kept terms greater than or equal to three.

Furthermore, having this list of tokens, we removed stop words from the reviews in order to bring only relevant information for future analysis. We then chose to lemmatize to unify the tokens in their meanings.

In addition, we found and removed cases in which the reviews were empty because the users themselves made a mistake.

Additionally, we used a Porter stemmer to reduce all modified words (inflections, derivations) to their root form, aiming to improve the final analysis.

In a complementary way, after all the previous treatment processes, we created a process for filtering all the rooms contained in a specific review using the stemmed reviews.

For this point, a list of rooms created by the authors in the chosen language was used, in this case, English, to encompass as many accommodations as possible and not lose information. In addition, we used the Wordnet library's synset function to obtain all potential synonyms for each of the words listed, aiming to cover even more the list of synonyms.

At the end of this process, we obtained a dataframe in which we had the information from the reviews (one sentence per line), their text treated, lemmatized, and stemmed, and a column containing the list of rooms mentioned in this review (Paice, 1990).

3.2.3 Sentiment Analyses

This topic is a crucial topic for the analysis, given that the dataset does not have any explicit feeling about the reviews and, in order to bring greater business value, it is necessary for us to be able to divide which rooms are most reviewed on positively and negatively ways. In this way, as it is not the final objective of the project, we use the textblob library that is capable of measuring the polarity of the message, as well as its subjectivity (Ahuja and Dubey, 2017). In this way, it is possible to map the sentiment of this review.

For this, we defined the polarity levels in 7 possible feelings. Knowing that this polarity varies from -1 to 1, and that 0 is considered neutral sentiment, we made the following mapping.

Table 3.1: sentiment mapping table

Sentiment	Interval
horrible	[-1, -0.75[
very negative	[-0.75, -0.5[
negative	[-0.5, 0[
neutral	0
positive	[0.01, 0.5[
very positive	[0.5, 0.75[
amazing	[0.75, 1]

Given these results, we verified whether the value obtained was within a defined range for each polarity measured in a sentence. We used this information to create a new column in the dataset, which indicates the sentiment of this sentence.

Furthermore, given that we only want to check the negative and positive feelings, we chose to remove all reviews classified as neutral, even considering that the quality of this sentiment analysis does not have perfect accuracy or precision.

3.2.4 Word Cloud

Before creating the tag cloud, we measured the frequency of each room that appeared in the room column. We created a dictionary with the accommodations as key and frequencies as value. This way, the tag cloud could be generated from the previously calculated frequencies.

Since this work aims to help those who offer their accommodation on Airbnb, it is necessary to expose the information extracted from the reviews in an easy-to-understand format. For this, we made use of the wordcloud librarythat builds a graphic scheme where the size of the words corresponds to the frequency they appear in the text, highlighting the most relevant words (Hearst et al., 2020). We built two tag clouds, one with positive reviews, Figure 3.1, and one with negative reviews, Figure 3.2, as you can see below.

Figure 3.1: Types of rooms that appear the most in positive reviews.

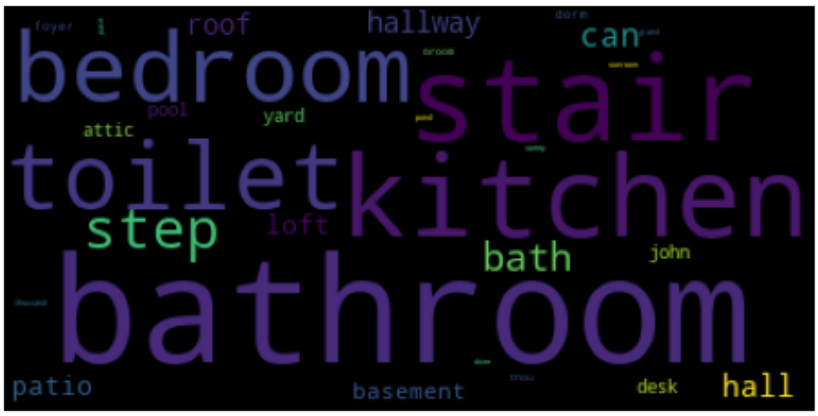


Figure 3.2: Types of rooms that appear the most in negative reviews.

3.3 Results

When analyzing both generated wordclouds, it is possible to see that there are easily observed highlights, indicating which accommodations the platform's hosts should focus their efforts on. Imagetically, from the result of the wordcloud, it is possible to notice that "kitchen", "bedroom", "stair", "bathroom" and its synonym "toilet" are the most evident in both wordclouds. In both positive and negative reviews, the most frequent word is "bathroom", followed by "kitchen". Furthermore, it is clear that, due to the use of wordnet, words like "Jhon", which is a synonym for bathroom in British English and "can" as an informal synonym for bathroom, in North American English appear as possible words for this wordcloud, negatively impacting the achievement of the result. One possibility was basically to combine all these possible synonyms in a single word, in order to obtain a better result in which the room was most reviewed, either negatively or positively, but such efforts were left for future implementations of this project.

Another point observed is that both wordclouds have very similar rooms, the image visualization of the wordcloud does not make it easier to obtain this information, however, as can be seen in the tables below, there is the following relationship between room and frequency of appearance. It is also important to note that for a number of 50000 reviews, after all the filters, those that list rooms and are positive or negative reviews are about 10000, in which 20% are negative reviews and the other 80% are positive.

Table 3.2: Table of Words vs Frequencies (Positive reviews)

Words	Frequencies
bathroom	2066
kitchen	1998
bedroom	1104
stair	537
step	384

Table 3.3: Table of Words vs Frequencies (Negative reviews)

Words	Frequencies
bathroom	730
kitchen	256
stair	242
bedroom	240
step	110

Thus, by selecting the 5 Words most reviewed positively and negatively, it is possible to better indicate to the owner of the establishment what should be the greatest care that he must have in relation to his environment in order to receive a better evaluation within the platform and also to bring greater customer satisfaction.

3.4 Conclusion

Observing the results obtained, it is possible to conclude that, in fact, the bathroom, kitchen, bedroom and stairs are the rooms with which an Airbnb host should pay more attention. It is recommended, above all, to stick to any problem that there may be in the bathroom, as this appears a much higher number of times than the other rooms in the negative reviews. It is important to notice that this study was carried out in a Dataset that contained reviews from airbnbs located in the cities of Los Angeles and New York, in the United States. Therefore, the results obtained must be considered within this context. Furthermore, as only English reviews were analyzed, the analysis is restricted to English speaking guests.

4

Análise da evolução dos títulos de pesquisas em computação da UFRJ por meio de *Differencial Tag Clouds*

Pedro Boechat e Pedro Kuchpil
Engenharia de Computação e Informação - Escola Politécnica

Resumo

Este trabalho aplica Differential Tag Clouds Airbnb, 2022 para a visualização dos títulos de publicações relacionadas à Computação realizadas por professores e pesquisadores da Universidade Federal do Rio de Janeiro ao longo dos últimos 20 anos. Desta forma, é possível estabelecer termos e temas que se mantiveram como protagonistas durante este período, e também identificar novas e velhas tendências na pesquisa dentro da área.

Abstract

This work applies Differential Tag Clouds (Xexéo, F. Morgado, and Fiuza, 2009) for the visualization of publication titles of teachers and researchers from the Federal University of Rio de Janeiro in computing-related areas on the last 20 years. In this way, it will be possible to establish terms and themes that maintened themselves as protagonits during this period, and also identify old and new tendencies of research inside this area.

4.1 Introdução

A Universidade Federal do Rio de Janeiro (UFRJ), ao longo de seus 101 anos, se consolidou como uma das melhores universidades da América Latina (PESC, 2022), em parte por conta da grande produção acadêmica de seus pesquisadores. Programas como o Programa de Engenharia de Sistemas e Computação (PESC), entre tantos outros da UFRJ, rotineiramente são classificado com a nota máxima da CAPES(THE - Times Higher Education, 2022), provando a competência e a importância da universidade para a pesquisa brasileira.

Com o intuito de analisar a evolução da pesquisa na UFRJ ao longo dos anos, este trabalho se utilizará de dados do Currículo Lattes de pesquisadores da universidade para responder às seguintes perguntas:

1. Quais foram os tópicos mais pesquisados dentro das áreas de computação?
2. De que forma ocorreu a mudança desses tópicos ao longo dos anos?

4.2 Dados

Os dados utilizados foram extraídos do Currículo Lattes de professores da universidade. A base de dados possui 70.740 publicações, contendo DOI, título, áreas de conhecimento, idioma e ano da publicação, e foi disponibilizada pelo Conecta UFRJ (UFRJ, 2022).

Após análise inicial, foi constatado que pouco menos de 5% dos dados não possuia título ou ano da publicação e 18% não possuia nenhuma área de conhecimento. Assim, começamos a manipulação dos dados tratando essas inconsistências, além de removendo quebras de linha, *tags* HTML e entradas nulas dos títulos. Ao analisar o idioma dos artigos, encontramos que cerca de 37% não possuíam o idioma e mais de 60% eram escritos na língua inglesa.

Considerando as áreas de conhecimento que possuíam mais publicações, vimos que grande parte delas eram áreas dos campos de medicina, biologia e química.

A fim de esclarecer a mudança do comportamento dos títulos de publicações ao longo dos anos, este trabalho se debruçará apenas sobre registros de áreas do conhecimento ligadas à Computação, que são 2131 na base, para que a variação dos títulos de publicações de um período específico sejam melhor percebidas quando comparadas à coleção geral. Além disso, só serão analisados artigos escritos em língua inglesa, que são maioria, para que o pré-processamento também possa ser o mesmo para todos os dados. Logo, geramos o mesmo gráfico para essas áreas.

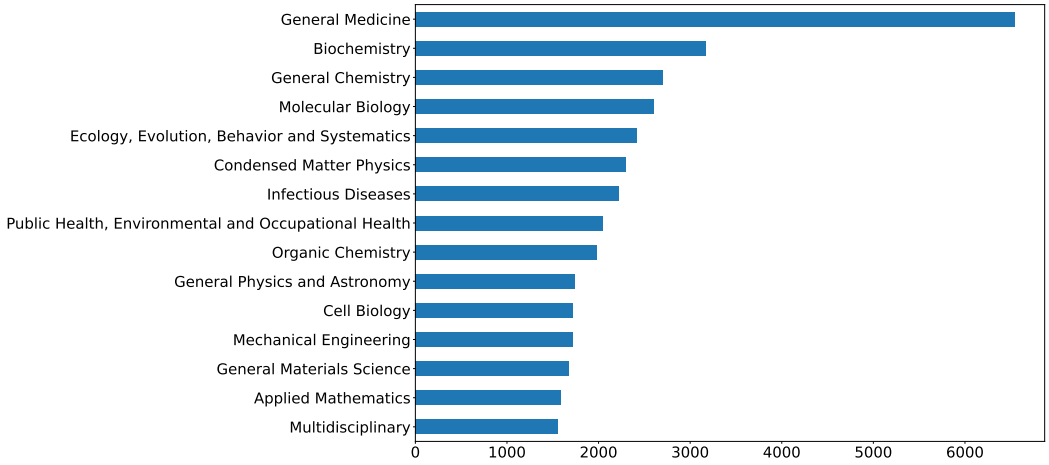


Figura 4.1: 15 áreas de conhecimento com mais publicações.

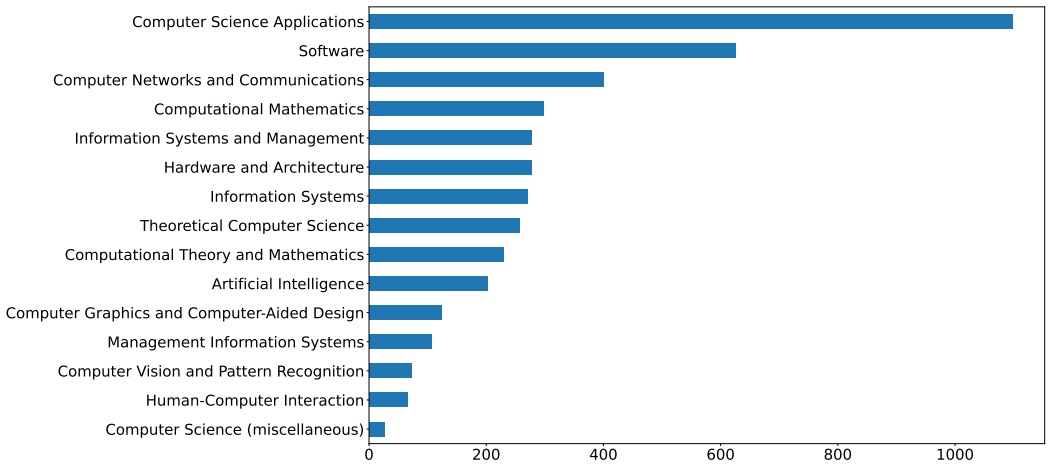


Figura 4.2: Áreas de conhecimento relacionadas à computação.

Também analisamos o número de publicações por ano e foi possível notar que as publicações de áreas de conhecimento relacionadas à computação cresceram de forma similar ao número total de publicações.

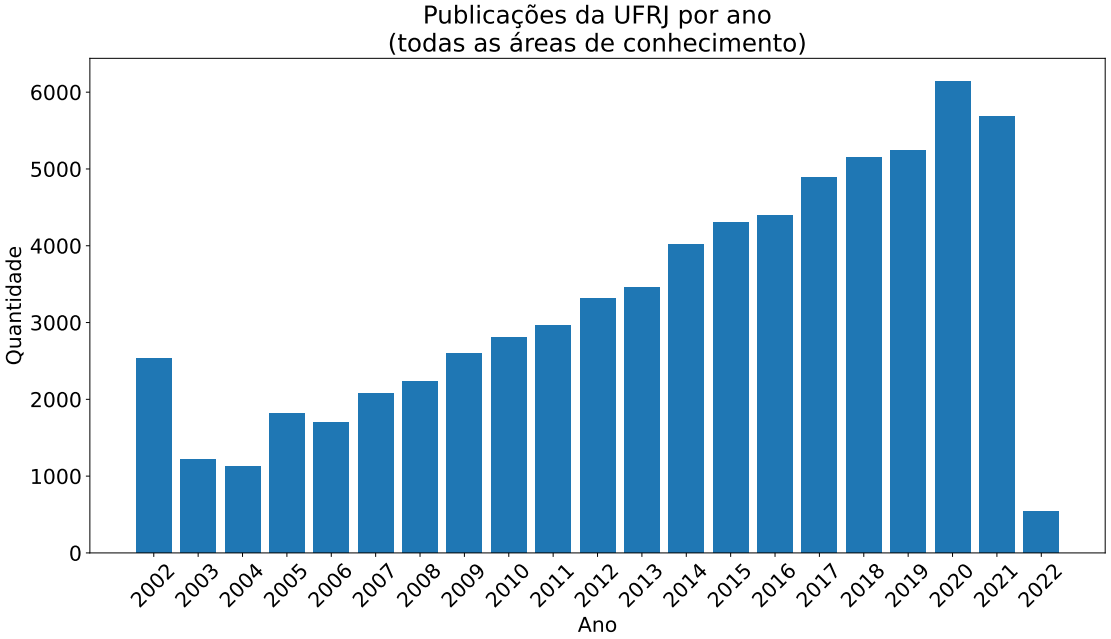


Figura 4.3: Artigos publicados por ano em todas as áreas de conhecimento.

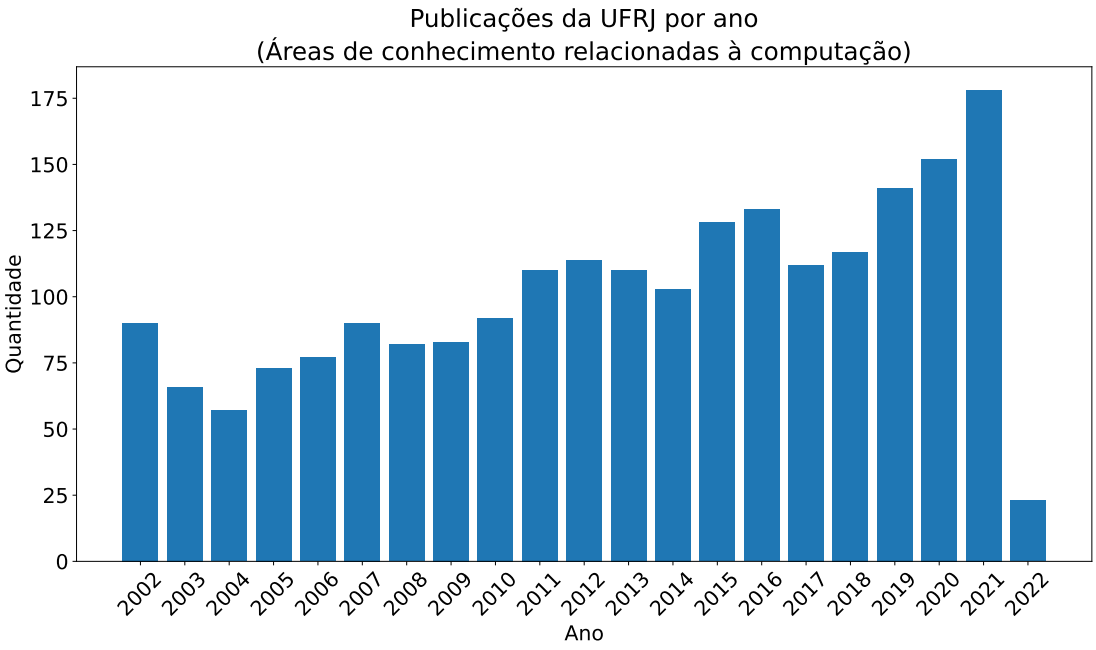


Figura 4.4: Artigos publicados por ano em áreas de conhecimento relacionadas à computação.

4.3 Metodologia

O trabalho foi desenvolvido em *Python*, carregando os dados disponibilizados como *JSON* para um dataframe do Pandas(McKinney, 2010).

Nos registros trabalhados, muitos dados do idioma da publicação estavam como nulos, ou então ainda erroneamente classificados. De modo a corrigir essa informação, foi utilizada a biblioteca `fasttext` (Joulin et al., 2016), que, com um auxílio de um modelo pré-treinado, é capaz de fazer essa classificação. Apenas registros cujo resumo havia probabilidade maior do que 90% de ser em inglês, de acordo com o modelo, foram considerados na análise.

Após a remoção dos registros que não seguiam as regras estabelecidas, foi realizado o processamento do campo de título dos artigos, com a remoção de *stopwords*, para que fossem considerados apenas termos relevantes à publicação. O *Stemmer* de *Porter* M. Porter, 2022 também foi aplicado, a fim de normalizar termos.

Continuando os passos descritos por Xexéo, F. Morgado e Fiuza, 2009, é calculado o *TF-IDF* dos termos, com base no ano de sua publicação. Para isso, foi criado uma coleção de documentos, um para cada intervalo de 3 anos (a partir de 2002) e contendo os títulos já processados de todas as publicações relevantes para o estudo datadas do respectivo intervalo.

Assim, é possível calcular a *Summary Tag Cloud*, que seleciona os 40 termos mais bem classificados na coleção de documentos. Ela está presente na Figura 4.5.

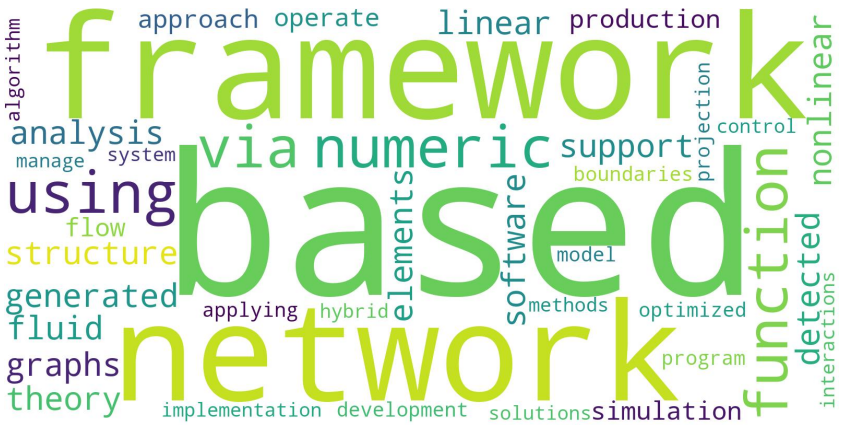


Figura 4.5: *Summary Tag Cloud* da coleção de títulos de artigos

Após essa etapa, a *Differential Tag Cloud* pode ser calculada para os intervalo de anos, considerando as palavras mais bem colocadas que não estão na nuvem de *summary*. Desta forma, podemos identificar os temas e as tendências que marcaram um período específico. As nuvens geradas estão da Figura 4.6 até a Figura 4.12.

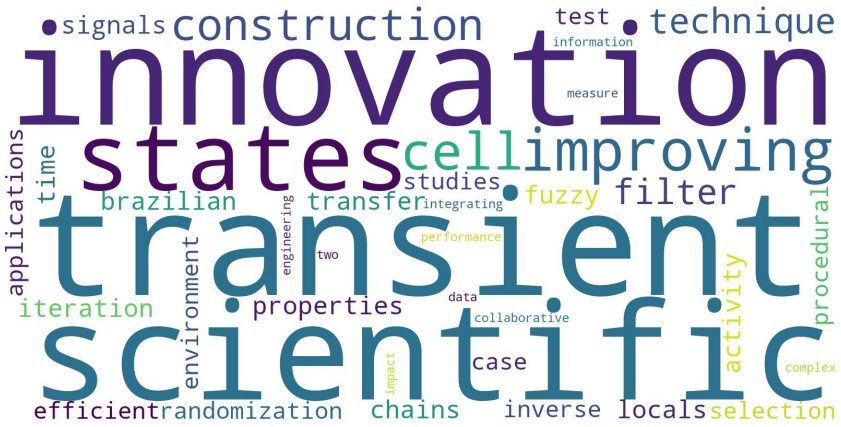


Figura 4.6: *Differencial Tag Cloud* de 2002 a 2004

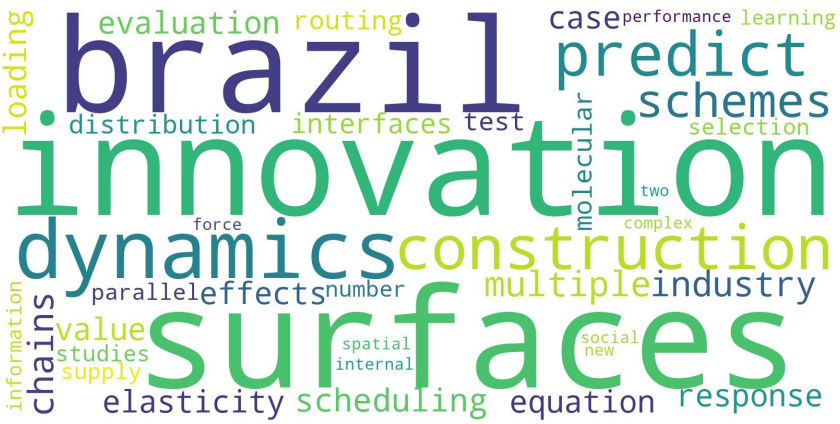


Figura 4.7: *Differencial Tag Cloud* de 2005 a 2007

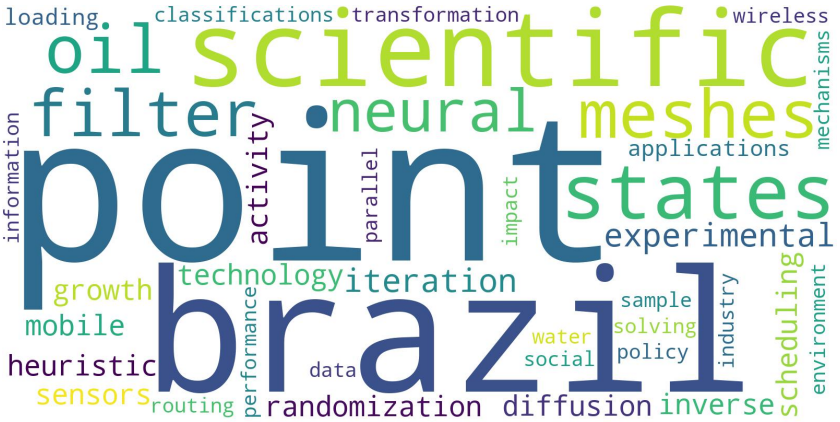


Figura 4.8: *Differencial Tag Cloud* de 2008 a 2010

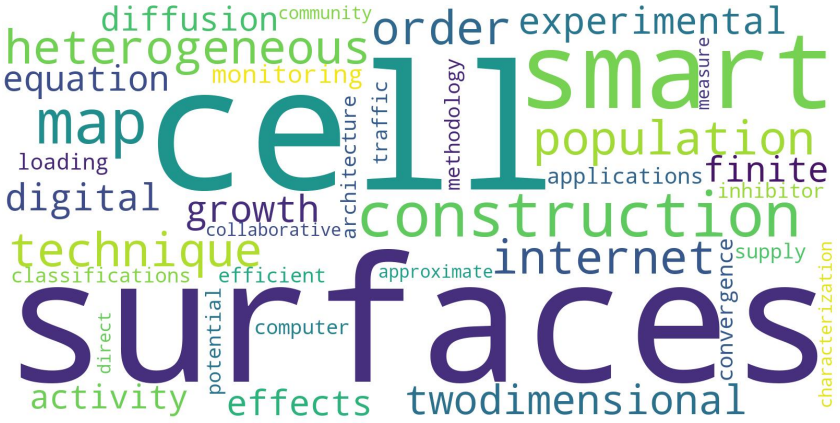


Figura 4.9: *Differencial Tag Cloud* de 2011 a 2013

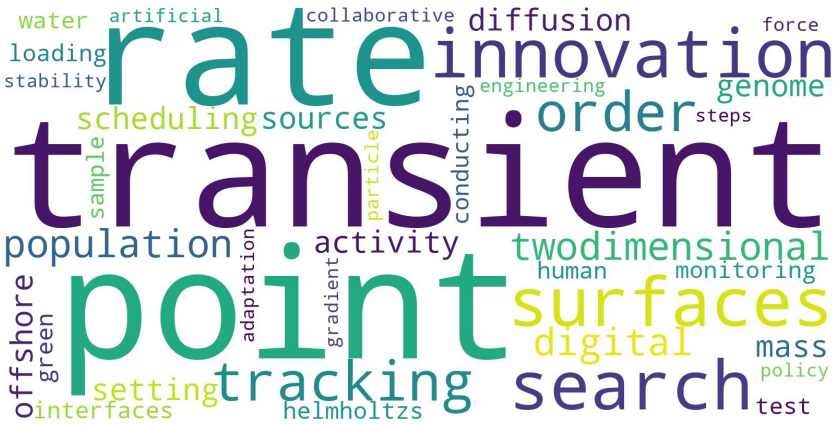


Figura 4.10: *Differencial Tag Cloud* de 2014 a 2016

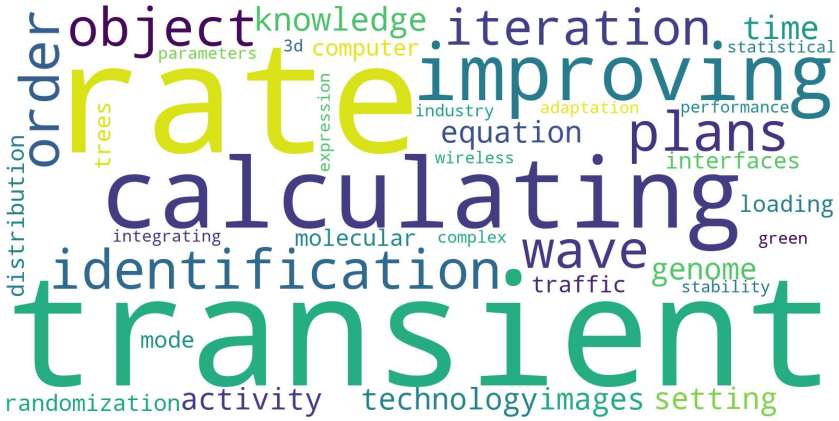


Figura 4.11: *Differencial Tag Cloud* de 2017 a 2019

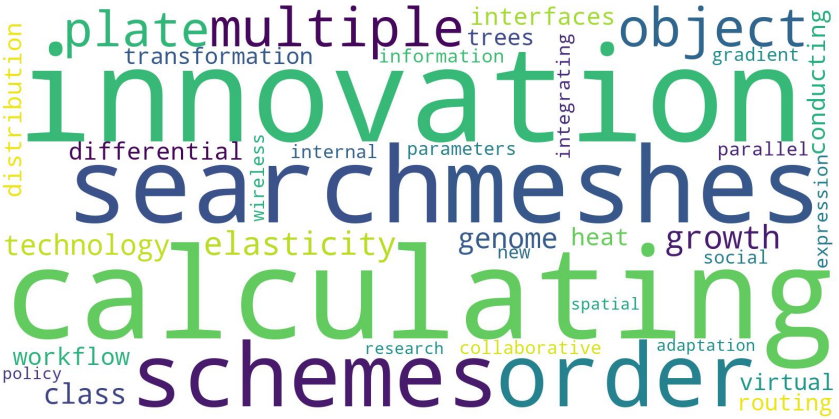


Figura 4.12: *Differencial Tag Cloud* de 2020 a 2022

4.4 Resultados e Conclusões

Como é possível observar nas imagens, ocorreram algumas mudanças significativas nos principais temas pesquisados na área de computação dentro da UFRJ ao longo do período analisado. Inicialmente, eram mais comuns publicações sobre aplicações científicas e sinais. Já entre 2008 e 2010, são presentes temas consideravelmente importantes, como pesquisas relacionadas a petróleo (pouco tempo depois da descoberta do Pré-Sal) e a redes neurais. Atualmente, nos três últimos anos, vale destacar os estudos sobre políticas, temas sociais e paralelismo. Cada um a sua maneira, são temas muito relevantes para a computação.

A criação de *Differencial Tag Clouds* para títulos de publicações da UFRJ permite acompanhar a evolução da pesquisa pública de excelência, sobre que temas já foram abordados e quais tendências estão sendo seguidas (ou criadas) dentro da instituição. Este trabalho sobre a área de computação pode ser replicado para qualquer outra, ou então ainda para a universidade como um todo.

Como trabalho futuro, cabe pensar em um gerador destas nuvens, analisando os dados públicos disponibilizados no Lattes dos pesquisadores de todo o Brasil. Cabe considerar também que diversos termos foram encontrados repetidas vezes, o que poderia ser evitado aumentando o tamanho da *Summary Tag Cloud*, ou então impedindo que palavras sejam adicionadas se estiverem em mais de um determinado número de nuvens diferenciais.

5

GTAGLIB: a Library to Generate Tags and Tag Clouds

Guilherme C. Mattos
Programa de Engenharia de Sistemas e Computação - COPPE

Abstract

Inside and outside the digital world, tags are an important way to describe or categorize documents and sets of documents, allowing humans and automated systems to more quickly and easily identify and understand their nature and contents. Meanwhile, tag clouds, or word clouds, are a way to present tags in a graphical format in which special attention is given to the relevance that the tags have. That allows human users to identify the terms that better represent the text.

This work presents GTAGLIB, a library to generate tags and tags clouds for documents and sets of documents. That library implements two algorithms described in the literature that aim to produce tags and tag clouds to summarize and differentiate documents. Finally, this work also shows how the library performs on a small set of texts and briefly discusses the results.

5.1 Introduction

From physical archives to digital information repositories, tags have been widely used to index data and facilitate access to information. In Archival Sciences, for example, the NOBRADE (*Norma Brasileira de Descrição Arquivística*, or Brazilian Archival Description Standard) has a field dedicated to that purpose (Conarq, 2006). On the internet,

multiple social networks, like Twitter[1] and Instagram[2] heavily rely on the use of tags (or "hashtags") to provide information discovery capabilities to their users.

However, while social networks and many other types of systems usually rely on users to add keywords to their content, tags assigned by users might not be reliable (Koutrika et al., 2008) or the amount of content to be tagged might make that task unfeasible to humans. For those reasons, automatic tag generation, also known as *tag recommendation* in the field of Information Retrieval, is relevant to many systems and has been a target of research for decades. For example, in (Belém, J. M. Almeida, and Gonçalves, 2016), several tag recommendation methods are presented, as well as a taxonomy to classify them.

In addition to the tags, another critical aspect of text tagging is how to display related information for human use. On Twitter, as mentioned earlier, for example, trending tags are always present, as a list, on the top right area of the pages (on the desktop version of its website), and there is also a page dedicated to that same purpose. Blogs, however, frequently have tags listed at the end or the beginning of posts. However, there are other ways to present tags to users, and one such way is through tag clouds, that are visual representations in which tags have their visual attributes and placement decided based on relevance, where relevance can be determined by features like popularity (frequency) and recency (Bateman, Gutwin, and Nacenta, 2008; Halvey and Keane, 2007).

This paper presents GTAGLIB, a library to generate tags and tag clouds from documents, based on the methods of (Xexéo, F. Morgado, and Fiuza, 2009) and (F. F. Morgado, 2010). Its motivation is the utility of automatic tag generation systems and tag clouds. Section 2 covers the details of the methods implemented in the library, while Section 3 talks about the library itself and its implementation decisions. Section IV describes the evaluation approach taken, and Section V shows the results of that approach. Section VI concludes the paper, also mentioning possible future works.

5.2 Differential and Set Summary Tags and Tag Clouds

Introduced by (Xexéo, F. Morgado, and Fiuza, 2009) and (F. F. Morgado, 2010), set summary and differential tag clouds are visual representations of tags that, as their name imply, seek to summarize a set of documents and differentiate individual documents (with respect to the set summary tags), respectively. In both works, a formal model for tag cloud construction is given and, despite not being the focus of the present text, it is important to introduce the basic concepts brought by it quickly.

The first concept is the "semantic field", which comes from (Marinchev, 2006) and, in the words of (Xexéo, F. Morgado, and Fiuza, 2009), "is a set of abstract concepts that, somehow, can be applied to an object aiming to build some understanding of it" (an

[1]Twitter: https://twitter.com/.
[2]Instagram: https://www.instagram.com/.

object, in this case, can be a document). Directly tied to it, the second concept is the "tag field", which is a set of tags that represent those concepts. The third concept is the "abstract tag cloud", which is basically the combination of tags (from the tag field) with abstract features, that will be later translated to a visual representation (like text color and size), where those features may reflect the relevance of their respective tags.

Based on those fundamental definitions, the authors introduced the "abstract set summary tag cloud", and the "abstract differential tag cloud". The first assumes the existence of a semantic field where the object is the union of the semantic fields of all documents, and a summarization function that determines which tags (with their attributes) from that semantic field better describe the document set. Then, the tag cloud is built from those selected tags.

In the differential tag cloud, on the other hand, the focus is document individuality. For this reason, it assumes the existence of a differentiation function that will determine if a member of the abstract tag cloud is enough to differentiate the document from the entire set. That is done by not considering tags in the set summary tag cloud.

With those basic definitions presented, (F. F. Morgado, 2010) introduces two concrete methods, or algorithms, to generate set summary and differential tag clouds. The following subsections present those algorithms to clarify the basic behaviors that the GTAGLIB aims to implement.

5.2.1 Method 1

In this method, the documents are first preprocessed, tokenized, stemmed (using the Porter Stemmer (M. F. Porter, 1980)), and have their stop words removed. Additionally, after POS-tagging, only words identified as nouns are kept. After that, bigrams are generated, and the data is represented under a TF-IDF model(Salton, 1989), and the following steps are performed.

1. Build a semantic field for each document, choosing the 40 most relevant terms (based on TF-IDF) for a given document as tags.
2. Use the semantic field to create an abstract tag cloud for each of those documents, where the attributes should be based on term relevance.
3. Generate the differential tag cloud from the abstract tag cloud (in other words, both are the same).
4. Create a term-document frequency matrix.
5. Apply Latent Semantic Analysis (Landauer, Foltz, and Laham, 1998), decomposing and reconstructing the term-document frequency matrix. In this step, it is important to note that (F. F. Morgado, 2010) does not fully explain what is done to the decomposed matrices before reconstruction.
6. Apply the Pearson Correlation Coefficient on the reconstructed matrix to get a term correlation matrix.

7. Pick a root term (possibly chosen by the user) and build a list with the most positively correlated terms to that root until 40 are selected (additional bigrams related to the root can also be selected).
8. Build the set summary tag cloud from the terms from the previous item.

5.2.2 Method 2

In method 2, the same preprocessing steps from method 1 are applied, and the documents are also represented under a TF-IDF model. One of the main differences is the expansion of the semantic field with synonyms of the terms already there. The steps of the algorithm are as follows.

1. Build a semantic field for each document, choosing the 40 most relevant terms (based on TF-IDF) for a given document as tags.
2. Expand the semantic fields with related concepts. In this case, "related concepts" mean synonyms of terms in the semantic field, obtained from *synsets* from the WordNet (Miller, 1995).
3. Use the semantic field to create an abstract tag cloud for each document, where the attributes should be based on term relevance. In this part, (F. F. Morgado, 2010) does not cover what should be the relevance of the additional terms.
4. Sort, in descending order, the union of terms from all abstract tag clouds by the number of semantic fields where they appear.
5. Build the set summary tag cloud from the first 40 terms obtained from the previous step.
6. For each document, build the differential tag cloud from the terms in its abstract tag cloud that are not present in the set summary tag cloud. Color synonyms added from *synsets* with a different color.

In both methods, the stemming is reversed before creating a tag cloud, but no information is provided on how to deal with situations where different words produce the same stem.

Regarding the tag cloud presentation specifics, Xexéo, F. Morgado, and Fiuza (2009) and F. F. Morgado (2010) establish no definition, regardless of method. In any case, one can find experiments and discussions about that matter in (Bateman, Gutwin, and Nacenta, 2008).

5.3 GTAGLIB

The GTAGLIB (where the first "g" in its name stands for "generation") is a Python library (currently in version 0.1) that aims to provide the community with yet another tool to help with the task of tag and tag cloud generation. In its current form, it attempts

to implement the methods proposed by Xexéo, F. Morgado, and Fiuza (2009) and F. F. Morgado (2010), and is publicly available on GitHub[3].

When compared to what is specified in the original algorithms of the differential and set summary tag clouds, the GTAGLIB also seeks to provide more flexibility in parameter selection. Moreover, its internal preprocessing steps allow: the use of semantic fields of different sizes; two stemming techniques, Porter (M. F. Porter, 1980) and Snowball/Porter2 (M. Porter, n.d.); the Wordnet lemmatizer [4], as an alternative to using a stemmer; the ability to use only a term-document frequency matrix across the algorithms, and the possibility of turning off the generation of bigrams. By default, and considering that the library, in its current version, aims to support only the English language, it also converts all word characters to ASCII characters. However, given that it can be undesirable in some circumstances, like in cases where the text is multilingual, this can also be disabled.

The implementation of the algorithms also has slight differences compared to what was originally proposed. They are as follows.

5.3.1 Method 1

The implementation of method 1 is mostly similar to what is specified in (F. F. Morgado, 2010), possibly differing mainly in three aspects, where the first is the paper on Latent Semantic Analysis (LSA). Although the author claims to use LSA, he does not provide any precise information about what is done between the decomposition and the reconstruction of the term-document matrix. However, while describing that technique, the literature review section of the same work focuses on dimensionality reduction. For that reason, in GTAGLIB (version 0.1), it was decided to apply that procedure during the LSA step, maintaining only the n most important concepts in the reconstructed term-document matrix, where, by default, n is that size of the semantic field in the original algorithm (40). Additionally, considering that (Landauer, Foltz, and Laham, 1998) states that the number of dimensions to be kept for the LSA to be effective is an "empirical issue", the desired value of n can be passed to the library as a parameter.

The second possible difference is related to what is done if the term passed as root to generate the set summary tag cloud is not present in the model (if it is not a noun present in the set of documents). The description of the original algorithm does not cover that case, while GTAGLIB's implementation simply returns an empty list of tags or does not generate a tag cloud.

The last difference resides in the fact that, during the generation of the set summary tags, no additional bigrams are added after the 40 terms related to the root are selected.

[3] `https://github.com/GuilhermeCaeiro/gtaglib`

[4] `https://www.nltk.org/_modules/nltk/stem/wordnet.html`.

5.3.2 Method 2

In method 2, the algorithm provided by (F. F. Morgado, 2010) also has steps that are unclear, and, for that reason, GTAGLIB's implementation has some differences, one of them being the way the expansion of the semantic fields occurs. While the original algorithm provides no information on how many or what type of synonyms should be selected, the GTAGLIB allows users to select an upper limit on how many synonyms should be added per each word in the original semantic field. Additionally, the synonyms used are limited to *synset* names that are nouns. That is done to reduce the inclusion of unwanted words (like verbs). If the number of available synonyms is greater than an upper limit u stipulated, the algorithm returns u random synonyms from those available.

5.3.3 Tag Cloud Generation

The tag cloud creation relies on two aspects: word relevance and styling. The word relevance of the tags passed to the tag cloud depends on which algorithm is being run and what are the parameters being used. For abstract and differential tag clouds, the library uses TF-IDF scores or term-document frequencies concerning the document associated with the tag cloud. The deciding factor is if the use of TF-IDF is enabled or not. Regarding set summary tag clouds, the type of the score depends on the same factor, but the final score is the sum of the TF-IDF scores or term-document frequencies for a given word across all documents.

As for tag clouds created using method 2, the additional tags will always receive the score of the least relevant tag originally present in the semantic field. This is done to reduce the impact of any eventual "noise" added by the inclusion of synonyms, because, in many cases, they may have little to no real relationship with the original documents. In the current version of the library, synonyms in the differential tag cloud are not colored differently.

Finally, regarding styling, that is done through the use of the library WordCloud[5], which is a mature, resourceful Python library for that purpose. That library can also be used in two ways. The first is by having GTAGLIB create a default WordCloud object and using it to generate the tag clouds, while the other is by passing an already set up WordCloud object to GTAGLIB and having it being used instead. That approach was taken to give users nearly full control of the tag cloud styling.

[5] `https://github.com/amueller/word_cloud`

5.4 Evaluation Approach

In order to evaluate the library presented in this work, the dataset fao30[6], that was introduced by (Medelyan and Witten, 2008), was utilized. It is a small corpus composed of 30 documents related to agricultural themes that was tagged by a team of professional annotators. When considering all documents, after the removal of words with only one character, it has 129142 words and, on average, 4304.73 per document. The total number of tags is 997, having an average of 33.23 per document.

Before providing the documents to GTAGLIB, the only preprocessing steps taken were the removal punctuation, numbers, and words with a length of one character (as mentioned previously). Stopword removal and other possibly relevant steps were left to be treated by the internal preprocessing capabilities that the library has. The parameters used with both methods can be seen in Table 5.1 and, except for the option to convert characters to ASCII, they were chosen to match specifications present in (F. F. Morgado, 2010).

Table 5.1: Main Testing Parameters

Parameter	**Value**
Semantic field size	40
Stemmer	Porter
Generate bigrams	Yes
Use only nouns	Yes
Use TF-IDF	Yes
Characters to ASCII	Yes

As for the information that was sought to be evaluated, it was the abstract and the differential tags generated by both algorithms proposed by (F. F. Morgado, 2010). Abstract tags, despite not being a main product of the second algorithm, they can be retrieved in GTAGLIB's implementation and, for this reason, were considered for testing in both methods. Also for the particular case of the algorithm 2, it was tested with and without the expansion of the semantic fields with synonyms and, in the case where those additional words were included, the impact of the inclusion of 3, 5, 7 and 10 synonyms per tag in the semantic field of each document was verified.

The metrics employed in the tests were the precision, the recall and the F1 scores, that, according to (Wikimedia Foundation, 2022) are defined defined by Equations (5.1), (5.2) and (5.3), respectively. In those equations *tp* (true positives) is the number of tags produced that are present among the tags that are expected, *fp* (false positives) is the number of tags produced that are not present among the tags that are expected and *fn*

[6]Dataset fao30: https://github.com/LIAAD/KeywordExtractor-Datasets.

(false negatives) is the number of tags that are expected, but were not generated. As for their meaning, (Wikimedia Foundation, 2022) defines precision as "the fraction of relevant instances among the retrieved instances" (where "relevant instances" are the expected tags) and recall as "the fraction of relevant instances that were retrieved". The F1 score, on the other hand, is a metric that tries to combine both metrics in a balanced way (for example, to achieve a high F1 score, precision and recall must also be high).

$$precision = \frac{tp}{tp + fp} \tag{5.1}$$

$$recall = \frac{tp}{tp + fn} \tag{5.2}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{5.3}$$

When it comes to the time taken to generate the tags from the documents, it was also measured, making it important to mention the testing environment: a PC with an Intel Core i7-4720HQ CPU, 16GB of DDR3 1600 RAM and a SSD Crucial BX500 of 480GB. The operating system was Windows 10 Home 21H1, but the tests were run on Windows Subsystem for Linux (WSL 1), that ran Ubuntu 18.04.4 LTS.

Lastly, regarding tag cloud generation, given the fact that, in this work, it was relegated to another library, it is something difficult to properly asses. For this reason, this text will only comment a few examples of what the GTAGLIB (alongside the WordCloud library) is capable of generating. After that, at the end of the results section, a known issue found with GTAGLIB's implementation will also be briefly commented.

5.5 Results

Following the approach established previously, in this section, the first results to be presented are for method 1. In that method, abstract and differential tags are the same and achieve, across the 30 documents used for testing, an average precision of 0.095, a recall of 0.119 and a F1 score of 0.104, meaning that, out of the expected tags, only a very small number are being successfully generated by GTAGLIB.

When it comes to method 2, the results for the abstract tags are displayed in Figure 5.1. In that figure, it is possible to observe that, with the addition of synonyms to the semantic fields, average precision and, consequently, average F1 score fall. Recall, on the other hand, shows a marginal gain with the addition of up to 3 synonyms per tag, and stays the same with further additions. That difference in behavior when compared to precision is caused by the fact that precision is negatively impacted by the additional, frequently irrelevant (noise), synonyms, while recall only takes into consideration what was generated correctly.

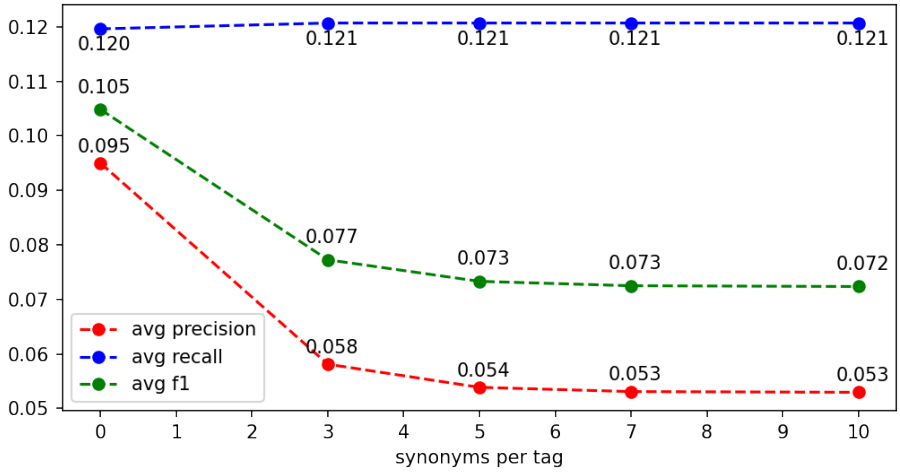


Figure 5.1: Results for abstract tags using method 2.

As for the differential tags, Figure 5.2 shows a slightly different scenario, with recall starting at a lower value, growing with the addition of up to 5 synonyms per tag, and then falling with more additions. That's caused by the removal of tags that are among the set summary tags. In other words, with zero synonyms, some relevant, expected tags end up as set summary tags and are not considered as differential tags, impacting all metrics negatively. However, with the inclusion of synonyms, some of these additional words might end up among the set summary tags as well, while some relevant, expected abstract tags go to the set of differential tags, impacting recall positively. Later, when recall falls, that might still be an effect of what is being included in the set summary tags and what is not. It "might" because the selection of the synonyms to be included in the semantic field is a random procedure, making the additional tags vary if the number of additions is smaller than the number of available synonyms.

Now, focusing on the scores obtained by both methods, and disregarding the differences in evaluation methodology, the precision, recall and F1 scores are less than half the numbers presented by (Medelyan and Witten, 2008) for those same metrics on a much larger dataset related to the one used in the present work. However, their method generated tags based on a domain-specific controlled vocabulary and not from words extracted directly from the text.

Moving on to the evaluation of tag cloud generation, in Figure 5.3, the set summary tag cloud for the entire corpus was created using the method 1 and the word "agriculture" as root. In that tag cloud, it is possible to observe that, using that word as root, several tags have a clear logical connection with "agriculture", like "rice production", "food production" and "agriculture policy". It is also noticeable that the most relevant terms are represented with bigger font sizes, while the less relevant are smaller. However, it

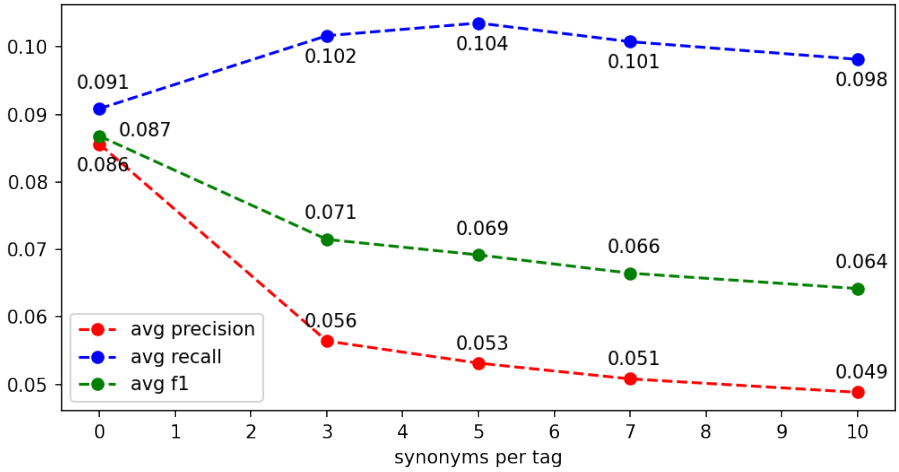


Figure 5.2: Results for differential tags using method 2.

is important to note that it was not evaluated if the tags presented summarize well the set of documents under the perspective of the concept of "agriculture".

When it comes to tag clouds produced using method 2, the situation is different. Despite having the most relevant tags highlighted with bigger fonts, the tag clouds are prone to contain noise, that are terms unrelated to the document or set of documents that is being represented. In Figure 5.4, for example, the differential tag cloud has the word "capacitance", that probably was included as a synonym of "capacity". However, "capacitance" has no relation with the document, making its presence in the tag cloud awkward.

Next, covering the processing times, Table 5.2 lists the time taken to generate tags (without tag clouds) using method 2 (values obtained while producing the previous results). The average time in this case is 50.79 seconds, which represents, approximately, 1.69 seconds per document. From Table 5.2, it is noticeable that, with the small corpus employed in this work, the increase in the number of synonym additions did not result in a relevant increase in processing time. It is also important to mention that, within that time, all three types of tag clouds are generated.

As for method 1, if considered only tag generation, it took 50.13 seconds to produce abstract and differential tags, without set summary tags or tag clouds. On the other hand, if set summary tags and tag cloud are generated, that time grows substantially because of the LSA step and the calculation of correlations. In one example, it took 224.03 seconds to produce all 3 types of tags and tag clouds. In method 1, if tag clouds are produced, the time also increases, but to a smaller extent, taking only 70.65 seconds.

Finally, covering a known issues with the library, during the tests, it was found that, by applying the POS-tagger after tokenization and stop word removal, but before stemming,

Figure 5.3: A set summary tag cloud from method 1.

Table 5.2: Processing Times For Method 2

Synonyms/Tag	Time (s)
0	50.18
3	50.37
5	51.45
7	50.98
10	50.98

it is possible for the POS-tagger to let words that are not nouns to be recognized as nouns. In (F. F. Morgado, 2010), the POS-tagging is executed last, after stemming. However, as of the moment of the writing of this text, it is not possible to assess if that would solve the issue.

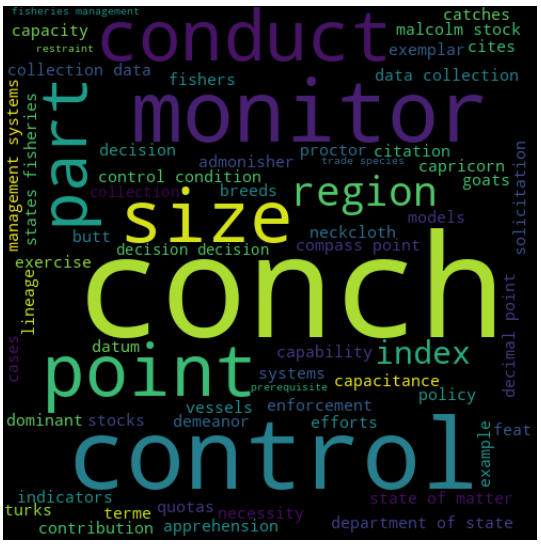


Figure 5.4: A differential tag cloud from method 2.

5.6 Conclusions

In this work, a portion of the capabilities of the GTAGLIB were shown and the results demonstrated that it can achieve some success within the proposals of the algorithms it implements. However, it is unfortunate that those same results were not good, and considerably low if compared to what is presented in (Medelyan and Witten, 2008), although it is important to note that the methods implemented in the library, despite being contemporary to that other work, do not rely on a controlled vocabulary. Another relevant aspect that can impact the results, but was not evaluated in this work, is the fact that some tags that are used as reference are not explicitly present in the text, like "disease surveillance" for the first document of the corpus.

Additionally, given the known issue presented in the previous section, in the current version, potential users might want to completely preprocess the documents before providing then to the library.

Considering that the library is in its early version (0.1) and that the tests were mostly limited to parameters stipulated by the original algorithms, when it comes to future works, they might include fixing bugs, testing the library with more parameter variations, testing it on a bigger corpus (specially one that is more commonly used for benchmarking, so that the results could be compared to other works), improving the performance of the library, implementing other tag generation techniques and bundling the library into add-ons for data mining tools, like KNIME[7] and Orange[8].

[7]KNIME: https://www.knime.com/.
[8]Orange: https://orangedatamining.com/.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

6

Classificação Indicativa a Partir de Sinopse Estendida

Ana Clara Correa da Silva e Gabriel S. Luna
Programa de Engenharia de Sistemas e Computação - COPPE

Abstract

Film culture has grown significantly in recent years. The sheer number of streaming services makes movies one of the world's most convenient forms of entertainment today. Films help to learn and inspire social change, but they can also negatively affect viewers. The purpose of this article is to programmatically predict the suitability of movie content for children based on extended synopses. We propose an architecture based on the Vector Space Model. We achieved 75% accuracy and an F1 score of 71%.

Resumo

A cultura cinematográfica cresceu expressivamente nos últimos anos. O grande número de serviços de streaming coloca os filmes como um dos mais convenientes formas de entretenimento no mundo de hoje. Filmes ajudam a aprender e a inspirar mudanças sociais, mas eles também podem afetar negativamente os espectadores. O objetivo deste artigo é prever programaticamente a adequação do conteúdo do filme para crianças com base em sinopses estendidas. Propomos uma arquitetura baseada no Modelo Espaço Vetorial. Alcançamos 75% de acurácia e de pontuação F1 71%.

6.1 Introdução

Classificação parental é uma diretriz criada para dar aos pais uma melhor compreensão do conteúdo de mídia que seus filhos consomem. As organizações de classificação geralmente são compostas por especialistas na indústria do entretenimento que criam e monitoram conjuntos de padrões de classificação. Esses padrões são aplicados de forma consistente em quase todas as novas mídias, para que os pais possam ter certeza de que receber uma classificação Livre é adequado para crianças de todas as idades. Cada tipo de mídia tem suas próprias diretrizes de classificação exclusivas. Embora essas classificações possam ser uma ferramenta útil, os pais são incentivados a pesquisar cada sistema de classificação de mídia para que compreendam completamente que tipo de conteúdo pode ser incluído. Em última análise, cabe aos pais decidir se o conteúdo é adequado ou não para suas famílias. No trabalho Romer et al., 2018, os autores mostraram que os pais são menos restritivos da visão infantil de violência armada em filmes PG-13 quando apresentam personagens cujo uso de armas é visto como justificado. Na visão dos autores, a aparente aceitação do aumento da violência armada em filmes PG-13 pode ser parcialmente atribuída à percepção de que a violência nesses filmes é justificada. Isso mostra a necessidade de informação e complementos para ajudar na tomada de decisão por parte dos pais.

6.1.1 Iniciativas Formais

Nos Estados Unidos a maioria das classificações são modelada de acordo com o sistema de classificação de filmes da Motion Picture Association (MPA) [1]. Um resumo das categorias pode ser verificado na tabela 6.1. As categorias possuem equivalentes para conteúdos produzidos e/ou transmitidos na TV. A participação voluntária, com classificações a serem determinadas pelas redes de transmissão participantes individualmente, isto é, para conteúdos não categorizados pela MPA. As diretrizes em si não têm força legal, e não são usados em programas de esportes, notícias ou durante anúncios comerciais. Muitos serviços de televisão online, como Hulu, Amazon Video e Netflix, também usam o sistema de diretrizes, juntamente com fornecedores de vídeo digital, como iTunes Store e Google Play, e players de mídia digital, incluindo Amazon Fire TV, Apple TV, Android TV, e plataformas Roku.

O objetivo deste trabalho é apresentar uma classificação indicativa de filmes baseada em sinopses estendidas utilizando o modelo vetorial. Para tal, apresentamos outros estudos que utilizaram uma abordagem semelhante e discutimos seus resultados na seção 6.2. Mostramos a metodologia utilizada para coletar os dados necessários na seção 6.3 e a descrição do modelo utilizado na seção 6.4. Discutimos os resultados na seção 2.4 e encerramos o artigo com considerações finais na seção 6.6.

[1] https://www.motionpictures.org/film-ratings/

Tabela 6.1: Classificação Indicativa de acordo com o MPA

Símbolo	Descrição
G	General Audiences (Livre) Livre para todas as idades. Não contém nenhum material inapropriado para crianças.
PG	Parental Guidance Suggested (Orientação dos pais) Alguns materiais podem não ser apropriados para crianças. Pais são incentivados a dar "orientação parental", pode conter materiais inapropriados para crianças mais novas.
PG-13	PG-13 – Parents Strongly Cautioned (Orientação dos pais Recomendado) Alguns materiais podem ser inapropriados para menores de 13 anos. Pais são incentivados a serem cuidadosos, alguns materiais podem ser inapropriados para pré-adolescentes.
R	Restricted (Restrito) Menores de 17 anos requer um acompanhamento dos pais ou responsáveis. Contém alguns materiais adultos, pais devem aprender mais sobre o filme antes de levar suas crianças.
NC-17	Adults Only (Exclusivo Para Adultos) Proibido para menores de 18 anos. Material claramente adulto, menores de idade não são permitidos assistir ao filme (pode ser usado para classificar filmes pornográficos).

6.2 Trabalhos Relacionados

Há diversos trabalhos quer procuraram detectar violência a partir de vídeos como em Saad et al., 2022 e Khan et al., 2019. Porém isso exige um grande capacidade de processamento e muitas das vezes é inviável.

No trabalho proposto em Martinez et al., 2019 os autores utilizaram uma abordagem baseada em uma ampla gama de recursos projetados para capturar características lexicais, semânticas, sentimentais e de linguagem abusiva. Os autores treinaram um modelo de classificação de aprendizado profundo que faz uso de contexto em um conjunto de dados de 732 roteiros de Hollywood categorizados por especialistas como conteúdo violento. Os resultados mostraram que as características semânticas e de sentimento são os preditores mais importantes de violência nesses dados.

Já em Shafaei et al., 2019, os autores utilizaram uma arquitetura de classificação baseado em Redes Neurais treinadas a partir de roteiros completos de filmes e análises de sentimentos das falas dos personagens. Nada foi mencionado acerca da precisão mediante filmes slasher, que são filmes em que assassinos matam aleatoriamente suas vítimas e não possuem falas no filme.

Na monografia Chai, 2021, a tarefa de classificação consiste em um pipeline com 5 modelos treinados. Uma cena de detecção de nudez é treinada usando YOLOv4 para detectar possíveis cenas expondo partes íntimas e genitais. Enquanto isso, a detecção de cenas violentas é treinada usando RNN-LSTM personalizado para detectar possíveis cenas de luta e sangue. Em seguida, a detecção de palavrões usa a API Text-to-Speech do Google para transcrever o áudio antes de alimentá-lo em uma biblioteca personalizada de palavrões. Por fim, os modelos de álcool e drogas são treinados usando recursos extraídos do VGG-16 e depois inseridos em um classificador CNN de uma classe. Tem sido uma

tendência na área de classificação parental automática o uso de diferentes modelos para cada aspecto do conteúdo.

Essas abordagens demandam grande capacidade de processamento computacional bem como grande capacidade de armazenamento para lidar com tamanho volume de dados. Muitas vezes essas soluções são inviabilizadas por falta de disponibilidade desses recursos. Se faz necessário esforço no desenvolvimento de abordagens "leves"ou "baratas"para acessibilizar a automatização da tarefa de classificação indicativa, o qual é o objetivo desse trabalho.

6.3 Coleta de Dados

A maior fonte de dados sobre filmes e conteúdos semelhantes hoje é o site do IMDB[2]. Dado que o IMDB não disponibiliza APIs ou outro tipo de interface para extração de dados nem os disponibiliza para download, a coleta de dados foi feita utilizando técnicas de raspagem de dados. Raspagem de dados é uma técnica para extrair informações de vários documentos da web automaticamente. Ele recupera o conteúdo relacionado com base na consulta, agrega e transforma os dados de um formato não estruturado em uma representação estruturada. Os parâmetros de busca foram pensados para extrair sinopses com a maior diversificação de classificações indicativas. Como há um maior número de roteiros na língua inglesa, utilizamos apenas sinopses em inglês e consequentemente todo o pré-processamento foi modelado para a língua inglesa. A URL de busca resultante foi `https://www.imdb.com/search/title/?title_type=feature&release_date=2010-01-01,2021-12-31&languages=en&adult=include`. O resultado pode ser visto na figura 6.1. O código foi construído na linguagem python Mitchell, 2018, utilizando as bibliotecas BeautifulSoup para lidar com o processo de raspagem e Requests para fazer as requisições ao site do IMDB. Utilizamos a função sleep para ajudar a evitar que o servidor fique sobrecarregado com muitas solicitações em um período de tempo muito curto. Com o sleep foi possível fazer o script parar por um certo período de tempo e evitar que servidor ficasse sobrecarregado por conta das requisições de forma iterativa. Dessa forma, os servidores (IMDB) não conseguiram encontrar um padrão comum claro para essas solicitações e eventualmente gerar um banimento.

Para cada filme na página, coletamos Id, título, duração, gênero, Rating score e então finalmente realizamos uma nova requisição para coletar a sinopse na página do filme. Filmes que não possuíam informações de classificação indicativa ou que não possuíam sinopses foram descartados e o programa saltava para a próxima interação do *loop*.

[2]https://www.imdb.com/

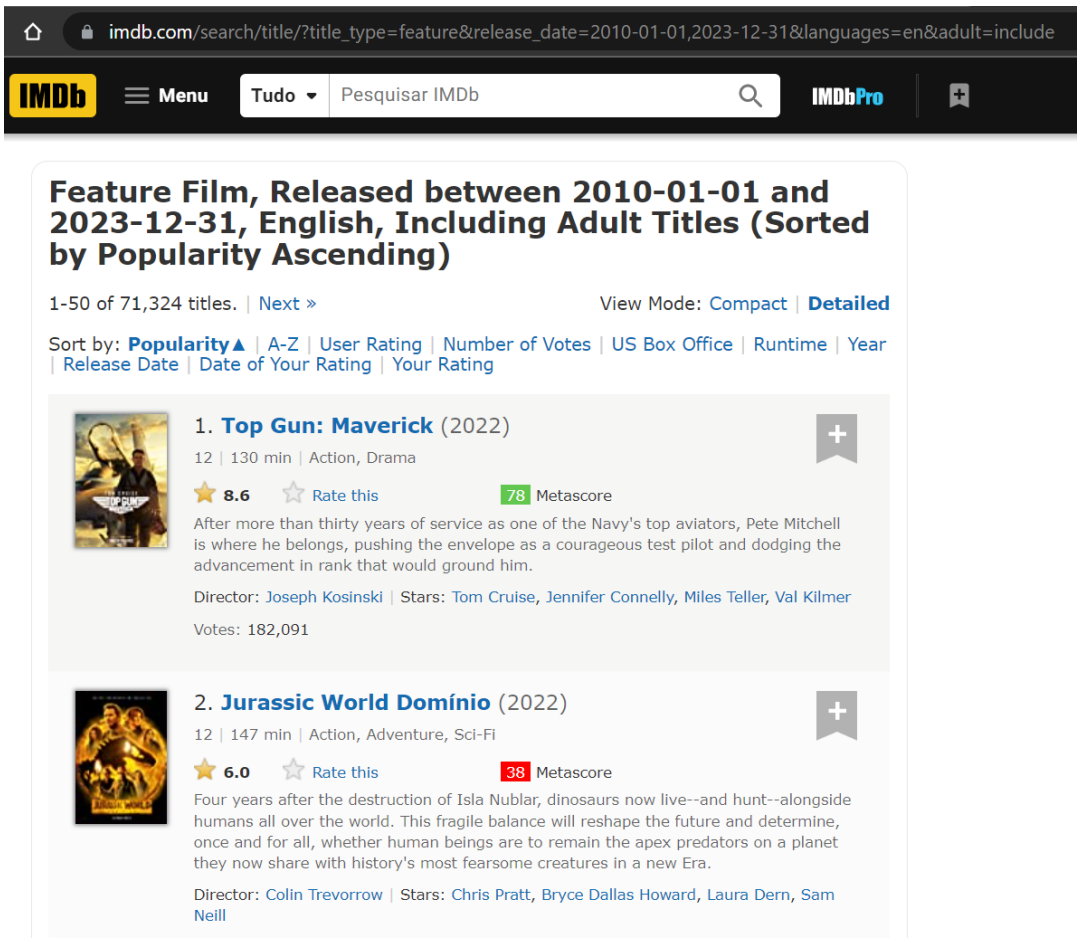


Figura 6.1: Site IMDB com resultados de busca

6.3.1 Exploração de Dados

Foram coletados um **total de 6.837** filmes. As categorias encontradas no conjunto de dados podem ser verificadas na figura 6.2. No conjunto há filmes que foram lançados exclusivamente para a TV e por isso possuem uma classificação diferente porém equivalente a da MPA (seção 6.1.1).

Na figura é possível ver que mais da metade dos filmes são restritos, ou seja, há conteúdos impróprios para menores e cabe aos pais fazer uma avaliação mais apurada. Outra análise é acerca da quantidade de filmes restritos ou não para menores de acordo com o gênero indicado nas figuras 6.3 e 6.4

De acordo com o conjunto de dados, os 5 gêneros mais propensos a terem conteúdos inadequados são: drama, comédia, suspense, ação e crime. Já os 5 gêneros mais propensos a terem conteúdos livres são: Drama, comédia, Aventura, Ação e romance . Os filmes podem ser classificados com um ou mais gêneros. Sendo drama e comédia os gêneros mas presentes em ambas as classificações é possível dimensionar a dificuldades de distinguir esses casos de forma automática.

Approved	1
G	23
MA-17	1
NC-17	8
PG	434
PG-13	1056
R	2333
TV-14	160
TV-G	15
TV-MA	326
TV-PG	45
TV-Y	2
TV-Y7	15

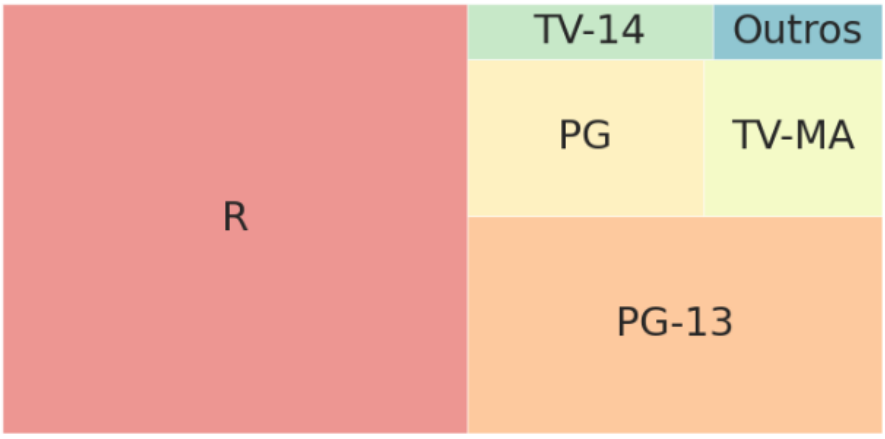


Figura 6.2: Quantidade de filmes por categoria

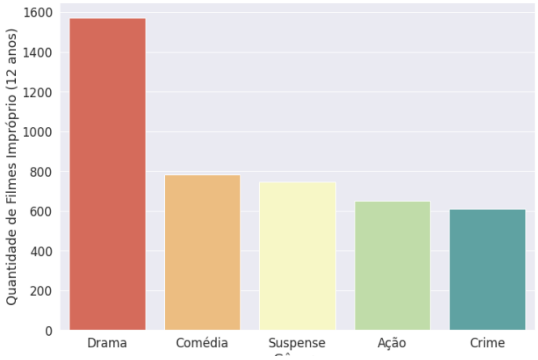


Figura 6.3: Quantidade de filmes inapropriados por gênero

Outra análise para melhor compreendermos o conjunto se dá nas categorias de classificação mais presentes demonstradas na figura 6.5.

Para fazer testes iniciais e solver a principal dificuldade dividimos o conjunto de dados em dois grupos: conteúdos permitidos para menores de 12 anos e conteúdos não permitidos. A proporção se encontra demonstrada na figura 6.6.

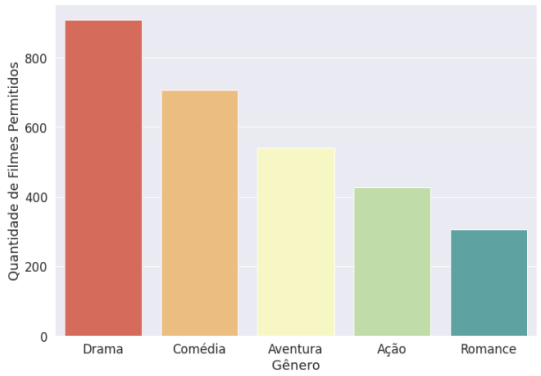


Figura 6.4: Quantidade de filmes com classificação livre por gênero

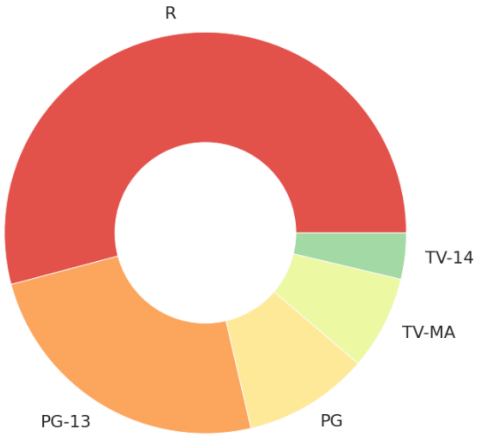


Figura 6.5: Categorias de Classificação Indicativa mais Presente no Conjunto de Dados

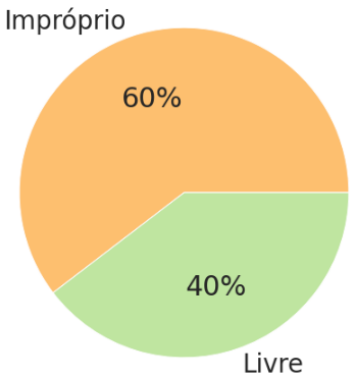


Figura 6.6: Quantidade de filmes permitidos para crianças até 12 anos

6.4 Modelo

Para a criação do modelo, foi utilizado o PySpark junto com suas bibliotecas de Machine Learning. O PySpark consiste em um interface que permite acessar o Spark utilizando código em python. O spark foi originalmente desenvolvido na AMPLab da universidade de Berkeley em 2009, teve como objetivo principal a criação de uma estrutura de processamento de dados poderosa, que fornece uma ferramenta de fácil utilização, onde sua arquitetura é baseada no esquema master-slave Inoubli et al., 2018. Seu módulo MLib (Machine Learning) fornece diversas bibliotecas para serem utilizadas no aprendizado de máquina de modelos. Um pipeline de Machine Learning foi utilizado para a criação do modelo de aprendizagem, sendo o pipeline um fluxo completo que combina diversos algoritmos de Machine Learning em conjunto. Há diversas etapas para processar e aprender com os dados oferecidos, exigindo que uma sequência de algoritmos seja seguida. O pipeline utilizado para a construção desse trabalho está representado na figura abaixo:

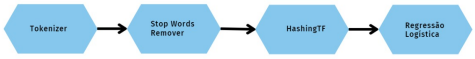


Figura 6.7: Pipeline de Machine Learning

1. Tokenização (Análise Léxica): Consiste na separação e armazenamento de tokens, é nesse processo também que se retira acentos e coloca o texto em caixa baixa se necessário.
2. Stop Word Remover : É utilizado para a remoção de palavras sem valor semântico.
3. HashingTF : Mapeia uma sequência de termos para suas frequências de termo usando o truque de hashing.
4. Regressão Logística : É um modelo de análise de regressão, e é utilizado para predizer a probabilidade de uma resposta binária (sim ou não) baseado em uma ou mais variáveis independentes.

Para o treinamento do modelo, foi utilizado o dataset colhido a partir da coleta de dados explicada na seção 3 deste artigo. Com isso separamos os conteúdos em labels na qual foi atribuído o valor 0 para conteúdo apropriado para criança e 1 para conteúdo não apropriado para crianças.

6.5 Resultados

Foram separados 70% do conjunto para o treinamento do modelo e 30% para o teste desse modelo. Abaixo temos o resultado gerado pelo modelo.

Tabela 6.2: Métricas

Métrica	Valor
Acurácia	75%
Precisão	70%
Recall	71%

6.5.1 Resultados no portal de notícias g1.com.br

Foi feito um script para pegar todas as principais notícias do portal do g1.com.br afim de predizer se o conteúdo é apropriado ou não para menores de 18 anos. Com o web scraping do portal, no dia 26/06/2022 foi realizado a consulta e obtemos a seguinte notícia Globo, s.d. classificada como imprópria para menores de 18 anos de acordo com o algoritmo.

6.6 Conclusão

Nesse trabalho foi demonstrado que uma solução "leve"em termos de capacidade computacional não afeta negativamente na solução do problema mapeado. Com o modelo é possível dar suporte aos pais para selecionar melhor os conteúdos expostos a seus filhos e proteger a inocência que a infância precisa. O código fonte utilizado nesse projeto encontra no link de uma pasta no drive com notebooks utilizados Google Drive - Parental Rating Como trabalhos futuros pretendemos avaliar se o modelo pode ser aplicado para classificar conteúdos de outras mídias como jogos, websites e outros materiais comum à crianças.

7

Predição de Gêneros Cinematográficos com Base em Legendas

João Pedro G de Souza e Matheus Avellar de Barros
Programa de Engenharia de Sistemas e Computação - COPPE

Resumo

Legendas de filmes desempenham um papel muito importante, tanto na área de acessibilidade, aumentando a qualidade de vida de pessoas com deficiência, quanto em termos de preferências pessoais, possibilitando ao público em geral assistir a conteúdos no idioma original. Porém, além do seu papel no alcance dos conteúdos, legendas, como transcrições de filmes inteiros, carregam informações relevantes acerca das mídias a que são associadas. Por isso, este trabalho busca averiguar uma das informações que poderiam ser inferidas a partir de legendas: o gênero cinematográfico.

7.1 Introdução

Acessibilidade é um fator crucial para a sociedade, pois possibilita a inclusão de uma maior parcela da população em diferentes atividades. Essa crucialidade é refletida na tecnologia, onde diferentes dispositivos e *softwares* são desenvolvidos para melhor adequar a experiência ao público deficiente ou incapacitado – como, por exemplo, programas leitores de texto ou de tela, chamados *Text To Speech*, para cegos e deficientes visuais.

Nesse contexto, legendas são uma importante ferramenta de acessibilidade na tecnologia. Elas podem ser usadas para introduzir conteúdo para um público que não compre-

ende um determinado idioma, e permitem a surdos e deficientes auditivos compreender e aproveitar conteúdos como vídeos, filmes e desenhos animados em sua totalidade. Em um estudo realizado em 2022, 31% dos entrevistados brasileiros declararam preferir assistir a conteúdo com o idioma original e legendas em português, enquanto que, em países do leste asiático como China e Coreia do Sul, esta porcentagem chega aos 72% (Shevenock, 2022).

Legendas possuem um impacto tão profundo que a gigante multinacional Google anunciou, em 2022, planos para lançar um par de óculos de realidade aumentada que introduz tradução simultânea de conversas através de legendas visíveis somente ao usuário.[1]

Dada sua importância, é de interesse geral verificar quais informações podemos extrair ou inferir a partir delas. Por exemplo, será possível identificar o gênero cinematográfico de um filme – como Ação, Aventura, Fantasia – a partir somente de sua legenda? Serão os diálogos suficientes para informar um algoritmo qual classificação designar a um filme? Este trabalho procura responder a essas perguntas.

Utilizaremos um conjunto de dados extensivo de legendas em português brasileiro de diversos filmes para treinar um algoritmo de aprendizado de máquina em seus gêneros cinematográficos; em seguida, utilizaremos o algoritmo treinado para tentar estimar o gênero de filmes que o algoritmo ainda não "viu", e tentaremos estimar o quão perto da verdade o programa chegou.

7.2 Conjunto de dados

Para nosso corpus, decidimos utilizar o *dataset* de legendas em português da OpenSubtitles,[2] disponibilizado por Lison e Tiedemann, 2016. A decisão se deu por dois fatores principais:

- Esse conjunto de dados já contém a classificação de gêneros cinematográficos embutida, evitando assim a necessidade de um segundo *dataset* para associar legendas individuais aos seus respectivos gêneros; e
- O *dataset* é enorme e possui legendas o suficiente para a realização do trabalho.

O *dataset* original consiste em um arquivo ZIP de 7.54 GB, contendo arquivos XML com legendas de 3 milhões de filmes distintos.[3] Para nosso conjunto, utilizamos somente legendas de filmes de 2016 e 2017, o que já agrega 19,467 arquivos de legenda.

[1] B. Heater. "Google Glass's successor teased at I/O". TechCrunch (11 mai.2022). Disponível em: https://techcrunch.com/2022/05/11/google-glasss-successor-teased-at-i-o/. Acesso em 17 jun.2022.

[2] http://www.opensubtitles.org/

[3] *Dataset* disponível em: https://opus.nlpl.eu/OpenSubtitles-v2018.php

7.2.1 Pré-tratamento

Um tratamento inicial foi realizado nos arquivos que compuseram nosso *dataset*, feito através de um *script* dedicado para tal. Foi necessário convertê-los de XML para um formato textual simples, e separamos a legenda dos metadados desnecessários. Os arquivos tratados possuem uma linha contendo os gêneros cinematográficos, separados entre si por um caractere "`;`", e as linhas seguintes o conteúdo textual da legenda.

Além disso, algumas medidas foram tomadas para evitar ruído. Legendas com menos de 100 linhas de diálogo foram totalmente excluídas, e as primeiras e últimas 20 linhas de diálogo foram removidas de todas as legendas restantes.

O raciocínio por trás desta última medida está no fato de que, em muitos casos, as legendas são escritas ou traduzidas por grupos de voluntários não afiliados com os produtores do filme – chamados de *fansubbers*.[4] E, em grande parte, esses grupos inserem créditos em suas legendas. Assim, o nome das equipes poderia poluir os dados – se, por exemplo, um grupo fosse responsável por uma grande parte das legendas de filmes de terror, o algoritmo aprenderia a classificar filmes como "terror" ao encontrar o nome da equipe, e este não é o objetivo do experimento.

Após o pré-tratamento, passamos a 19,293 arquivos de legenda – isto é, uma perda de aproximadamente 200 legendas.

7.3 Desenvolvimento

A classificação do gênero de um filme é uma do tipo *multilabel*, ou 1 para N, pois um filme pode ter mais de um gênero. Por exemplo, o filme Avatar (2009), segundo o Internet Movie Database (IMDb), é dos gêneros Ação, Aventura e Fantasia.[5] Porém, para realizar esta tarefa é preciso um modelo que consiga classificar cada legenda com mais de um gênero, ou simplificar o problema para utilizar modelos que classifiquem apenas um.

Por isso, decidimos aproximar o problema de previsão de gênero cinematográfico por dois métodos diferentes: de um lado, treinar um classificador por gênero e *Classifier Chain* (Read et al., 2009), que combinam vários classificadores binários em um modelo que consegue utilizar correlações entre eles; e, do outro, um aprendizado de *fuzzy* K-NN. Assim, poderíamos comparar os resultados posteriormente para determinar qual método é mais adequado.

[4] "Fansubbers: conheça os fãs que produzem legendas para séries". Correio Braziliense: Diário de Pernambuco, (3 mai.2015). Disponível em: https://s.avl.la/8f4q1. Acesso em 17 jun.2022.

[5] https://www.imdb.com/title/tt0499549/

7.3.1 Estratégia de Classificação

Inicialmente, queremos utilizar os documentos de texto nos modelos de aprendizado de máquina, mas para isso é necessário extrair vetores numéricos do conteúdo dos documentos. Por esse motivo, utilizamos o algoritmo *Term Frequency times Inverse Document Frequency* (tf-idf), que utiliza a frequência dos termos e a frequência de documento inversa para produzir um peso para cada termo em cada documento e, com isso, documentos que utilizem as mesmas palavras relevantes tendem a ter vetores similares (Manning, Raghavan e Schütze, 2008). Além disso, utilizamos o *stemmer* para normalizar as palavras e removemos os *stop words*. Apesar do tf-idf já atribuir um peso menor para as palavras muito frequentes, quando as removemos, o tamanho do vetor de *features* diminui, com isso simplificamos o modelo e temos um ganho computacional.

Em seguida implementamos a primeira estratégia que consiste em treinar um classificador por gênero. Nesta parte utilizamos a função do scikit learn[6] chamada *Multi Output Classifier* e os algoritmos *Radius Neighbors Classifier* e *Random Forest Classifier* como estimadores. Em seguida iniciou-se a busca por melhores hiper-parâmetros em cada modelo. Nesta parte utilizamos a função *Grid Search CV*[7], que realiza uma busca exaustiva para encontrar os hiper-parâmetros que apresentaram o melhor resultados dentre os especificados para busca. Contudo realizar a busca utilizando todo o *dataset* não foi viável, devido a falta de um computador que conseguisse realizar esta tarefa. Desse modo, a busca pelos hiper-parâmetros foi feita com 10% do conjunto total de dados. Somente após esta busca foi feito o treinamento dos modelo final.

Seguindo para segunda estratégia, que consiste na utilização do *Classifier Chain*[8], utilizamos os hiper-parâmetros encontrados na primeira estratégia e os mesmos estimadores. No entanto, alteramos a função *Multi Output Classifier* pelo *Classifier Chain* do scikit learn e com isso, poderemos comparar os resultados obtidos.

7.3.2 Fuzzy *K*-NN

Devido à necessidade de classificar legendas em mais de um gênero cinematográfico, não era possível utilizar somente a solução de K-Nearest Neighbors. Esta classifica pontos em somente uma categoria, baseado nos K vizinhos mais próximos do ponto avaliado.

Contudo, existe uma variação, denominada *fuzzy* K-Nearest Neighbors (abreviada aqui "FkNN"). Desenvolvido por Keller, Gray e Givens, 1985, o algoritmo altera o K-NN para permitir a classificação de um ponto em múltiplas categorias simultaneamente.

[6] https://scikit-learn.org. Acesso em 17 jun.2022.
[7] GridSearchCV. Acesso em 17 jun.2022.
[8] https://scikit-learn.org/stable/modules/multiclass.html. Acesso em 17 jun.2022.

Fizemos a implementação do algoritmo descrito pelo artigo de 1985 em um ambiente virtual do Google Colaboratory ("Colab"),[9] na linguagem de programação Python.[10]

Também houve um tratamento dos dados de legenda antes da etapa de classificação. Palavras às vezes são capitalizadas, pontuações variam, e verbos são conjugados. Assim, é interessante realizar uma padronização do texto de entrada. Isso foi feito pelos seguintes passos:

1. Separação de acentos e sinais gráficos via normalização NFKD[11] – e.g. "`ç`" (U+00E7) se torna um par composto por "`c`" (U+0063) e "`COMBINING CEDILLA`" (U+0327);
2. Conversão para texto ASCII, ignorando caracteres fora do padrão – assim removendo os sinais gráficos separados no passo anterior;
3. Conversão para minúsculas – e.g. "`C`" vira "`c`";
4. Remoção de espaços em branco supérfluos no início ou final da linha – e.g. "` a b c `" vira "`a b c`";
5. Finalmente, cada palavra é passada pelo "RSLP Stemmer" (Removedor de Sufixos da Língua Portuguesa), disponível na biblioteca NLTK.[12]

O último passo, chamado "*stemming*", serve para aglomerar palavras similares com grafias distintas – por exemplo, "garoto" e "garotas" são palavras diferentes, mas com significados muito próximos; após o *stemming*, ambas se tornam "garot", e podem ser mais facilmente contadas automaticamente como múltiplas ocorrências de um único conceito.

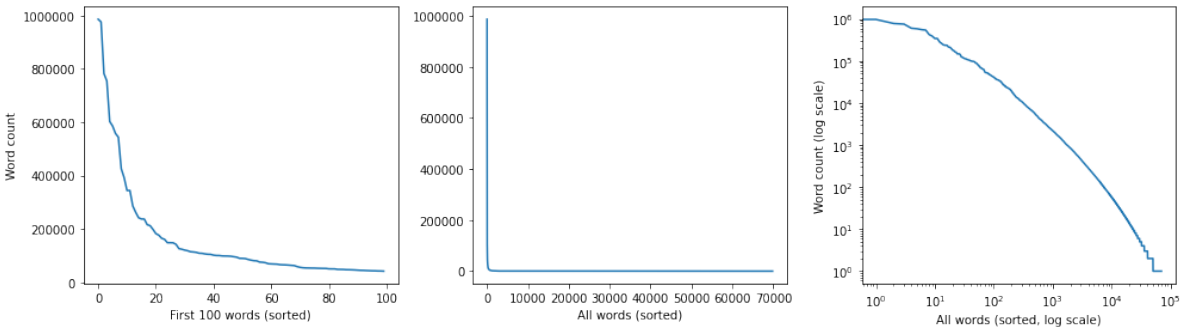


Figura 7.1: Distribuição de ocorrência das 100 palavras mais comuns, de todas as palavras, e de todas as palavras em escala logarítmica, respectivamente. Amostragem de 7,000 arquivos de legenda aleatórios.

A Fig. 7.1 mostra a ocorrência de palavras em uma amostragem do *dataset*, após o tratamento de palavras descrito acima.

Em seguida, para uma melhor redução de ruído no conjunto de dados, as 100 palavras mais frequentes foram removidas. Estas provavelmente estão presentes em todos os

[9] https://colab.research.google.com/
[10] https://www.python.org/
[11] Disponível em: https://unicode.org/reports/tr15/#Norm_Forms
[12] Disponível em: https://www.nltk.org/_modules/nltk/stem/rslp.html

filmes, e portanto não contribuem muito para a classificação de gênero. Similarmente, palavras presentes em somente 1 filme também foram retiradas do conjunto.

Finalmente, para preparar os dados para introdução no algoritmo, foram criados a matriz X de *inputs* e o vetor y de *output*.

X é uma matriz onde cada linha representa um filme, e cada coluna representa uma palavra. Os valores são as frequências relativas de cada palavra naquele filme, como porcentagem de 0 a 100; i.e. $X_{ij} \in [0, 100]$.

y é um vetor de gêneros de filme. Cada linha representa o mesmo filme da linha correspondente em X, e cada coluna é a porcentagem de inclusão do filme no gênero correspondente – e.g. se um filme é classificado como Ação e Aventura, ele possui 50% na coluna correspondente ao gênero de Ação, 50% em Aventura, e 0 em todos os outros campos. Assim, similarmente, $y_{mn} \in [0, 100]$.

7.4 Resultados

7.4.1 Random Forest Classifier, Classifier Chain e Multi Output Classifier

Comparando os mesmos estimadores na primeira estratégia e na segunda, é possível ver que quando o estimador era o Radius Neighbors Classifier a primeira estratégia apresentou, no geral, o mesmo resultado que a segunda. A média ponderada da precisão, do recall e do F1-score entre todos as classes foi 0.93, 0.61 e 0.69 respectivamente. E a média macro da precisão, do recall e do F1-score entre todos as classes foi 0.91, 0.40 e 0.54 respectivamente. Porém, quando comparamos o resultado com o estimador Random Forest Classifier, o resultado na segunda estratégia foi ligeiramente melhor que a primeira. A média ponderada da precisão, do recall e do F1-score entre todos as classes foi 0.97, 0.82 e 0.88 respectivamente nas duas estratégias, mas a média macro foi diferente. Utilizando o Classifier Chain obtivemos um resultado melhor nas seguintes classes: Crime, Game-Show, History, Reality-TV, War. A Tabela 7.1 apresenta os resultados do Random Forest Classifier nas duas estratégias. Este resultado é, provavelmente, devido a correlação entre as classes, que o algoritmo Classifier Chain utiliza para classificar melhor. Por exemplo: as classes Reality-TV e Game-Show tem uma correlação de 0.909795, as classes Crime e Mystery tem uma correlação de 0.337380 e as classes Biography e History tem uma correlação de 0.305275

Dentre as estratégias apresentadas, a que utilizou o estimador *Random Forest Classifier* conseguiu os melhores resultados. E comparando a utilização do *Random Forest Classifier* com o *Classifier Chain* obtivemos um resultado um pouco melhor que o mesmo estimador com o *Multi Output Classifier*.

Tabela 7.1: Resultado com o estimador Random Forest Classifier

média macro	precisão	recall	F1-score
1^a **estratégia**	0.91	0.65	0.74
2^a **estratégia**	0.91	0.66	0.75

7.4.2 Fuzzy *K*-NN

O algoritmo de K-NN depende de uma escolha adequada de K – isto é, quantos vizinhos serão comparados para classificar um ponto. Assim, executamos o algoritmo com valores ímpares incrementais para $K \in [3, 20)$, e calculamos a soma de erro para cada para determinar o K com menor erro.

O erro é calculado de duas formas diferentes. Para o FkNN em si, o erro é a distância entre os vetores, utilizando a fórmula padrão:

$$\text{Erro}_{\text{dist}}(a, b) = \sqrt{\sum_i (a_i - b_i)^2}$$

Os somatórios de erros encontrados estão presentes na tabela 7.2, que relaciona uma escolha de K com seu erro total. A amostra possuiu 7,000 arquivos. O menor erro encontrado foi para $K = 11$; resultados posteriores do FkNN usam este valor para K.

Vale ressaltar que a escolha de K pode flutuar conforme a amostra, e pode ser dependente dos dados. Assim, este valor específico não é uma recomendação por parte dos autores.

Tabela 7.2: Erro encontrado para diferentes valores de K, com valores arredondados para o número inteiro mais próximo

K	3	5	7	9	11	13	15	17	19
Erro	4,916	2,400	1,460	1,001	611	1,069	920	1,498	4,108

A segunda implementação de função de erro foi usada para avaliar o desempenho do algoritmo. Ela retorna um valor no intervalo $[0, 1]$, e é definida pela equação 7.1:

$$\text{Erro}_{\text{pct}}(a, b) = \frac{1}{2} \sum_i |a_i - b_i| \tag{7.1}$$

Para a execução final do algoritmo, após separação entre pontos de treino e teste, foram usados 6,180 pontos classificados corretamente – os "vizinhos" –, e o FkNN classificou

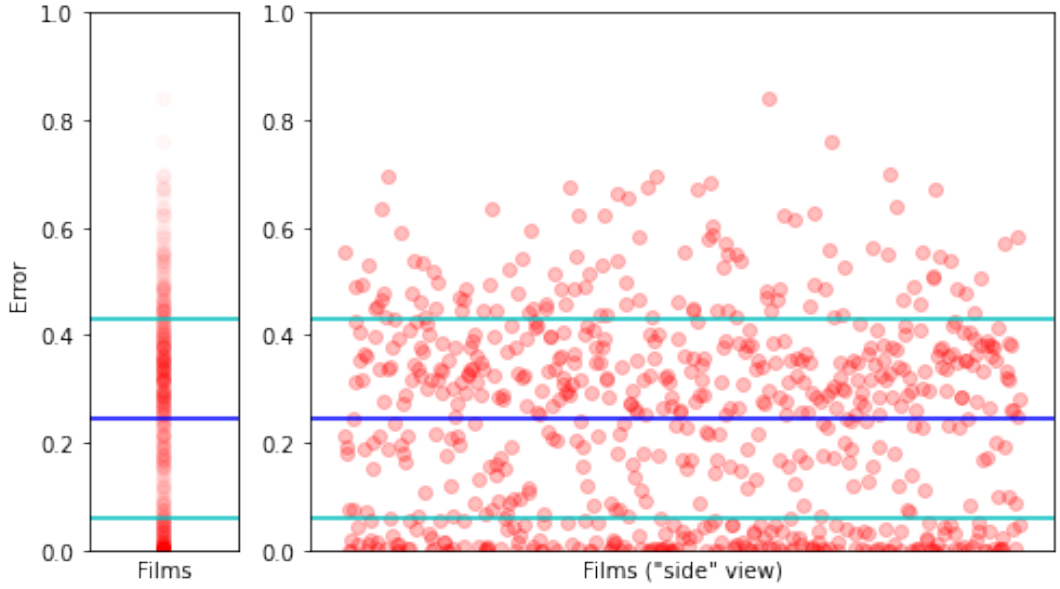


Figura 7.2: Erro_{pct} por filme – sobrepostos horizontalmente, e separados, respectivamente. Cada ponto vermelho representa um filme classificado via FkNN, e sua posição vertical corresponde ao erro em relação à classificação real.

687. A Fig. 7.2 indica o erro por cada filme classificado pelo FkNN. A linha azul é a média de erro encontrado ($\overline{\text{err}} = 0.24$), e as linhas ciano indicam o desvio padrão ($\sigma = 0.19$). O erro mínimo foi ≈ 0.00 (0 nas primeiras 16 casas decimais), e o erro máximo foi 0.84. Dentre todos os pontos classificados, 178 (25.90%) possuem erro menor que 0.05.

7.5 Conclusão

Dada a importância da consideração por acessibilidade em artigos de cultura, como filmes e vídeos, e o papel que legendas cumprem como veículo de inclusividade, evidencia-se que experimentações com essas ferramentas trazem maiores *insights* sobre como melhor utilizá-las.

Mostramos que, ainda que com certo sucesso seja possível prever gêneros cinematográficos a partir de legendas, esta não é uma tarefa trivial, e as técnicas utilizadas neste trabalho não são suficientes para garantir um percentual de acerto aceitável para seu uso no mundo real. Contudo, há valor experimental e acadêmico em determinar os limites, tal qual as capacidades, de análises realizadas sobre legendas.

Em trabalhos futuros, seria interessante a comparação com o desempenho de outros algoritmos de aprendizado de máquina não testados aqui. Além disso, há uma curiosidade dos autores no que tange às diferenças entre idiomas – será mais fácil (ou mais difícil) prever gêneros se as legendas estão em outras línguas? Aguardamos ansiosamente futuros desenvolvimentos na área.

8

Classificação de Estilo Musical Usando Técnica de Mineração de Texto

Rodrigo Pagliusi e Vitor Paes
Programa de Engenharia de Sistemas e Computação - COPPE

Resumo

Com o crescente uso da mineração de texto, o uso dessa técnica no universo musical tem ganhado notoriedade. É possível usá-la para descrever e trazer informações sobre as letras e seus estilos musicais, principalmente adicionando elementos de Inteligência Artificial como componente para interpretação do texto.

Por essa razão, esse estudo propõe a classificação de músicas em seu estilo musical, usando a técnica de mineração de texto, a partir de dados obtidos em um site especializado em música.

8.1 Introdução

O uso de termos como Text Mining, Text processing ou mesmo Text analitics, relativamente recentes, busca indicar um processo semi-automatizado para extração de conhecimento de fontes de dados não-estruturados, como documentos de Word, arquivos PDF, fragmentos de texto, arquivos XML, etc. Isto o difere da própria mineração de dados, onde os dados estão previamente estruturados em bancos de dados.

Esse novo panorama tem fornecido subsídios para uma área particular da Computação Musical, a Classificação Musical, que fornece informações valiosas sobre a evolução dos

padrões de composição e também na geração de catálogos de músicas (Simões, Lourenco e J. Almeida, 2007). “Atualmente, a maioria dos sites oferece um grande número de itens (por exemplo, filmes, músicas, páginas da web, etc.) Encontrar conteúdos relevantes de acordo com os gostos de cada um tornou-se, assim, um desafio. (Domingues et al., 2014)”

As relações entre as pessoas e músicas de diversos estilos e países são estreitas, devido ao acesso à informação e às tecnologias de comunicação, em especial sites especializados, que fornecem uma fonte rica de dados não estruturados de forma prática e acessível. Por esse motivo este trabalho utiliza como fonte aberta de acesso a dados musicais, o “Letras”, um site que integra musicas dos mais variados gêneros, além das suas letras, traduções e vídeos, limitando-se a análise e identificação da composição e padrões musicais do gênero conhecido como “Alternativo”.

Neste artigo, exploramos uma técnicas de mineração de texto para capturar o estilos de dados textuais das letras das músicas.

Este artigo tem como objetivo o estudo de música alternativa e propõe um experimento, visando estimular o conhecimento e métodos para classificação por estilo musical. Após essa rápida introdução ao tema, a seção 2 apresenta trabalhos relacionados. Na seção 3 são discutidos fundamentos teóricos e na seção 4 são tratados resultados e discussões. Por último, na seção 6 são apresentadas as considerações finais com as limitações e trabalhos futuros.

8.2 Trabalhos Relacionados

Como trabalhos relacionados destacamos: um sistema automático de classificação de letras usando técnica de mineração de texto (Jareanpon et al., 2018), a análise de partituras de música clássica (Simões, Lourenco e J. Almeida, 2007) e a mineração de dados musicais para distribuição de música eletrônica (Pachet, Westermann e Laigre, 2001).

O artigo de Jareanpon et al. (2018), *Automatic lyrics classification system using text mining technique*, propõe um sistema automático de classificação de letras usando a técnica de mineração de texto, na língua tailandesa, baseada na emoção. As 120 letras usadas neste experimento que são categorizadas em 3 grupos: amor, falta e música de coração partido. A média das letras é de 262,9 palavras.

O trabalho de Simões, Lourenco e J. Almeida (2007), *Using Text Mining Techniques for Classical Music Scores Analysis* , se destaca ao classificar a música com base nas informações das partituras, e fornece contribuições para o processamento da representação simbólica da música, além de uma análise posterior. Técnicas de mineração de texto suportam o processamento de partituras enquanto técnicas de classificação são usadas na construção de modelos de decisão. A análise envolveu a seleção de recursos e

a avaliação de vários algoritmos de mineração de dados, garantindo extensibilidade para repositórios maiores ou problemas mais complexos.

Já em *Musical data mining for electronic music distribution*, Pachet, Westermann e Laigre (2001) propõem um método de classificação amparada em uma técnica para classificação baseada na análise de co-ocorrência e correlação usada pa. São usadas informações textuais referentes a títulos de músicas ou artistas. Com base em uma técnica de agrupamento, mostraram que grupos podem revelar gêneros musicais específicos e permitir a classificação de títulos musicais de maneira objetiva.

8.3 Fundamentos Teóricos

8.3.1 Mineração de texto

A mineração de textos (Text Mining) é um processo que utiliza técnicas de análise e extração de dados a partir de textos, frases ou apenas palavras. Envolve a aplicação de algoritmos computacionais que processam e identificam informações úteis e implícitas. Suas principais contribuições estão relacionadas à busca de informações específicas em documentos, à análise qualitativa e quantitativa de grandes volumes de textos, e a melhor compreensão do conteúdo disponível em documentos textuais

8.3.2 Mineração de música

As músicas possuem uma natureza variada de elementos, como estilo, ritmo, letra, etc. Para extrair as informações é necessário algoritmos e sistemas sofisticados, que combinam processamento de sinal e técnicas de aprendizado de máquina. A mineração de música é a aplicação de técnicas de mineração de dados para fins de processamento de música, abrangendo classificação de gênero, classificação de emoção/humor, agrupamento de música, anotação automática de tags, impressão digital de áudio, detecção de música cover, bem como mapas e visualização auto-organizados.

8.4 Utilização da técnica de mineração no estilo musical alternativo

8.4.1 Descrição do processo de construção do código de mineração, técnica utilizada, resultados identificados

O computador utilizado para o experimento foi:

- Processador Intel(R) Core (TM) i3-10100F CPU @ 3.60GHz 3.60 GHz.

- 16,0 GB de memória RAM.
- Não foi usado GPU para rodar o experimento.
- Placa mãe PRIME H410M-E, modo UEFI, fabricante ASUSTeK COMPUTER INC.

A coleta de dados foi feita em um site especializado, onde foram extraídas 1200 letras de 6 estilos musicais diferentes. O `https://www.letras.com/` fornece aos usuários letras de músicas, traduções, vídeos com legendas, fotos do artista, discografia. As pessoas têm a opção de filtrar por estilos de músicas de acordo com seu interesse.

O presente trabalho utiliza a linguagem de programação Python para extrair o texto da internet. O código consulta site, pela aba que indica o estilo e busca aqueles mais acessados, pegando 1000 músicas de cada estilo. Essa informação é armazenada em formato XML. O código separa as letras por estilo musical, e identifica o nome da letra e o artista.

Após escolher as métricas de classificação e pegar as músicas no site, foi feito um experimento com as seguintes etapas:

1. Carregar a base de dados: escolha do estilo musical para treinamento do programa.
2. Pré-procesamento: Separa o estilo escolhido que deseja treinar.
3. Tokenização: As palavras do texto de input são tokenizadas de acordo com o BERT multi-lingual.
4. Criação da base de dados.
5. Modelo e treinamento.
6. Execução e avaliação.

8.5 Resultados e Discussões

Os resultados obtidos no experimento deste estudo serão detalhados a seguir.

A Figura 8.1 apresenta o progresso da acurácia conforme o passar das épocas do experimento, o qual indica uma melhora ao longo do tempo, portanto, com mais dados, melhor será o modelo.

A Figura 8.2 apresenta o progresso da perda conforme o passar do tempo, que aponta para melhora com o passar das épocas, ou seja, com mais dados o modelo será melhor.

Os resultados do treinamento foram 0,66% de perda e 99,91% de precisão. Já o resultado do teste foram 22,99% de perda e 94,17% de acurácia.

Sendo a precisão definida como a fração de itens classificados corretamente entre o número total de itens analisados.

A perda é definida como a soma de todos os erros cometidos para cada exemplo.

Como houve ineficiência no teste, como aproximadamente 23% de perda, suspeita-se que o algoritmo possa ter feito overfitting.

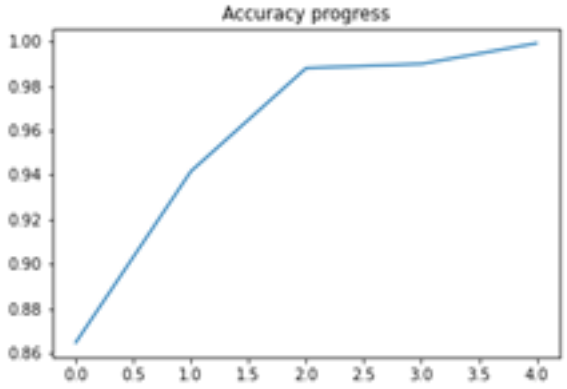


Figura 8.1: Progresso da precisão com as épocas

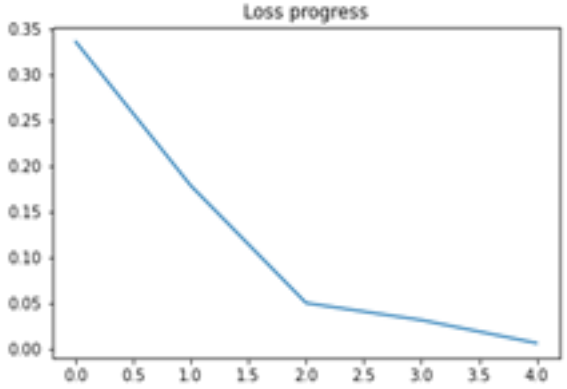


Figura 8.2: Progresso da perda com as épocas

8.6 Considerações Finais

Dentre as limitações para a execução deste estudo, estão a formulação da base de dados, que foi analisada utilizando um quantitativo limitado de letras, sendo necessário um estudo mais sofisticado e robusto para tal, como uma quantidade maior e melhor aprofundamento da revisão da literatura. Por conta disso a estrutura apresentada é ainda um modelo genérico a ser refinado. Outro ponto discutível é em relação ao quantitativo de trabalhos relacionados para análise da literatura.

Para trabalhos futuros, seria interessante lapidar a estrutura do código definida neste estudo, com observações mais abrangentes da literatura, além é claro de realizar a análise de mais estilos musicais com um quantitativo maior de elementos e técnicas disponíveis na literatura, acrescentado uma maior abrangência nas observações com as letras, ritmo e notas musicais.

9

Comparação Automática das Diretrizes Curriculares Nacionais (DCN) e a Classificação Brasileira de Ocupações (CBO)

Eduardo Vieira Marques Pereira do Valle e Herbert Salazar dos Santos
Programa de Engenharia de Sistemas e Computação - COPPE

Resumo

A chegada da 4ª Revolução Industrial trouxe novos desafios ao futuro do trabalho, a medida que praticamente todas as ocupações são impactadas por novas tecnologias. Para agravar este cenário, a COVID-19 acelerou mudanças no mercado de trabalho e gerou altas taxas de desemprego. Neste contexto, fica evidente a importância de as Instituições de Ensino Superior (IES) possuírem ferramentas para avaliar a empregabilidade de seus egressos. Todavia, as IESs brasileiras carecem de informações para análise do contexto do mercado de trabalho no qual estão inseridas. O objetivo deste estudo é comparar as Diretrizes Curriculares Nacionais (DCN) dos cursos de graduação com as ocupações presentes na Classificação Brasileira de Ocupações (CBO), permitindo entender quais vagas de emprego um egresso de um curso está apto a concorrer. Para realizar a comparação, foi utilizado o Modelo de Espaço Vetorial para calcular a similaridade entre as descrições das ocupações da CBO e as DCN. Os resultados apresenta baixa similaridade entre cursos e ocupações, indicando que novas abordagens deverão ser utilizadas para que seja possível uma posterior análise aprofundada da aderência dos

cursos ao mercado de trabalho brasileiro, uma vez que será possível calcular a empregabilidade dos cursos de graduação dentro do contexto em que estão inseridos.

9.1 Introdução

As preocupações com o futuro do trabalho não são novas. Há dois séculos, a ideia de novas tecnologias substituindo a força de trabalho humana fez surgir movimentos como os luditas, onde um grupo de trabalhadores têxteis destruíram máquinas têxteis que lhes tirariam seus empregos (Okolie, Nwosu e Mlanga, 2019). Atualmente, a Quarta Revolução Industrial – também conhecida como Indústria 4.0 – traz novas questões sobre o futuro com o avanço da Inteligência Artificial, biotecnologia e nanotecnologia, acelerando mudanças no trabalho e novas preocupações com o desemprego tecnológico – quando o número de empregos perdidos para a tecnologia é maior do que a capacidade dessa força de trabalho de ser realocada no mercado de trabalho (Y. Lima et al., 2021).

A pandemia do COVID-19 tornou as preocupações com o desemprego mais críticas. As taxas de desemprego durante a COVID-19 foram comparáveis às da Grande Recessão, acelerando as mudanças no mercado de trabalho (Coibion, Gorodnichenko e Weber, 2020). Além disso, cenários previstos para alguns anos – como consultas médicas remotas, ensino remoto e *home office* (Barbosa, Y. Lima, Lyra et al., 2019) (Barbosa, Y. Lima, Mioto et al., 2017) – passaram a ser realidade já em nossos dias (Fraga, 2020).

Nesse contexto, os trabalhadores tendem a buscar mais qualificações, ou ainda requalificações, para aumentar sua empregabilidade individual – ou seja, a capacidade pessoal de obter e manter um emprego (Hillage et al., 1999). No entanto, em tempos de crise as pessoas evitam gastar tempo e dinheiro com qualificações que não tragam impactos imediatos em seus salários (The Economist, 2020).

Portanto, a sinergia entre o conteúdo ensinado nos curso de graduação com as demandas do mercado de trabalho é crucial para a relevância das Instituições de Ensino Superior (IES). Nos Estados Unidos, os alunos de graduação veem o *wage premium* diminuir, ou seja, a diferença entre o salário médio dos graduados versus o salário médio de trabalhadores apenas com Ensino Médio está cada vez menor. Essa queda no *wage premium* fez com que as matrículas no Ensino Superior caíssem 8% nos últimos anos (The Economist, 2020). Este fenômeno não se limita apenas a um país. Um exemplo disso é um estudo feito pelo Institute for Fiscal Studies, estimando que 20% dos estudantes britânicos teriam mais sucesso financeiro se não tivessem concluído um curso de Ensino Superior (Waltmann et al., 2020).

O desencontro entre o que é lecionado na graduação e o que o mercado de trabalho espera de seus trabalhadores faz com que as IES sejam vistas como menos importantes para a educação. No Brasil, por exemplo, se prevê um corte orçamentário de 18% para as universidades e institutos de Ensino Superior (CONIF, 2021). Esses cortes colocam em risco o funcionamento pleno das universidades, podendo ocasionar até mesmo de uma

paralisação total de suas atividades, afetando 25% do estudantes de ensino superior no país, o que representa mais de 2 milhões de pessoas (UFRJ, 2021).

Para serem relevantes e mostrarem sua importância para governos e sociedade, as IES precisam, então, desenvolver estratégia para aumentar a empregabilidade e *wage premium* de seus alunos, uma vez que conseguir melhores empregos é um dos principais motivos para se iniciar um Curso Superior (Gatbonton e Aguinaldo, 2018). Uma das estratégias que podem ser adotadas pelas IES é utilizar informações sobre o mercado de trabalho para apoiar a tomada de decisões. Indicadores focados no cenário atual e futuro das diferentes ocupações permitem que as IES atuem para desenvolver em seus alunos as habilidades necessárias ao mercado de trabalho, fazendo melhorias no portfólio de cursos, currículos dos cursos e qualificação do corpo docente e ajudando os seus egressos a obter, manter ou trocar de emprego caso necessário.

Entretanto, mensurar empregabilidade não é uma tarefa simples (Harvey, 2001). Muitos são os fatores que podem influenciar na construção da empregabilidade, que pode ser vista e avaliada de diferentes pontos de vista (Van Der Heijde e Van Der Heijden, 2006). Além da dificuldade natural de mensurar a empregabilidade, estudos sobre o tema voltados para a realidade da América do Sul são limitados (Jones, Idrovo-Carlier e Rodriguez, 2021), dificultando ainda mais a vida dos gestores na hora de tomar decisões para fazer ajustes na grade curricular de seus cursos de acordo com as demandas do mercado de trabalho.

Considerando o cenário brasileiro, 40% dos jovens graduados não possuem empregos qualificados – empregos que demandam habilidades específicas de profissionais graduados (B. Lima e Gerbelli, 2020). Ao mesmo tempo, metade das indústrias brasileiras tem problemas com falta de mão de obra qualificada (CNI, 2020) e mais de 70% dos recrutadores afirmam que tem problemas para encontrar candidatos com conhecimento mínimo para ocuparem as vagas ofertadas (Easy2Recruit e GazzConecta, 2020). Esses dados mostram que há um problema real de desencontro entre o que é ensinado nos cursos de graduação brasileiros e as demandas do mercado de trabalho, evidenciado pela lacuna entre as qualificações do profissional formado e as qualificações necessárias ao mercado de trabalho.

Todavia, os gestores de IES não possuem ferramentas que possam auxiliar no processo de escolha e aperfeiçoamento dos cursos a fim de propiciar a seus alunos o desenvolvimento contínuo de habilidades de empregabilidade que estejam de acordo com as competências requeridas pelo mercado de trabalho (Santos, 2021). Além disso, o acompanhamento de egressos por parte das IES ainda é limitado, com as poucas soluções disponíveis não tendo uma visão de futuro, tornando inviável a criação de um planejamento para amenizar os impactos que as constantes mudanças no mercado de trabalho podem trazer para a empregabilidade dos egressos de diferentes cursos (Santos, 2021). Portanto, os gestores de IES não possuem informações suficientes para tomarem decisões no presente que os deixem preparados para o futuro.

Dentre as informações importantes que faltam aos gestores para avaliarem a empregabilidade dos cursos de graduação está a relação entre cursos e ocupações (Santos, 2021). Entender quais ocupações alunos de determinado curso de graduação terão as competências necessárias para trabalharem seria uma ferramenta poderosa para auxiliar o ajuste do desencontro entre cursos e ocupações (Santos, 2021). Se tal relação existisse, seria possível, por exemplo, entender o cenário atual e futuro da empregabilidade de futuros egressos, permitindo observar a oferta de vagas de emprego para os cursos de graduação em determinada região e as remunerações possíveis para profissionais da área, permitindo aos gestores avaliarem a viabilidade da criação de novos cursos ou ajustarem a oferta de vagas para novos alunos dentre os cursos já disponíveis, alocando os recursos de forma otimizada para cada curso.

Essa relação também pode ser útil para outros ajustes dentro de uma IES, como a precificação de mensalidades dos cursos de acordo com os ganhos possíveis dos alunos, sendo uma importante ferramenta de marketing para as instituições privadas. Por fim, a criação de uma metodologia para ligar os cursos com ocupações traria a vantagem de ser auto-ajustável às mudanças do mercado de trabalho, permitindo que as IES possam se preparar para novas ocupações que surjam e outras que deixem de existir (Santos, 2021).

Portanto, o objetivo deste trabalho é, utilizando uma metodologia de comparação textual, criar uma ligação entre cursos de graduação brasileiros com ocupações ofertadas no Brasil. Para isso, serão utilizadas as Diretrizes Curriculares Nacionias (DCNs) dos cursos de Ensino Superior do Brasil, que regulamentam o as competências que devem estar presentes no currículo dos cursos de graduação.

Além disso, serão utilizadas as descrições das competências necessárias para profissionais que trabalha em alguma ocupação dentro das famílias ocupacionais presentes na Classificação Brasileira de Ocupações (CBO), que lista as ocupações ofertadas no Brasil. As famílias ocupacionais são grupos de ocupações correlatas que possuem competências similares para que sejam performadas. A fim de avaliar a hipótese e atingir o objetivo do trabalho, utilizamos o modelo vetorial para verificar a semelhança entre as DCNs definidas pelo Ministério da Educação com as famílias de ocupações presentes na CBO.

Este trabalho está dividido em cinco seções. Além desta Introdução, a Seção 9.2 descreve os principais conceitos de empregabilidade e similaridade de textos, enquanto a Seção 9.3 detalha o método do modelo vetorial utilizado nos experimentos. A Seção 9.4 relata os resultados obtidos pelos experimentos e por fim a Seção 9.5 traz as considerações finais, limitações e trabalhos futuros.

9.2 Fundamentação Teórica

Esta seção busca apresentar os conceitos essenciais para a realização deste trabalho. Para que seja possível compreender a relevância da empregabilidade para as IES, se faz necessário entender primeiro o que de fato é empregabilidade, como mensurá-la

e como ela se aplica ao Ensino Superior. Além disso, é importante entender como funciona o conceito de similaridade entre textos, com o propósito de demonstrar como é possível utilizar este conceito para criar a relação entre cursos de graduação e ocupações, permitindo a posterior análise da empregabilidade de cursos de Ensino Superior.

9.2.1 Empregabilidade

Este conceito nasceu por volta de 1955 como sendo o fator determinante para garantir um emprego remunerado no futuro próximo (Thijssen, Van der Heijden e Rocco, 2008), mas começou a ser estudado empiricamente apenas por volta dos anos 90 (Van Der Heijde e Van Der Heijden, 2006) com o crescente interesse pelo tema para pautar políticas públicas, dadas as incertezas sobre as relações de trabalho do futuro, causadas pela ideia de que as longas carreiras e empregos estáveis estavam com os dias contados (Hillage et al., 1999).

A empregabilidade pode ser analisada através de diferentes aspectos (Hillage et al., 1999) (Van Der Heijde e Van Der Heijden, 2006). Para que seja possível mensurá-la, então, é importante antes definir o conceito de empregabilidade uma vez que essa definição influenciará na escolha dos indicadores para mensuração, uma vez que o conceito é aberto à interpretação (Alpek e Tesits, 2020). Uma das definições mais aceitas para empregabilidade é que esta é "a capacidade de obter um emprego inicial, manter o emprego e obter um novo emprego, se necessário" (Hillage et al., 1999), sendo uma definição mais abrangente e que possibilita definir empregabilidade para diversos pontos de vista (Santos, 2021), porém outras definições são encontradas na literatura. Portanto, definir de antemão o conceito de empregabilidade a ser utilizado em um estudo é muito importante para transformar este fenômeno social em algo que se possa mensurar (Harvey, 2001).

A mensuração de empregabilidade pode acontecer em diferentes níveis (Van Der Heijde e Van Der Heijden, 2006). Isso significa que é possível mensurar a empregabilidade de um indivíduo, de uma região, de uma instituição ou de um grupo de pessoas (Alpek e Tesits, 2020). Isso faz com que os fatores que influenciam a empregabilidade variem de acordo com o nível e a definição adotados, podendo estes serem mensurados via mineração de dados ou modelos probabilísticos, com a empregabilidade sendo a agregação de todos esses fatores. Os fatores que influenciam a empregabilidade incluem a idade dos indivíduos, a performance escolar e o desempenho acadêmico, as oportunidades de desenvolvimento providas pelas instituições de ensino, o contexto ao redor da empregabilidade que está sendo avaliada, a capacidade de mobilidade dos indivíduos entre regiões, adaptação a mudanças, flexibilidade, área de estudo, setor de trabalho, gênero, etnia, classe social, habilidades linguísticas, habilidades genéricas como o senso de responsabilidade, liderança e tomada de decisão, e habilidades ocupacionais específicas e relacionadas a área de atuação do profissional, entre outros (Santos, 2021).

A educação superior é capaz de influenciar a empregabilidade dos indivíduos, principalmente os que acabaram recentemente a graduação e tem pouca ou nenhuma experiência profissional, pois quanto maior o tempo de formado, mais as experiências

profissionais influenciarão a empregabilidade do indivíduo em detrimento da graduação (Smith, McKnight e Naylor, 2000). As IES tem, ainda, o papel de prover oportunidades de desenvolvimento de empregabilidade e experiência extracurricular, que contribuem para a empregabilidade individual (Harvey, 2001) (Mishra, Kumar e Gupta, 2016). Por representar um papel tão importante na empregabilidade dos indivíduos, SANTOS defende uma nova abordagem para a empregabilidade, focada no ponto de vista do Ensino Superior e validada com gestores de IES públicas e privadas brasileiras (Santos, 2021), expandindo o conceito de empregabilidade para definir que a empregabilidade de um curso de graduação é "o potencial do curso de formar egressos capazes de obter um primeiro emprego qualificado, se manter neste emprego ou obter um novo emprego qualificado, se necessário."(Santos, 2021). Dentre os fatores que influenciam este conceito de empregabilidade focado em cursos de graduação está o saldo de emprego – diferença entre admitidos e demitidos nas ocupações relacionadas ao curso – a probabilidade de automação, o salário médio e *wage premium* dessas ocupações (Santos, 2021). Estes fatores que influenciam a empregabilidade dos cursos mostram a importância da relação entre os cursos de graduação e as ocupações, uma vez que sem essa ligação não é possível avaliar a empregabilidade de cursos de graduação.

9.2.2 Similaridade entre Textos

Similaridade textual trata em analisar o quanto uma palavra, sentença ou documento é semelhante a outro, e medir esse aspecto entre textos é relevante em tarefas como recuperação de informações, agrupamento de documentos, tradução automática, entre outras (Gomaa, Fahmy et al., 2013). Na literatura é apresentado em duas maneiras distintas para verificar a correspondência entre textos(Gomaa, Fahmy et al., 2013), elas seriam a similaridade lexical, que explora a presença da mesma sequencia de caracteres entre palavras presentes nos textos; e similaridade semântica, a qual investiga a semelhança entre significados, ideias e contextos entre textos.

Neste trabalho há o foco maior na similaridade lexical, em virtude disso houve o estudo de abordagens e soluções voltadas para este aspecto entre elas estariam o abordagem booleano e a distancia do cosseno entre vetores de palavras. A solução booleana consiste em avaliar a similaridade textual entre dois textos pelo uso da lógica booleana, averiguando a proximidade através do número de palavras iguais presentes em cada conjunto de termos dos documentos(Zobel e Moffat, 2006).

Já a abordagem vetorial equivale a representar o documento como um vetor de pesos de cada termo presente no texto, existindo várias formas distintas para calcular esses pesos, depois da geração dos vetores de cada documento é avaliado a similaridade entre eles através de um algoritmo(Berry, Drmac e Jessup, 1999). Alguns desses algoritmos seriam o calculo do cosseno entre os vetores e a distância euclidiana entre eles. Essa solução como é apresentado em (Dubin, 2004) apresenta dificuldade de delimitar quando surgiu, sendo atribuída como modelo do espaço vetorial no trabalho (Salton, 1989) em 1989, no entanto suas ideias existem em trabalhos anteriores mas com nomenclaturas diferentes.

9.3 Método

O experimento utiliza o modelo vetorial para avaliar a semelhança entre os documentos que descrevem as DCNs e as descrições de competências esperadas para os profissionais que atuam em alguma ocupação dentro das famílias de ocupação do CBO. Os vetores das DCNs apresentam o peso calculado pela frequência do termo–inverso da frequência nos documentos(tf-idf) de cada termo presente no documento, enquanto os das famílias de ocupações teriam o mesmo tamanho dos vetores descritos anteriormente, já que descrevem os mesmos termos, e denotariam apenas um valor booleano da presença do termo no documento. A fim de determinar a semelhança entre os vetores é calculado o cosseno entre eles.

Ademais, para diminuir a quantidade de termos que seriam utilizados nos vetores do experimento, só foram selecionados os termos nos documentos que não estavam presentes nos conjuntos de stop-words da língua portuguesa, além disso foi realizado a poda desses termos para uma versão mais simplificada para reduzir redundâncias de escritas distintas para termos com o mesmo significado.

Para que fosse possível realizar a comparação de textos, alguns passos prévios foram necessários. Foram utilizados as colunas de descrição das competências recomendadas famílias ocupacionais presente na CBO, base mantida pelo Ministério do Trabalho. Todavia, esta base de dados possui erros de codificação na fonte. Três diferentes ferramentas para tentar detectar a codificação e corrigir os erros foram utilizadas mas não tiveram sucesso. Por fim, foi necessário fazer um trabalho manual para corrigir os erros apresentados nas tabelas. O problema desta abordagem é que o arquivo pode apresentar erros de codificação que não foram identificados.

Já em relação às DCNs, não há uma base oficial aberta para consultar as habilidades esperadas em egressos dos cursos. Portanto, para realizar o experimento, foi necessário a construção de um crawler – robô de varredura de páginas *web* – para fazer o *download* dos arquivos *PDF* das DCNs no site oficial do Ministério do Trabalho (Ministério da Educação, 2022). Este site possui as DCNs de grupos de cursos de graduação brasileiros, além de documentos para aprovação e modificação das diretrizes durante os anos. Para simplificar o processo, todos os documentos referentes a um grupo de cursos foram agregados através de um analisador construído para extrair o texto dos arquivos *PDF*. Com o texto extraído, estes foram convertidos para os formatos *CSV* e *JSON*, a fim de serem utilizados pelo algoritmo de comparação textual.

9.4 Resultados

Para avaliação dos resultados escolhemos duas famílias ocupacionais – **Engenheiros civis e afins** – que possui as ocupações Engenheiro Civil de aeroportos, edificações, estruturas metálicas, ferrovias e metrovias, geotécnica, hidrologia, hidráulica, pontes e viadutos, portos e vias navegáveis, rodovias, saneamento, túneis, transportes e trân-

sito, e Tecnólogo em construção civil – e **Analistas de tecnologia da informação** – que possui as ocupações Analista de desenvolvimento de sistemas, Analista de redes e de comunicação de dados, Analista de sistemas de automação e Analista de suporte computacional. Essas famílias possuem diferentes abordagens quando observamos as ocupações. A família de ocupações **Engenheiros civis e afins** possui trabalhadores com, no mínimo, Ensino Superior Completo, considerando todo o território brasileiro. Já a família de ocupações **Analistas de tecnologia da informação** possui 30% dos trabalhadores com nível de escolaridade menor do que Ensino Superior (Classificação Brasileira de Ocupações, 2022). Isso possibilitará como o algoritmo se comportará para casos onde profissões são ocupadas majoritariamente por graduados e profissões que possuem trabalhadores de diferentes níveis de escolaridade.

Para a família ocupacional **Analistas de tecnologia da informação**, seis grupos de cursos de graduação ficaram empatados na liderança como os cursos mais adequados para esta família, com cosseno de 0,121426. Estes cursos foram: Sistemas de Informação, Engenharia de Software, Engenharia de Computação, Computação (Licenciatura), Computação, Ciência da Computação. Após estes, o curso identificado como mais indicado para essa família de ocupações é Administração Pública, com cosseno de 0,090039.

Para a família ocupacional **Engenharia civil e afins**, o grupo de cursos de graduação com maior cosseno foi Engenharia (0,125958), seguido por Tecnológicos Cursos Superiores (0,075183), e Sistemas de Informação (0,059893).

Os resultados do experimento apresentam resultados satisfatórios, uma vez que, em uma análise inicial, as correlações entre cursos e ocupações para os dois casos apresentados estão condizentes. Entretanto, há um grau baixo de similaridade entre as diretrizes dos grupos de cursos e as descrições das famílias de ocupações. Isso pode ter ocorrido devido uma relação baixa entre o que é proposto e desenvolvido em faculdades com o que é exigido no mercado de trabalho; ou em virtude dos problemas apresentados na coleta das bases de dados utilizados no experimento descritos na seção 9.3.

9.5 Conclusão

O mercado de trabalho sempre esteve em evolução, porém a chegada da Quarta Revolução Industrial impactou ainda mais as ocupações, com o rápido avanços de tecnologias como Inteligência Artificial. Enquanto carreiras centenárias estão desaparecendo, novas ocupações vão surgindo, e praticamente todas vem sendo transformadas. Além disso, a pandemia de COVID-19 vem transformando a maneira como empresas, instituições, governos e indivíduos trabalham, trazendo dentro de pouco tempo mudanças que eram prospectadas para um futuro a médio/longo prazo.

Neste contexto, governos, instituições e cientistas estão procurando soluções para esse problema que cada dia fica mais evidente. No Brasil, assim como em outros países da América Latina, as preocupações sobre o emprego focam no curto prazo e o futuro tende a ser deixado de lado. Embora úteis, o foco no curto prazo deixa uma pergunta: como

nos prepararmos para esse futuro? Responder a essa pergunta não é uma tarefa trivial, uma vez que abre um leque de novos questionamentos.

Gestores de Instituições de Ensino Superior de todo o Brasil demonstraram a falta de ferramental para auxiliar na avaliação dos cursos frente as demandas do mercado de trabalho. Estudos nessa área são limitados na América do Sul, deixando uma enorme lacuna a ser explorada. Neste contexto, em que 40% dos jovens brasileiros não possuem emprego qualificado ao mesmo tempo em que há falta de profissionais qualificados para ocuparem outros postos de trabalho, fica evidente a necessidade de estudos que sejam capazes de impulsionar a empregabilidade dos brasileiros.

Portanto, focado nas tomadas de decisão por parte das IES, este trabalho se propôs a criar uma conexão entre cursos de graduação e ocupações brasileiras, a fim de possibilitar a avaliação dos fatores que influenciam na empregabilidade de cursos de graduação. Para isso, foi utilizado uma metodologia de comparação textual através do modelo vetorial, permitindo comparar as Diretrizes Nacionais Curriculares, responsáveis por determinar as habilidades que cada curso de graduação brasileiro deve transmitir aos seus alunos, com as descrições de competências necessárias por um profissional que atua em alguma ocupação dentro das famílias ocupacionais da Classificação Brasileira de Ocupações.

As observações encontradas nos experimentos realizados mostram resultados promissores, embora o subconjunto de teste seja limitado e os resultados apresentem um grau de similaridade baixo entre as definições de grupos de cursos e das famílias de ocupações. Dentre os fatores que podem ter contribuído para esse baixo grau de similaridades estão desde o desencontro entre habilidades transmitidas e competências desejáveis, até os erros de codificação encontrados ao analisar a base da CBO. Outro fator que pode ter contribuído para o baixo grau de similaridade foi a decisão de juntar documentos que aprovam ou modificam as DCNs, uma vez que são documentos que apresentam termos que não estão diretamente relacionados com as habilidades desejáveis em cada curso de graduação, sendo uma limitação deste trabalho. O uso de apenas uma abordagem para medir a similaridade dos documentos também representa uma limitação. Como não possuímos um método de avaliação automática, escolhemos um subgrupo de famílias ocupacionais para fazer a análise. Como trabalho futuro, devemos criar um método de avaliação que torne menos custoso a avaliação para todas as famílias de ocupações, permitindo a avaliação de mais casos. Além disso, será necessário realizar o experimento através de várias abordagens de similaridade textual para delimitar melhor a relação entre os documentos, além de analisar a relação dos cursos especificamente as ocupações do mercado de trabalho ao invés de grupos de cursos. Para isso, será necessário melhorar o *crawler* a fim de considerar apenas os documentos que representam as habilidades desejáveis de cada curso. Por fim, um Delphi com especialistas de cada uma das áreas analisadas deve ser realizado futuramente para validação dos resultados obtidos.

10

Extração de Títulos de Artigos Científicos a Partir de Resumos

Rodrigo Peregrino e Victor Hugo Ventura
Programa de Engenharia de Sistemas e Computação - COPPE

Resumo

Nosso objetivo neste trabalho é utilizar o modelo de linguagem T5 para extrair títulos de artigos científicos utilizando seus resumos.

10.1 Introdução

Modelos de linguagem são essencialmente distribuições de probabilidade sobre uma sequência de palavras ou *tokens*: para cada sequência $w_1, \ldots, w_n$ (que também denotaremos por $w_{1:n}$), assinalamos uma probabilidade $P(w_1, \ldots, w_n)$. Estes modelos são treinados em vastos conjuntos de texto (por exemplo, todos os livros em uma biblioteca virtual, todos os artigos da Wikipédia, etc) para gerar tais probabilidades. Ainda assim, como o número de frases que podem ser geradas em uma linguagem é infinito, certamente teremos sentenças nunca antes vistas durante o treinamento para as quais temos que assinalar probabilidades, portanto soluções ingênuas como dividir número de vezes que a sequência foi vista pelo número total de sequências vistas não funcionaria.

Por este motivo, diversas suposições adicionais que podem ser colocadas sobre a distribuição de probabilidade inicial foram introduzidas. Uma delas é o modelo n-grama, a ideia por trás desta técnica é utilizar um pequeno conjunto de palavras como contexto e tentar adivinhar a próxima palavra. Assim, uma grande sequência de palavras pode ser divididas em pequenas sequências de bigramas ou trigramas, e dado um corpus linguís-

tico grande o suficiente, essas pequenas sequências são muito mais prováveis de aparecer do que a sequência original.

Como exemplo, se queremos estimar a probabilidade da palavra "é" vir depois de "Meu nome", podemos simplesmente fazer a contagem no conjunto de treinamento de quantas vezes vimos a palavra "é" depois de "Meu nome":

$$P(\text{é} \mid \text{Meu nome}) = \frac{C(\text{Meu nome é})}{C(\text{Meu nome}).}$$

Para uma sentença qualquer, podemos expandir a probabilidade pela regra da cadeia:

$$\begin{aligned} P(w_1, \ldots, w_n) &= P(w_1)P(w_1 \mid w_2) \ldots P(w_n \mid w_{1:n-1}) \\ &= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1}). \end{aligned}$$

O modelo n-grama aproxima a probabilidade de uma palavra dadas todas as anteriores na sequência pela probabilidade dadas apenas as $n-1$ palavras anteriores. Por exemplo, para o modelo bigrama temos a aproximação

$$P(w_j \mid w_{1:j-1}) \approx P(w_j \mid w_{j-1}),$$

logo

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1}).$$

O problema destes modelos é não conseguir capturar contexto em longas distâncias, para isso precisamos de modelos mais poderosos baseados em redes neurais. Antes de introduzir estes modelos mais complexos, vamos apresentar o problema de sumarização de texto.

10.2 Problema

O problema que nos propomos a estudar é o de sumarização de texto. Sumarização significa essencialmente criar um subconjunto das palavras ou frases usadas em um documento que representa o mais fielmente possível a informação contida no documento original. Mais especificamente, nosso problema é a geração de títulos de artigos científicos a partir de seus abstratos, ou seja, que o modelo consiga, dado o abstrato de um artigo, gerar um título apropriado que seria aceito por revistas científicas.

Geralmente há duas abordagens que são reconhecidas como sumarização, que são extração e abstração. Na primeira abordagem frases e palavras são extraídas do documento sem modificação e tentam apresentar a informação mais relevante para do texto, como *tags*, índices, palavras-chave, ou mesmo extrair certos parágrafos ou *headings*. Na abordagem abstrativa, uma representação semântica do texto deve ser construída para então

condensar as informações documento mas mantendo o significado. Como o abstrato de um artigo já é uma espécie de sumário para o artigo inteiro, estamos essencialmente tentando extrair o sumário do sumário para gerar o título do artigo.

10.3 Transformers

A arquitetura de *transformers* foi introduzida em (Vaswani et al., 2017) e rapidamente tomou o lugar das redes neurais recorrentes e da arquitetura LSTM (*Long short-term memory*, um tipo de rede recorrente) como o modelo mais usado para problemas de processamento de linguagem natural. Uma de suas principais vantagens sobre as redes recorrentes é a capacidade de processamento em paralelo, que é atingido utilizando o mecanismo de *self-attention*, em oposição ao mecanismo sequencial de recorrência (Jurafsky e Martin, 2009) em que a ideia geral é que o *output* no tempo $t-1$ é colocado novamente na rede como *input* no tempo t, ou seja, temos que esperar a resolução dos cálculos do passo anterior para seguir com o treinamento da rede.

10.3.1 Embeddings

Modelos de linguagem baseados em redes neurais utilizam as chamadas *word embeddings* para representar palavras de forma mais eficiente. *Embeddings* são vetores de números reais que representam cada uma das palavras no vocabulário do conjunto de dados. Os vetores são utilizados para modelar o significado das palavras e esperamos que vetores mais próximos representem palavras relativamente similares. Estes vetores são então passados para as redes neurais que caracterizam os modelos de linguagem. *Transformers* como BERT e ELMo utilizam *embeddings* contextuais, que mudam conforme o contexto da palavra e são aprendidos durante o treino da rede, em oposição a *embeddings* estáticos como Word2Vec e GloVe (Devlin et al., 2019).

10.3.2 Self-Attention

A camada de *self-attention* de um *transformer* começa com uma sequência de vetores $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ e retorna uma sequência $(\mathbf{y}_1, \ldots, \mathbf{y}_n)$ do mesmo tamanho. Para cada vetor no *output*, a camada tem acesso à todos os *inputs* com índice menor ou igual, por exemplo para $\mathbf{y}_3$, a camada utiliza $\mathbf{x}_1, \mathbf{x}_2$ e $\mathbf{x}_3$. Além disso, todos os cálculos de cada vetor no *output* são realizados de independentemente dos cálculos dos outros *outputs*, fazendo esta parte da rede facilmente paralelizável.

O mecanismo de atenção é uma técnica que tenta aumentar a influência de certos dados que consideramos importantes e ao mesmo tempo tenta diminuir a influência de outros considerados menos relevantes. A auto-atenção ou *self-attenstion* compara os elementos da sequência de *input* e assim decide quais elementos são mais relevantes. A forma de

comparação adotada entre estes elementos é o produto interno, que é conceitualmente simples e computacionalmente barato. Este valor é interpretado como uma pontuação ou *score* que nos diz a similaridade entre os vetores:

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j.$$

Estas pontuações são então normalizadas e chegamos a um conjunto de vetores de probabilidade α_{ij} (o que significa que $0 \leq \alpha_{ij} \leq 1$ e $\sum_j \alpha_{ij} = 1$ para todo i):

$$\begin{aligned}\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \\ &= \frac{e^{\text{score}(\mathbf{x}_i, \mathbf{x}_j)}}{\sum_{k=1}^{i} e^{\text{score}(\mathbf{x}_i, \mathbf{x}_k)}}, \quad \forall j \leq i.\end{aligned}$$

Estes α_{ij} dão a relevância de cada *input*, ou seja, quanto maior α_{ij}, maior a relevância de $\mathbf{x}_j$ para o *output* $\mathbf{y}_i$,

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{x}_j.$$

Esta é a ideia básica da noção de atenção. Agora, no artigo original a função de atenção é descrita como "um mapeamento entre uma consulta (*query*) e um conjunto de pares de valores-chave para uma saída, onde a consulta, as chaves, os valores e a saída são todos vetores." (Vaswani et al., 2017) Cada um destes conceitos, consulta, chave e valor, tem uma matriz associada ($\mathbf{W^Q}$, $\mathbf{W^K}$ e $\mathbf{W^V}$, respectivamente) que desempenha os papéis acima descritos porém de uma forma refinada.

Definindo os vetores de consulta, chave e valor

$$\mathbf{q}_i = \mathbf{W^Q}\mathbf{x}_i, \mathbf{k}_i = \mathbf{W^K}\mathbf{x}_i, \mathbf{v}_i = \mathbf{W^V}\mathbf{x}_i.$$

O valor da pontuação agora será dado pelo produto interno entre o vetor de consulta de $\mathbf{x}_i$, o foco da atenção, e o vetor chave de $\mathbf{x}_j$, o contexto. Além disso, para evitar problemas numéricos, dividimos o valor da pontuação pela raiz quadrada da dimensão dos vetores de consulta e chave, que chamamos de d_k,

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}.$$

A definição dos α_{ij} permanece a mesma. Em contraste, para chegarmos no *output* final, fazemos o somatório anterior mas agora utilizando o vetor de valores dos $\mathbf{x}_j$:

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j.$$

Como dito anteriormente, uma das grandes vantagens deste método é que, como cada saída é calculada de forma independente das outras saídas, podemos facilmente paralelizar o algoritmo. Fazendo N vetores de entrada como linhas da matriz $\mathbf{X} \in \mathbb{R}^{N \times d}$, todos os cálculos anteriores podem ser expressos por multiplicações de matrizes. De fato, definindo

$$\mathbf{Q} = \mathbf{X}\mathbf{W^Q}, \mathbf{K} = \mathbf{X}\mathbf{W^K}, \mathbf{V} = \mathbf{X}\mathbf{W^V},$$

a função de *self-attention* pode ser expressa de forma sucinta

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}.$$

10.3.3 Transformers Unidirecionais e Bidirecionais

Uma ressalva sobre esta fórmula é que no caso unidirecional, para a i-ésima linha, queremos apenas os índices j tais que $j \leq i$, portanto temos que substituir as entradas no triângulo superior da matriz $\mathbf{QK}^T$ por menos infinito para que depois de aplicada a exponencial estas entradas se anulem. No caso bidirecional, podemos utilizar a fórmula anterior sem modificações.

10.3.4 O Bloco Transformer

Agora que descrevemos a camada de *self-attention*, podemos falar sobre a arquitetura padrão de um bloco *tranformer*. O bloco começa com a camada de *self-attention* e e é seguido por uma camada de normalização com uma conexão residual com o *input*. Após isso, temos uma camada *feedforward*, ou seja, completamente conectada, seguida de mais uma normalização com conexão residual com o *input* da camada *feedforward*. O resultado desta normalização é a saída do bloco. Podemos descrever o bloco pelas seguintes equações:

$$\begin{aligned}\mathbf{z} &= \text{Norm}(\mathbf{x} + \text{Attention}(\mathbf{x})),\\ \mathbf{y} &= \text{Norm}(\mathbf{z} + \text{FeedForward}(\mathbf{z})).\end{aligned}$$

A conexão residual significa simplesmente que somamos o *input* da camada ao seu *output* antes de passar para a próxima, pular camadas melhora o aprendizado e dá acesso a diferentes representações do *input* para camadas posteriores (He et al., 2016). Normalização significa que calculamos a média μ e o desvio padrão σ do vetor e então o normalizamos para ter uma média zero e desvio padrão 1:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma}.$$

Além disso, há dois parâmetros de aprendizado γ e β,

$$\text{Norm}(\hat{\mathbf{x}}) = \gamma\hat{\mathbf{x}} + \beta.$$

10.3.5 Multihead Attention

A ideia por trás deste mecanismo é que as palavras em uma frase podem se relacionar de diversas formas diferentes, sendo muito difícil para um bloco de atenção conseguir aprender todas as relações e possivelmente sendo dominado por um ou outro sentido. Então, separando a função de atenção em várias partes não comunicantes, cada parte poderia aprender uma relação diferente. Assim, cada *head* ou cabeça i deverá ter suas próprias matrizes de consulta, chave e valor, $\mathbf{W}_i^{\mathbf{Q}}$, $\mathbf{W}_i^{\mathbf{K}}$ e $\mathbf{W}_i^{\mathbf{V}}$, respectivamente. Os vetores de consulta e chave tem dimensão d_k e os vetores de valor tem dimensão d_v, dessa forma $\mathbf{W}_i^{\mathbf{Q}} \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}_i^{\mathbf{K}} \in \mathbb{R}^{d\times d_k}$ e $\mathbf{W}_i^{\mathbf{V}} \in \mathbb{R}^{d\times d_v}$ para cada cabeça. Se temos h

cabeças, ao final da execução da função de atenção temos h matrizes $N \times d_v$, para voltar à dimensão de saída correta utilizamos uma matriz de projeção $\mathbf{W^O} \in \mathbb{R}^{hd_v \times d}$.

$$\begin{aligned}
\mathbf{Q}_i &= \mathbf{X}\mathbf{W}_i^{\mathbf{Q}}, \\
\mathbf{K}_i &= \mathbf{X}\mathbf{W}_i^{\mathbf{K}}, \\
\mathbf{V}_i &= \mathbf{X}\mathbf{W}_i^{\mathbf{V}}, \\
\text{head}_i &= \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i), \\
\text{MultiHead}(\mathbf{X}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W^O}.
\end{aligned}$$

Podemos substituir a camada de *self-attention* simples pela nova camada de *multihead attention* no bloco *transformer* e manter o resto do bloco igual (Jurafsky e Martin, 2009).

10.3.6 Positional Embeddings

Ao contrário das redes neurais recorrentes, o mecanismo de *self-attention* é independente da ordem das palavras, portanto esta informação deve ser codificada de alguma outra forma. *Transformers* utilizam *embeddings* posicionais das palavras para levar em conta essa informação. Para uma sequência de *tokens*, cada *token* é associado a um índice e estes índices são codificados por *embeddings*, da mesma forma que os *tokens*. A partir daí, ambas *embeddings* são somadas para chegar em uma *embedding* composta, e esta então será o *input* para a rede neural.

10.4 Encoder-Decoder

O modelo *encode-decoder* é um tipo de arquitetura de redes neurais que pode ser usado tanto com redes recorrentes como com *transformers*. Esta estrutura é composta de três componentes:

- O *encoder*, que toma uma sequência $\mathbf{x}_n$ e gera uma sequência de representações intermediárias $\mathbf{h}_n^e$, o *encoder* é bidirecional;
- O vetor de contexto $\mathbf{c}$, que é uma função dos estados $\mathbf{h}_n$;
- O *decoder*, que toma $\mathbf{c}$ como *input* e gera uma sequência de estados ocultos $\mathbf{h}_m^d$, que pode então ser transformada nos *outputs* $\mathbf{y}_m$, o *decoder* é unidirecional.

10.4.1 Mecanismo de Atenção no Encoder-Decoder

Esta é uma variante da ideia geral de atenção aplicada ao modelo de *encoder-decoder* com o intuito de passar informações sobre todos os estados ocultos do *encoder* para o *decoder*. Temos primeiro que o vetor de contexto $\mathbf{c} = f(\mathbf{h}_1^e, \ldots, \mathbf{h}_n^e)$ é função dos estados ocultos do *encoder*. O problema é que como o tamanho dos *inputs* varia, não podemos passar todos os estados ocultos codificados diretamente para o *decoder*. A ideia então é

criar um vetor de comprimento fixo fazendo uma soma ponderada de todos os estados ocultos do *encoder*, assim, o vetor de contexto $\mathbf{c}_i$ será atualizado a cada passo levando em conta todos estes estados. O estado oculto do *decoder* então será função deste contexto, do estado anterior e do *output* anterior do *decoder*,

$$\mathbf{h}_i^d = g(\hat{\mathbf{y}}_{\mathbf{i-1}}, \mathbf{h}_{i-1}^d, \mathbf{c}_i).$$

Analogamente ao caso do *self-attention*, para calcular o contexto precisamos dar uma pontuação para a relevância de cada estado do *encoder* para o estado atual do *decoder* ($\mathbf{h}_{i-1}^d$, pois ainda não temos o estado $\mathbf{h}_i^d$, que está em função do contexto $\mathbf{c}_i$),

$$\text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e) = \mathbf{h}_{i-1}^d \cdot \mathbf{h}_j^e.$$

Então normalizamos para chegar aos vetores de probabilidades α_{ij},

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e)),$$

onde a soma é sobre todos os estados ocultos do *encoder*. Então, podemos calcular o contexto no passo i como a média ponderada dos estados ocultos do *encoder*,

$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j^e.$$

10.5 O Modelo T5

Este modelo é um *transformer* que utiliza a arquitetura *encoder-decoder*, sua implementação segue a forma do *transformer* original proposto em (Vaswani et al., 2017). A sequência inicial de *tokens* é passada para os *embeddings* (são utilizados *embeddings* posicionais) que são então passados para o *encoder*. O *encoder* é composto de vários blocos *transformer* como descritos acima, com a exceção de que na camada de normalização não há termo de viés e é aplicado *dropout* na camada *feedforward*.

O *decoder* tem uma estrutura parecida com a do *encoder*, mas há um mecanismo de atenção logo após cada camada de *self-attention* que atende o *output* do *encoder*. (Raffel et al., 2020) O *output* do último bloco do *decoder* é passado a uma camada densa e depois a uma função softmax. Todos os mecanismos de atenção do modelo são policefálicos (*multihead*).

10.6 A base de dados

A base de dados utilizada neste trabalho contém aproximadamente 269 mil trabalhos científicos, desde 1881 até 2022, e foi extraída da base de dados Scopus, da Elsevier, em 19 de Janeiro de 2022, sendo selecionados todos os documentos em inglês que contém o

termo "*energy storage*"em seu título, *abstract* ou palavras-chave. Todos os documentos tiveram o *abstract* e o título *tokenizados*, sendo removidos todos cujo o *abstract* fosse maior que 512 *tokens* ou o título fosse maior que 64 *tokens*. Após tal filtragem restaram aproximadamente 260 mil documentos (96,49%), que então foram separados da seguinte maneira: 80% no conjunto de treino, 10% no conjunto de validação e 10% no conjunto de teste.

10.7 Experimentos

Todos os experimentos aqui apresentados foram rodados em uma máquina local com as seguintes configurações: AMD Ryzen 5 3600, memória de 32 GB DDR4, NVIDIA GeForce GTX 1050 Ti 4 GB, e armazenamento SSD M.2 2280. O sistema operacional da máquina é o Microsoft Windows 11 e foi configurado um *container* Docker com WSL2 e sistema operacional Ubuntu 18.04.6 LTS. Mais informações sobre a configuração do ambiente de execução podem ser encontradas no *Dockerfile* disponibilizado no GitHub.

Os hiperparâmetros utilizados em todos os experimentos foram os padrões da função *Seq2SeqTrainingArguments* da biblioteca *transformers*, com exceção do tamanho de lote, que foi utilizado 4, e a regularização, que foi utilizado 0,01.

No primeiro experimento foi realizado *fine-tuning* com toda a base de treino. Inicialmente o objetivo era executar 5 épocas, entretanto, devido as limitações de tempo e da máquina disponível, foram realizadas 2,08 épocas, o que levou aproximadamente 44 horas. Foi estimado que para completar as 5 épocas seriam necessárias mais 66 horas.

O segundo experimento foi realizado a partir do resultado obtido no primeiro e foram executadas 100 épocas adicionais, reduzindo a base de treino a 1% da base original. Este experimento levou aprozimadamente 12 horas para ser concluído e o objetivo foi verificar se o modelo seria capaz de atingir níveis de precisão satisfatórios no conjunto de treino e estimar quantas épocas seriam necessárias para atingir tal nível, sem se importar com *overfitting*.

Os autores do T5 alegam que uma mesma instância do modelo é capaz de aprender diferentes tarefas de maneira simultânea, como traduzir para diferentes idiomas, bastando ser informado no início da entrada qual tarefa deve ser realizada. Deste modo, foi realizado um terceiro experimento onde ao invés de utilizarmos apenas o *abstract* como entrada, foi acrescentado no início a informação da fonte de publicação com objetivo de verificar se o modelo consegue realizar sugestões de título mais apropriadas de acordo com a fonte da publicação desejada. Para este exemplo foi realizado o treinamento com 10% da base de dados por 5 épocas, o que levou aproximadamente 12 horas.

10.8 Resultados

Nesta seção serão apresentados os resultados obtidos com os experimentos realizados. Para avaliar os resultados serão apresentados o valor da função de perda no conjunto de treino e de teste, assim como a métrica *Rouge Score* no conjunto de teste, que avalia o F1-score entre os n-gramas e sequências dos títulos gerados.

As Tabelas 10.1–10.3 mostram os resultados obtidos durante o treinamento de cada um dos experimentos realizados. No primeiro experimento é possível notar que o treinamento estava sendo efetivo porém lento. Nas primeira épocas do segundo experimento foram obtidos resultados melhores que o primeiro experimento no conjunto de treino e validação mas rapidamente o modelo se tornou sobreajustado. No terceiro experimento a evolução do treinamento foi mais rápida que no primeiro embora o modelo tenha atingido os mesmo resultados, um treinamento mais longo parece se fazer necessário nos experimentos 1 e 3.

Tabela 10.1: Evolução do treinamnto do primeiro experimento

Epoch	Tr. Loss	Val. Loss	Rouge1	Rouge2	Rougel	Rougels
0,19	2,40	2,12	41,36	22,24	36,40	36,42
0,38	2,24	2,06	41,61	22,98	37,49	37,53
0,58	2,15	2,01	42,53	23,05	37,43	37,46
0,77	2,14	1,98	42,90	23,27	37,90	37,93
0,96	2,32	1,96	43,07	23,43	38,11	38,14
1,15	2,01	1,94	43,18	23,63	38,19	38,21
1,34	2,00	1,92	43,35	23,64	38,32	38,35
1,53	1,98	1,91	43,40	23,63	38,32	38,35
1,73	2,12	1,90	43,40	23,70	38,34	38,37
1,92	1,94	1,89	43,63	23,84	38,50	38,53

Em seguida, a Tabela 10.4 mostra a comparação entre o título gerado pelos diferentes experimentos assim como pelo modelo sem *fine-tunning* e o título original. Ambos exemplos presentes nesta tabela estão no conjunto de treino do segundo experimento e fora do conjunto de treino do primeiro e terceiro. É possível notar que o segundo experimento gera títulos muito semelhantes ao verdadeiro, sendo idêntico em um dos casos, enquanto o primeiro e o terceiro experimento geram títulos semelhantes entre si. Em todos os casos o modelo sem *fine-tunning* falhou em gerar uma frase válida enquanto que os modelos treinados geraram títulos no aceitáveis.

Em seguida, a Tabela 10.5 mostra comparação semelhante a da anterior porém com ambos exemplos que não estavam presentes no treinamento de qualquer experimento. É possível notar que o segundo experimento deixou de gerar títulos muito semelhantes ao verdadeiro porém conseguiu no segundo exemplo um resultado satisfatório. O primeiro e o terceiro experimento geram títulos idênticos entre si no segundo exemplo porém no

Tabela 10.2: Evolução do treinamnto do segundo experimento

Epoch	Tr. Loss	Val. Loss	Rouge1	Rouge2	Rougel	Rougels
4	1,70	1,89	44,58	24,54	39,49	39,46
8	1,43	1,95	43,72	23,44	38,43	38,50
12	1,31	2,04	43,87	23,45	38,29	38,33
16	1,15	2,12	43,88	23,02	37,94	37,96
20	1,11	2,21	44,20	23,75	38,20	38,16
24	0,97	2,30	43,60	22,63	37,32	37,34
28	0,90	2,37	43,41	22,59	37,55	37,49
32	0,80	2,44	43,10	22,69	37,25	37,25
36	0,76	2,52	43,13	22,37	37,40	37,42
40	0,69	2,59	42,77	22,15	36,85	36,81
44	0,66	2,65	42,38	21,80	36,48	36,56
48	0,56	2,72	42,09	20,92	36,03	36,04
52	0,58	2,78	42,85	21,67	36,69	36,67
56	0,46	2,82	42,64	21,25	36,36	36,37
60	0,44	2,88	41,55	21,00	35,68	35,65
64	0,43	2,92	41,85	21,51	35,88	35,89
68	0,44	2,96	41,51	21,12	35,77	35,76
72	0,43	2,99	41,69	21,46	35,78	35,78
76	0,42	3,03	41,80	21,03	35,49	35,54
80	0,37	3,06	41,40	21,23	35,35	35,40
84	0,33	3,08	41,18	21,21	35,17	35,17
88	0,37	3,08	41,29	21,18	35,31	35,35
92	0,31	3,11	41,88	21,38	35,59	35,53
96	0,35	3,11	41,19	21,14	35,12	35,19
100	0,32	3,12	41,49	21,27	35,26	35,32

Tabela 10.3: Evolução do treinamnto do terceiro experimento

Epoch	Tr. Loss	Val. Loss	Rouge1	Rouge2	Rougel	Rougels
1	2.39	2.23	41.33	21.98	36.34	36.39
2	2.30	2.16	41.84	22.48	36.94	36.98
3	2.23	2.14	41.73	22.27	36.88	36.91
4	2.15	2.12	42.09	22.55	37.06	37.11
5	2.11	2.12	42.11	22.63	37.26	37.30

primeiro o resultado obtido foi bem diferente, sendo que o terceiro experimento falhou em gerar um título satisfatório.

A Tabela 10.6 mostra alguns exemplos gerados variando apenas a fonte da publicação no modelo treinado com o terceiro experimento, assim como o título original (primeira

Tabela 10.4: Comparação dos títulos gerados dentro do conjunto de treino do segundo experimento

Experiment	**Title**
True Title	Project to reduce greenhouse gas emissions
t5-small	, Suncor Energy, Statoil, and Texaco will participate in the
1st	CO2 capture and geologic storage project
2nd	Project to reduce greenhouse gas emissions
3rd	CO2 storage: A $20 million CO2 capture project
True Title	Versatile input converter for battery charging in renewable energy power systems
t5-small	, the flyback converter for battery charging. The flyback converter is a simple
1st	A flyback converter for battery charging
2nd	Converter for battery charging in renewable energy power systems
3rd	A flyback converter for battery charging

linha) e a quantidade de vezes que este título foi gerado. No total foram 5907 fontes de publicação testadas, sendo que nenhuma conseguiu reproduzir o título original com perfeição, além disso, a maioria das fontes de publicação testada gerou o mesmo título. Apesar disso, é possível notar diferenças em alguns casos específicos, sendo algumas diferenças pequenas, como trocar "*Li*"por "*lithium*"enquanto outras mudaram totalmente o titulo, embora tenham-o mantido válido.

Tabela 10.5: Comparação dos títulos gerados fora dos conjuntos de treino

Experiment	Title
True Title	Hollow Spheres Consisting of SnS Nanosheets Conformally Coated with S-Doped Carbon for Advanced Lithium-/Sodium-Ion Battery Anodes
t5-small	is a promising anode material for lithium- and sodium-ion batteries due
1st	In Situ Produced SnS Nanosheets Coated with S-Doped Carbon
2nd	In Situ Generated SnS Nanosheets@C HEAT Spheres with Poly
3rd	Self-sacrifice Template and Carbon Source for High-Performance Lith
True Title	Optimum output temperature setting and an improved bed structure of metal hydride hydrogen storage reactor for thermal energy storage
t5-small	„„ 76 to 90% and GEOR from 65
1st	Stabilization of metal hydrides reactor for thermal energy storage
2nd	Stable hydrogen discharging performance of metal hydrides reactor for thermal energy storage
3rd	Stabilization of a metal hydride reactor for thermal energy storage

Tabela 10.6: Comparação dos títulos gerados no terceiro experimento variando a fonte da publicação

Título	**Ocorrências**
Monodispersed MnO nanoparticles in graphene-an interconnected N-doped 3D carbon framework as a highly efficient gas cathode in Li-CO2 batteries	0
MnO nanoparticles dispersed in a graphene-interconnected N-doped 3D carbon framework for Li-CO2 batteries	5458
MnO nanoparticles dispersed in a graphene-interconnected N-doped 3D carbon framework for lithium-co2 batteries	197
Ultrafine MnO nanoparticles dispersed in a graphene-interconnected N-doped 3D carbon framework for Li-CO2 batteries	30
Electrospun nanofibers for high-performance Li-CO2 batteries	1

10.9 Conclusão

De maneira geral, o primeiro experimento é o que parece ter apresentado melhor potencial de extrapolação, o que é coerente com os números apresentados nas Tabelas 10.1 – 10.3, mostrando que se treinado corretamente o modelo parece ser capaz de gerar títulos com sucesso. O segundo experimento mostra que o modelo tem grande potencial de aprender a tarefa e replicar os dados de treino com exatidão. Enquanto que o terceiro experimento não mostrou significativas diferenças em relação ao primeiro, apenas parecendo ser necessário que o modelo treinasse com mais dados.

Em relação ao terceiro experimento, mudar a fonte de publicação não parece ter tido um grande impacto na maioria dos casos, embora em alguns casos tenha alcançado resultados bem interessantes, sendo bem diferentes do título original porém ainda sendo um título satisfatório para o artigo.

Este trabalho mostra sobretudo o potencial de tal modelo para realização da tarefa proposta, abrindo diversos caminhos para melhoria como por exemplo repetir todos os experimentos fixando as uma mesma base de treino e teste; treinar em um ambiente com melhores configurações para ser possível treinar com todo conjunto de dados por mais épocas, ou até que as métricas definidas atinjam valores satisfatórios; estudar o comportamento da métrica *Rouge score*; testar em outras bases de dados, contendo artigos mais variados e de diversas áreas e assuntos; realizar um estudo mais aprofundado na incorporação da informação da fonte da publicação.

11

On the Performance of Twitter-Specific Language Models on a Brazilian Portuguese Tweets Dataset for Sentiment Analysis.

Felipe Bevilaqua Foldes Guimarães
Programa de Engenharia de Sistemas e Computação - COPPE

Abstract

Tweet sentiment analysis is one of the most common tasks studied in the literature where machine learning (ML) methods are applied to analyze social media data. Recently, in Natural Language Processing (NLP) and other areas of ML, transformers models achieved remarkable progress in a wide variety of tasks, including sentiment analysis. Some recent work has shown the potential of Twitter-specific pre-trained models for Twitter-related tasks. In this work, we evaluate how two Twitter-specific language models perform without any additional fine-tuning in the dataset TweetSentBr, which contains Tweets in Brazilian Portuguese. For TimeLMs, an English monolingual model, we have translated the text to English to use the model. For XLM-T, a multilingual model, we have used the original Portuguese text. For both pre-trained models evaluated, we obtained competitive results on metrics such as accuracy and F1 score without additional training or fine-tuning, thus showing that these pre-trained models can achieve excellent generalization capabilities.

11.1 Introduction

Machine learning (ML) methods have been widely used in the last decade to analyze and process social media data. On Twitter-related data specifically, sentiment analysis is one of the most common ML tasks researched in the literature. The main goal of sentiment analysis is to classify the most probable sentiment associated with a Tweet. A common approach is to classify a sentence as a negative, neutral, or positive sentiment, which can provide valuable information when monitoring the launch of a new product or the public sentiment related to a politician in elections.

In this context, deep learning (DL) methods have become one of this task's most used algorithm families. More specifically, transformer-based architectures have recently become dominant in natural language processing (NLP), outperforming other neural models such as convolutional and recurrent neural networks in many NLP tasks, including sentiment analysis (Wolf et al., 2020).

This work evaluates how two Twitter-specific transformer-based language models perform without additional fine-tuning in the dataset TweetSentBr. This dataset contains annotations of sentiment analysis for 15000 Tweets in Brazilian Portuguese (Brum e Graças Volpe Nunes, 2017).

11.2 TweetSentBr Dataset

The dataset TweetSentBr (Brum e Graças Volpe Nunes, 2017) comprises a corpus of 15.000 tweets related to some Brazilian TV Shows. Seven different annotators manually annotated those Tweets into negative, neutral, and positive classes. The number of occurrences of each class for both the training and test set is in 11.1.

Tabela 11.1: TweetSentBr Dataset (adapted from (Brum e Graças Volpe Nunes, 2017)

Classes	***Training Set***	***Test Set***	***Total***
Positive	5.745 (44%)	903(45%)	6.648
Negative	3.414 (26%)	512 (25%)	3.926
Neutral	3.840(29%)	586 (29%)	4.426
Total	12.999	2.001	15.000

The authors slightly modified the tweets in the dataset. The transformations included replacing numbers (dates, currency values) by 'NUMBER' token and replacing user names and links by the tokens USERNAME and URL, respectively. Lastly, they have

trimmed the repetition of characters to a minimum of 3 repeat characters (Brum e Graças Volpe Nunes, 2017). Emojis and other elements of informal text were kept.

11.3 HuggingFace transformers

The Transformers library (Wolf et al., 2020), developed by HuggingFace, is dedicated to supporting Transformer-based architectures and facilitating the distribution of pre-trained models. At the library's core is an implementation of the Transformer, designed for research and production.

The library mirrors the standard NLP machine learning model pipeline: process data, apply a model, and make predictions. A critical NLP-specific aspect of the library is the implementation of the tokenizers necessary to use each model. Tokenizer classes can either be instantiated from a corresponding pre-trained model or can be configured manually (Wolf et al., 2020).

Transformers is open-source and community designed to facilitate users to access large-scale pre-trained models, build and experiment on top of them, and deploy them in downstream tasks with state-of-the-art performance. The library has gained significant organic traction since its release and is set up to continue to provide core infrastructure while helping to facilitate access to new models (Wolf et al., 2020).

11.4 Twitter-Specific Language Models

Some recent work has shown the effectiveness of starting with existing pre-trained generic language models and continuing training them on Twitter corpora for Twitter-related NLP tasks(Barbieri, Camacho-Collados et al., 2020). In this work, two language models pre-trained on Twitter-specific data were used.

TimeLMs (Loureiro et al., 2022) is an English monolingual language model based on roBERTa, pre-trained in a generic corpus and fine-tuned with Twitter data. TimeLMs is periodically retrained with recent tweets on a continual learning strategy that aims to enhance its capacity to deal with future and out-of-distribution tweets. The latest release of TimeLMs fine-tuned for sentiment analysis is available at HuggingFace Hub at `https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`. By this writing, the model had been trained in more than 124 million tweets from January 2018 to December 2021.

XLM-Twitter or XLM-T (Barbieri, Anke e Camacho-Collados, 2021) is an XLM-R model pre-trained on a multilingual Twitter dataset starting from XLM-R checkpoints. XLM-T was trained on about 198 million tweets and fine-tuned for sentiment analysis in 8 different languages (Arabic, English, French, German, Hindi, Italian, Portuguese

and Spanish). It is available in the HuggingFace Hub at `https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment`.

11.5 Experiments

In this work, we applied two Twitter-specific language models to the TweetSentBr dataset without additional fine-tuning. The models used were TimeLMs and XLM-T. To use TimeLMs, a monolingual language model, we have tried a methodology that comprises the translation of the Portuguese Tweets to the English language followed by the usage of the model. For XLM-T, as it is a multilingual language model, we could use the original Portuguese text.

For the TimeLMs experiment, we did the translations of the tweets with Googletrans, a free and unlimited Google translate API for Python. Googletrans is not an official Google Translate API and was used because it is free and easy to use.

For preprocessing, we used the tokenizers that were made available together with the pre-trained models. Therefore, we have not applied any manual pre-processing of the text, and the only changes in the raw tweets were the ones made by the dataset authors and the translation in the case of the TimeLMs experiment.

The notebook with the code used in the experiments is available at `https://colab.research.google.com/drive/1hD03gl49bRXugeAiexU6wHXEHb5Hjr9m?usp=sharing`.

11.6 Results

Following the described methodology, we have achieved competitive results compared to previous works. The results obtained from previous works and in the experiments are shown in Table 11.2.

From Table 11.2 we can see that the transformer-based models achieved the best results. While the best performance was still from Barros et al. (Barros, Pedrini e Dias, 2021) we highlight that our results were achieved using the tokenizers made available with the models and without any fine-tuning in the used dataset, therefore, done in a simple way and with potential room for improvement.

Tabela 11.2: Performance on TweetSentBr Dataset

Authors	***Classifier***	***F1-Score***	***Accuracy***	**Trained or Fine-tuned on TweetSentBr**
Brum e Graças Volpe Nunes, 2017	Naive Bayes	0.598	0.649	Yes
Brum e Graças Volpe Nunes, 2017	SVM	0.596	0.649	Yes
Sakiyama, Silva e Matsubara, 2019	TextCNN	0.656	0.684	Yes
Assis Nascimento, 2019	Ensemble (MLP+LR+GNB)	0.500	0.710	Yes
Barros, Pedrini e Dias, 2021	BERT base	0.729	0.747	Yes
Barros, Pedrini e Dias, 2021	BERT emoji	0.740	0.758	Yes
-	TimeLMS	**0.694**	**0.703**	**No**
-	XLM-Twitter (multilingual)	**0.709**	**0.720**	**No**

11.7 Conclusion

The results have shown that transformer-based architectures can achieve excellent generalization capabilities even without fine-tuning in a specific dataset. This suggests that using Twitter-specific pretrained language models might be a simple but powerful strategy to perform sentiment analysis classification on tweets.

In future works we pretend to evaluate the effect of fine-tunning the models as well as try the methodology in other datasets.

12

Um Estudo Direcionado à Comparação de Resultados de Algoritmos de Extração de Palavras-Chave Oriundas de Artigos Científicos Em Língua Portuguesa

Glauco Primo e Fernando Costa de Souza
Programa de Engenharia de Sistemas e Computação - COPPE

Resumo

Palavras-chave são de suma importância no contexto de busca e recuperação de documentos. Cada vez torna-se mais importante conseguir buscar de maneira eficiente na enorme quantidade de informações que obtem-se na internet, principalmente em tempos de ensino remoto. Neste contexto, torna-se indispensável a utilização de palavras- chave em documentos científicos afim de agilizar a compreensão e revelância de determinado documento.

Existem diversos trabalhos em língua inglesa que descrevem bem o comportamento de algoritmos extratores de palavras-chave e o quão bem eles performam. Contudo, não existem muitos trabalhos que fazem esse tipo de relação qualitativa em língua portuguesa, principalmente devido a dificuldade de obter uma boa base de dados para certificar a competência do algoritmo.

Este trabalho visa levantar uma base de dados em língua portuguesa e verificar o grau de acertos dos já conhecidos algoritmos de extração de palavras.

12.1 Introdução

Existe uma fonte de informações rápida, fidedigna, eficaz e científica que pode ser obtida em mecanismos de busca de alta relevância: artigos acadêmicos. Porém ao buscar por um termo simples como por exemplo, Covid-19, nestes mecanismos, obtem-se quase 5 milhões de documentos.

Neste mar de documentos, relevantes ou não, para sua pesquisa, torna-se não só importante como essencial, o uso de palavras-chave adequadas. Algoritmos de extração de palavras-chave de textos tornam-se ferramentas valiosas para ajudar na obtensão dos principais termos que melhor resumem um documento.

Existem diversos trabalhos realizados em língua inglesa que demonstram o comportamento de algoritmos extratores de palavras-chave, tanto os supervisionados (Witten et al., 1999), como os não supervisionados (Hasan e Ng, 2010). Contudo, não é uma tarefa fácil encontrar uma boa base classificada em língua portuguesa para testar a eficácia de tais algoritmos.

Na abordagem supervisionada, as palavras- chave são classificadas de forma binária, ou seja, é sim uma palavra-chave, ou não é uma palavra- chave. Existem diversos algoritmos de aprenzidado de máquina possíveis, basta obter um dataset devidamente classificado para tal feito. Ian H. Witten propõe por exemplo, KEA, um sofisticado algoritmo que utiliza métodos léxicos com aprendizado de máquina (Witten et al., 1999) para predizer as palavras-chave.

Os algoritmos não supervisionados podem ser resumidos como um problema de classificação utilizando grafos (Hasan e Ng, 2010), onde os nós são os termos e as arestas representam relações de recorrência. Basta obter, por exemplo, os 10 melhores nós e suas relações de recorrência no texto. Deve-se notar que as palavras que melhor sintentizam o texto, costumam aparecer logo no ínicio do documento quando se tratando de trabalhos acadêmicos. Logo, os primeiros termos encontrados, costumam recorrentemente serem os melhores, como é possível verificar na Figura 12.1.

O intuito deste artigo é verificar se os algoritmos extratores de palavras-chave não supervisionados como Single Rank (Wan e Xiao, 2008) e Position Rank (Florescu e Caragea, 2017) tem uma performance melhor ou pior que na literatura conhecida em língua inglesa, em termos de Precision, Recall e F1, métricas utilizadas pelos autores em seus trabalhos.

Para realizar este trabalho, precisou-se portanto, de um dataset grande o suficiente e relevante que provesse além das informações necessárias, o padrão ouro. O melhor caminho para a obtênção de um conjunto de documentos com um padrão ouro de qualidade foi justamente criando um dataset de papers acadêmicos, pois estes normalmente já tem as palavras-chave que o autor julgou serem mais pertinentes e que melhor sintetizaram o artigo. Sendo assim, o padrão ouro era facilmente obtido lendo as palavras-chave do autor.

O estudo avaliou a prevalência de sintomas de depressão e ansiedade em uma amostra de trabalhadores brasileiros de diversos segmentos, durante a pandemia da Covid-19. Foi também verificada a correlação entre as escalas de ansiedade e depressão dos instrumentos de rastreio. Foram coletados dados on-line por meio de três instrumentos: questionário sociodemográfico e ocupacional, a Depression, Anxiety and Stress Scale - Short Form e o Inventário de Saúde Mental Ocupacional. Participaram 503 profissionais, destes 78,5% do sexo feminino, com idade média de 41,38 anos, das quais 92% cursaram o ensino superior e residiam na região Sul do Brasil. Ambas as escalas detectaram maior prevalência de sintomas de ansiedade em mulheres (54,3% e 59,9%) e em pessoas solteiras (68,8% e 68,1%). Houve associação significativa entre desfechos de sintomas de ansiedade e depressão e prevalência de duas variáveis independentes: o contato com pessoas diagnosticadas com Covid-19 e sentir-se preocupado com a pandemia. O Inventário de Saúde Mental Ocupacional mostrou maior sensibilidade para aferir sintomas de ansiedade e discriminar os trabalhadores que apresentam sintomas daqueles que indicam ter saúde mental, quando comparado ao outro instrumento. Sugerem-se estudos longitudinais para capturar os efeitos de longo termo dos desfechos avaliados, a fim de aperfeiçoar a análise dos preditores dos valores críticos e não críticos dos sintomas de agravos à saúde mental.

Palavras-chave escolhidas pelo autor:
trabalhadores; prevalência; ansiedade; depressão; covid-19

Figura 12.1: Resumo retirado do abstract de um dos artigos do conjunto de dados, seguido das palavras- chave escolhidas pelo autor (Florescu e Caragea, 2017)

Foi criado um conjunto de 200 documentos a partir de 200 artigos acadêmicos, extraindo de cada um o título, o abstract e as palavras-chave.

Os algoritmos utilizados no contexto deste trabalho foram FirstPhrases, YAKE, TextRank, SingleRank, TopicRank, PositionRank e MultipartiteRank. Em seguida, bastava comparar as palavras-chave oriundas de cada algoritmo extrator e comparar com as palavras-chave do padrão ouro. Com isso, foi possível criar as métricas necessárias para as devidas comparações com a ampla literatura obtida em língua inglesa.

12.2 Trabalhos Relacionados

É possível encontrar uma vasta gama de trabalhos que abordam algoritmos geradores de palavras-chave, contudo, para o escopo deste trabalho, infelizmente, todos os trabalhos relacionados são em língua inglesa.

Alguns trabalhos utilizam algoritmos supervisionados para gerar as palavras-chave. Neste contexto é necessária um extenso conjunto de dados, onde as palavras são classificadas como palavra-chave caso positivo, e não palavra-chave, caso negativo. Para destacar alguns, Chengzhi propôs a utilização do modelo Conditional Random Fields (CRF) para a extração das palavras (Witten et al., 1999), que segundo os autores, se mostram mais eficiente que outros modelos de aprendizado de máquina como os modelos lineares e modelos de máquina de suporte de vetores (SVM).

Outro modelo supervisionado proposto por Ian, chamado de Automatic Keyphrase Extraction ou KEA (Witten et al., 1999), utiliza a Máquina de Naïve Bayes como algoritmo de aprendizado para treinar o seu conjunto de dados e extrair as palavras-chave.

No âmbito dos algoritmos não supervisionados, existem diversos trabalhos realizados em língua inglesa. Muitos utilizam o algoritmo Text Rank, que é uma melhoria proposta por Rada do já consolidado algoritmo Page Rank (Mihalcea e Tarau, 2004). O Text Rank é um algoritimo de ranking baseado em grafos que em sua ideia central retorna os termos mais recomendados, ou mais votados como vértices.

Outra melhoria foi a proposta por Suhan Pan, em seu trabalho An Improved TextRank Keywords Extraction Algorithm (Pan, Li e Dai, 2019). Neste TextRank melhorado, os autores escolhem como input para o TextRank um vetor de termos que eles já consideraram boas candidatas, pois utilizam o algoritmo TF-IDF e o algoritmo de entropia média para calcular os pesos das palavras de maior relevância no Texto.

Outro algoritmo que vale destaque, é o algoritmo PositionRank proposto por Corina Florescu e Cornelia Caragea (Florescu e Caragea, 2017). É proposta uma importante modificação em relação a um PageRank tradicional. Em um PageRank as palavras são rankeadas em relação ao documento inteiro, enquanto no PositionRank cada palavra possui um peso que relaciona todas as posições de ocorrencia da palavra. Esta palavra sim, é incorporada em um PageRank enviesado, que atribui uma “preferência” diferente para cada palavra.

12.3 Modelos para extração automática de palavras-chave

No presente trabalho, foram comparados os modelos de aprendizado não-supervisionado estatístico: FirstPhrases e YAKE. E modelos de aprendizado não-supervisionado baseado em grafos: TextRank, SingleRank, TopicRank, PositionRank e MultipartiteRank.

Nesta seção, o funcionamento de cada modelo será melhor detalhado.

12.3.1 Modelos de aprendizado não-supervisionado estatístico

Os métodos estatísticos se utilizam de métricas como frequência de cada palavra e co-ocorrência para determinar as palavras mais relevantes em um documento. Alguns métodos utilizam características simbólicas mais avançadas, como TF-IDF e YAKE!.

O método FirstPhrases é usado como linha de base. Ele simplesmente extrai as primeiras palavras do texto, ignorando as stopwords.

O método YAKE utiliza como base certas características do texto para pontuar a relevância das palavras. São 5 características. Casing, considera que termos em letras maiúsculas tendem a ser mais relevantes. Posição do termo, termos que estão mais no começo do texto tendem a ser mais relevantes. Frequência dos termos, termos que aparecem mais vezes em um texto tendem a ser mais relevantes. Relação do termo com o contexto, mede com quantos termos diferentes o candidato co-ocorre, termos mais relevantes co-ocorrem com menos termos diferentes. Termo frase diferente, termos que aparecem em múltiplas frases tendem a ser mais relevantes (“YAKE! Keyword extraction from single documents using multiple local features” 2020).

12.3.2 Modelos de aprendizado não-supervisionado baseados em grafos

Métodos baseados em grafos geram um grafo de termos relacionados no documento. São usados métodos de rankeamento de grafos para computar a importância de cada nó – palavra – no grafo.

No método TextRank, a conexão entre os nós no grafo ocorre a partir da co-ocorrência de palavras dentro de uma janela (o padrão é w=2). O grafo é não direcionado e não ponderado. O algoritmo utilizado para determinar a importância de cada nó é o PageRank, do Google, é uma algoritmo recursivo, onde a importância de cada nó é determinada pela importância dos nós que se relaciona (Mihalcea e Tarau, 2004).

O método SingleRank utiliza a mesma lógica e etapas que o TextRank, com a diferença de que o algoritmo PageRank é computado considerando os relacionamentos entre nós de forma ponderada e uma maior janela w = 10 (Wan e Xiao, 2008).

TopicRank utiliza uma abordagem distinta, onde cada nó no grafo é um tópico. Um tópico é identificado extraindo frases nominais, contendo a maior sequência de substantivos e adjetivos. O relacionamento entre os diferentes tópicos é dado pela sobreposição de palavras contidas em cada um, quanto maior a sobreposição, mais similares são os tópicos. Após o relacionamento entre os tópicos, é utilizado o mesmo algoritmo de ranqueamento do TextRank para rankear os tópicos. Para a seleção das palavras-chave, os tópicos mais importantes são selecionados e as palavras mais frequentes no tópico são selecionadas (Bougouin, Boudin e Daille, 2013).

No PositionRank, o grafo é construído assim como no TextRank, com as palavras sendo nós e os relacionamentos sendo co-ocorrências entre as palavras em uma janela de 10 palavras (w = 10), o peso de cada aresta é dado pelo número de vezes que as palavras apareceram na mesma janela. A principal diferença para o TextRank é no cálculo do rankeamento usando um algoritmo Pagerank enviesado pela posição das palavras no texto. A principal ideia é que palavras no começo do texto tendem a ser mais relevantes (Florescu e Caragea, 2017).

MultipartiteRank é baseado no TopicRank, considerando frases-chave em tópicos como nós no grafo. O relacionamento é direcionado entre as frases-chave é dado se elas pertencerem a diferentes tópicos e o peso é computado de acordo com a distância entre elas no documento. O algoritmo de ranqueamento das frases-chave é o mesmo que o TextRank (Adrien Bougouin, Florian Boudin, and Beatrice Daille, 2013).

12.4 Experimentos e Resultados

12.4.1 Base de dados

Os experimentos foram realizados utilizando como base de dados 197 artigos científicos publicados no portal SciELO - Brazil [1]. Os artigos foram obtidos buscando-se pelos termos: educação à distância; educação na pandemia; inclusão digital; jogos educacionais; e mobile learning. Na Tabela 12.1 está a distribuição de artigos em cada um dos termos.

Foram utilizados os títulos e resumos concatenados como textos de entrada para os algoritmos extratores de palavras-chave. Além disso, as palavras-chave dos autores foram utilizadas como o alvo para comparar os resultados dos algoritmos.

Todo o processo de extração de informações foi feito de forma manual pelos autores deste artigo.

A base de dados encontra-se disponível publicamente, assim como todo o código utilizado nos experimentos [2].

[1] `https://www.scielo.br`

[2] `https://github.com/CostaFernando/keyword-extraction-portuguese`

Tabela 12.1: Distribuição de documentos por termo de busca.

Termo	#Documentos
Educação a distância	15
Educação na pandemia	83
Inclusão digital	29
Jogos educacionais	30
Mobile learning	40

12.4.2 Métricas de avaliação

Foram utilizadas as seguintes métricas para avaliação dos experimentos: Mean Reciprocal Rank (MRR), Precision, Recall e F1. Visto que são utilizadas em trabalhos prévios da literatura (Rada Mihalcea and Paul Tarau, 2004; Xiaojun Wan and Jianguo Xiao, 2008; Kazi Saidul Hasan and Vincent Ng., 2010).

Essas métricas foram computadas utilizando diferentes rankings (k) dos resultados dos modelos (desempenho@k). Os valores de k utilizados variaram de 1 a 10. Por exemplo: precision@4 considera apenas as 4 primeiras palavras-chave retornadas pelo modelo para calcular a precisão.

A referência de alvo para comparação das palavras-chave relevantes são as palavras-chave que os autores de cada documento determinaram.

12.4.3 Resultados e Discussão

O objetivo principal dos experimentos foi determinar quais modelos de extração de palavras- chave melhor desempenham em textos científicos de língua portuguesa.

Para isso, foram comparados os modelos de aprendizado não-supervisionado estatístico: FirstPhrases e YAKE. E modelos de aprendizado não-supervisionado baseado em grafos: TextRank, SingleRank, TopicRank, PositionRank e MultipartiteRank.

Todos os modelos utilizaram as mesmas etapas de processamento de dados: remoção de palavras comuns do Português (stopwords), pontuação e sinais de parênteses.

Além disso, as configurações utilizadas em cada modelo foram as configurações padrão nas implementações dos modelos da biblioteca pke - python keyphrase extraction[3].

Na Figura 12.2, são comparados diferentes modelos da literatura utilizando a métrica MRR em diferentes rankings. Os modelos que obtêm melhores desempenhos são TopickRank e MultipartiteRank, chegando a 0,293 com o MultipartiteRank para k = 10.

[3] `https://github.com/boudinfl/pke`

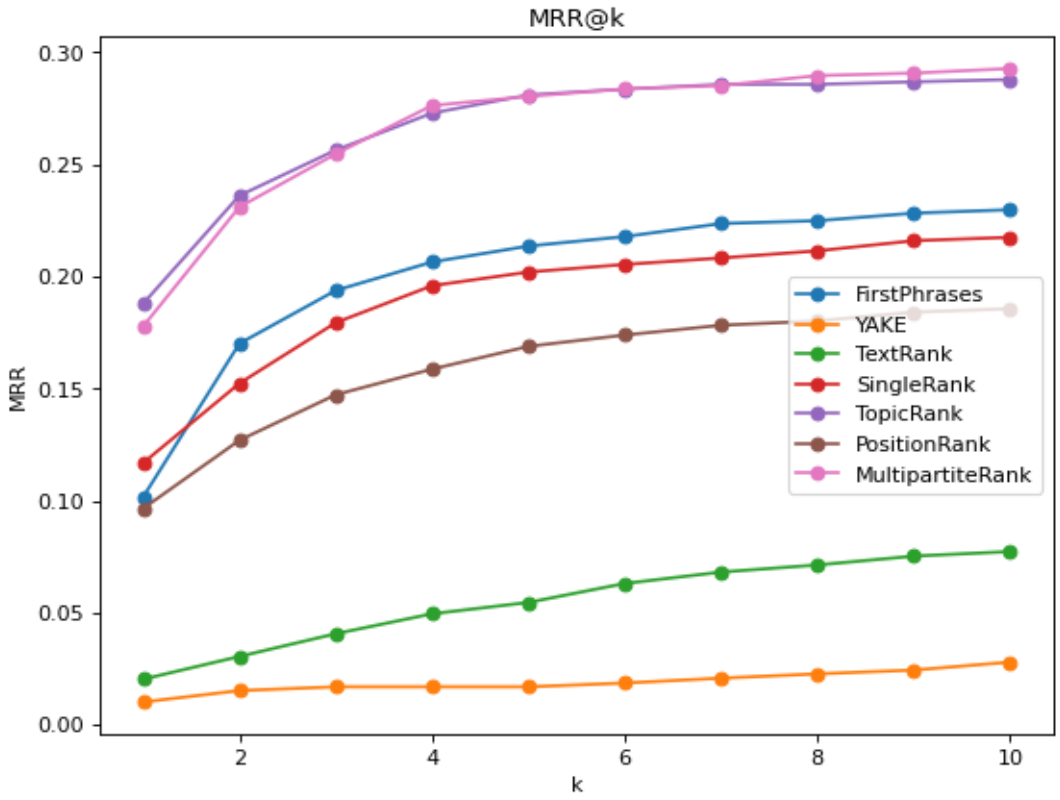


Figura 12.2: Distribuição de documentos por termo de busca

Os modelos YAKE e TextRank obtiveram os resultados mais baixos na métrica MRR, com o YAKE chegando a um valor máximo de MRR de 0,028 para k = 10.

Além disso, foram medidos precision (P), recall (R) e F1-score (F1) com k variando de 1 a 10 para todos os modelos.

O modelo MultipartiteRank obteve de forma geral os melhores valores para as métricas P, R e F1. Atingindo F1 máximo de 13,60% para k = 4. Como pode ser observado na Figura 12.3 e Tabela 12.2.

12.4.4 Considerações sobre os desempenhos dos modelos

Ao analisar o desempenho dos modelos, deve-se levar em consideração os dados utilizados como entrada. Foram resumos de artigos científicos, que normalmente contém maior densidade de informação, por serem mais curtos e conterem os termos mais relevantes.

O desempenho desses modelos pode ser diferente quando aplicados em textos completos, com maior quantidade de palavras e menos densidade de informação.

Isso pode ser verificado com o desempenho relativamente alto alcançado pelo algoritmo FirstPhrases. Esse era para ser um método apenas usado como linha de base para as

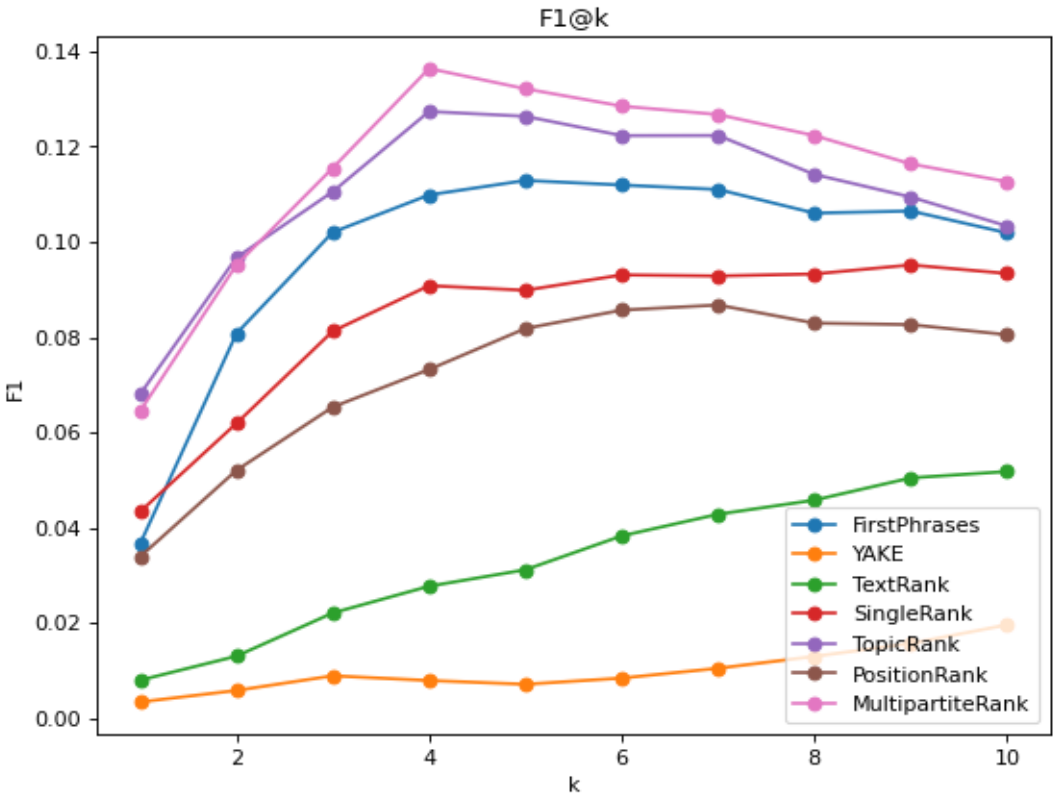


Figura 12.3: Curvas F1 para diferentes modelos

métricas, dada sua simplicidade. No entanto, ele conseguiu se sair melhor do que outros métodos, acertando palavras- chave relevantes citadas pelos autores dos artigos. Isso provavelmente se deve ao fato de os resumos dos artigos conterem maior densidade de palavras- chave do que o texto completo, favorecendo esse tipo de método.

Em contraste, o método YAKE obteve os mais baixos resultados. Indicando uma possível dificuldade de funcionar com textos menores, com maior densidade de palavras-chave, como os resumos dos artigos. Os resultados contrastam com os obtidos na literatura (Campos et al., 2020).

Tabela 12.2: Comparativo de diferentes modelos em termos de Precision (P), Recall (R) e F1- score para k = 2, 4, 6, 8. Os melhores resultados estão destacados em azul. YAKE obteve o desempenho mais baixo, com 1,3% F1 para k = 8.

Modelo	Top2			Top4			Top6			Top8		
	p_2	r_2	f1_2	p_4	r_4	f1_4	p_6	r_6	f1_6	p_8	r_8	f1_8
FirstPhrases	12,90%	5,90%	8,10%	11,80%	10,50%	11,00%	9,90%	13,20%	11,20%	8,40%	14,80%	10,60%
YAKE	1,00%	0,40%	0,60%	0,90%	0,70%	0,80%	0,80%	0,90%	0,80%	1,00%	1,80%	1,30%
TextRank	2,00%	1,00%	1,30%	2,90%	2,70%	2,80%	3,30%	4,70%	3,80%	3,60%	6,60%	4,60%
SingleRank	9,90%	4,60%	6,20%	9,60%	8,80%	9,10%	8,10%	11,20%	9,30%	7,30%	13,30%	9,30%
TopicRank	**16,00%**	**7,00%**	**9,70%**	13,70%	12,10%	12,70%	10,80%	14,40%	12,20%	9,00%	16,00%	11,40%
PositionRank	8,60%	3,80%	5,20%	7,70%	7,10%	7,30%	7,40%	10,30%	8,60%	6,50%	11,80%	8,30%
MultipartiteRank	15,70%	6,90%	9,50%	**14,70%**	**13,00%**	**13,60%**	**11,30%**	**15,20%**	**12,80%**	**9,60%**	**17,20%**	**12,20%**

12.5 Conclusão e Trabalhos Futuros

Este artigo teve o objetivo de propor uma abordagem ainda muito inexplorada, que é a de trabalhos que abordem a extração de palavras- chave de textos em língua portuguesa. Um grande desafio aqui foi de obter uma boa base devidamente classificada, fidedígnea e com uma boa base científica.

Com os 200 artigos em língua portuguesa, devidamente classificados com as palavras-chaves fornecidas pelos autores, foi possível fazer todos os experimentos utilizando os algoritmos já conhecidos como bons extratores de palavras- chave. E pode ser comprovada a eficácia destes classificadores utilizando todos os critérios e métricas supracitados.

Tentamos utilizar tais técnicas para gerar questões do tipo “complete as lacunas com a expressão adequada”, na qual utilizamos os extratores para ver 10 termos-chaves mais importantes em um texto, e gerar uma questão de prova utilizando a frase e o contexto em que se encontra esta palavra-chave. Contudo, os resultados não foram ótimos, e não existe uma boa literatura para comparação de resultados ainda. Como trabalho futuro gostaríamos de conseguir uma boa base classificada de questões desse tipo, de preferência de um banco de provas, para tentar alterar algum dos algoritmos não supervisionados abordados neste trabalho, ou mesmo partir para uma abordagem supervisionada.

Bibliografia

Ahuja, Shreya e Gaurav Dubey (ago. de 2017). "Clustering and sentiment analysis on Twitter data". Em: pp. 1–5. DOI: 10.1109/TEL-NET.2017.8343568.

Airbnb (2022). *Airbnb, O que é a Airbnb e como funciona*. URL: https://www.airbnb.com.br/help/article/2503/o-que-%C3%A9-o-airbnb-e-como-ele-funciona (acesso em 04/08/2022).

Alpek, L. e R. Tesits (2020). "Measuring Regional Differences in Employability in Hungary". English. *Applied Spatial Analysis and Policy* 13.2. Publisher: Springer, pp. 329–347. ISSN: 1874463X. DOI: 10.1007/s12061-019-09306-6. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85068225999&doi=10.1007%5C%2fs12061-019-09306-6&partnerID=40&md5=42b39dcd74c517600f4677b20b07fadb.

Assis Nascimento, Paulo de (2019). "Aplicando Ensemble para Classificação de Textos Curtos em Português do Brasil". Dissertação de Mestrado. Universidade Federal de Pernambuco.

Barbieri, Francesco, Luis Espinosa Anke e José Camacho-Collados (2021). "XLM-T: A Multilingual Language Model Toolkit for Twitter". *CoRR* abs/2104.12250. arXiv: 2104.12250. URL: https://arxiv.org/abs/2104.12250.

Barbieri, Francesco, José Camacho-Collados et al. (2020). "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". *CoRR* abs/2010.12421. arXiv: 2010.12421. URL: https://arxiv.org/abs/2010.12421.

Barbosa, Carlos Eduardo, Yuri Lima, Alan de Oliveira Lyra et al. (2019). *Healthcare 2030: A view of how changes on technology will impact Healthcare in 2030*. Inglês. 00000. URL: http://labfuturo.cos.ufrj.br/reports/healthcare2030.pdf.

Barbosa, Carlos Eduardo, Yuri Lima, Eduardo Mioto et al. (2017). *Working in 2050: A view of how changes on the work will affect society*. Inglês. 00000. URL: http://labfuturo.cos.ufrj.br/reports/working2050.pdf.

Barros, Tiago Martinho de, Helio Pedrini e Zanoni Dias (2021). "Leveraging Emoji to Improve Sentiment Classification of Tweets". Em: *Proceedings of the 36th Annual*

ACM Symposium on Applied Computing. SAC '21. Virtual Event, Republic of Korea: Association for Computing Machinery, pp. 845–852. ISBN: 9781450381048. DOI: 10.1145/3412841.3441960. URL: https://doi.org/10.1145/3412841.3441960.

Bateman, Scott, Carl Gutwin e Miguel Nacenta (2008). "Seeing things in the clouds". Em: *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia - HT '08*. ACM Press. DOI: 10.1145/1379092.1379130. URL: https://doi.org/10.1145/1379092.1379130.

Belém, Fabiano M., Jussara M. Almeida e Marcos A. Gonçalves (dez. de 2016). "A survey on tag recommendation methods". *Journal of the Association for Information Science and Technology* 68.4, pp. 830–844. DOI: 10.1002/asi.23736. URL: https://doi.org/10.1002/asi.23736.

Berry, Michael W, Zlatko Drmac e Elizabeth R Jessup (1999). "Matrices, vector spaces, and information retrieval". *SIAM review* 41.2, pp. 335–362.

Bougouin, Adrien, Florian Boudin e Béatrice Daille (out. de 2013). "TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction". Em: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan: Asian Federation of Natural Language Processing, pp. 543–551. URL: https://aclanthology.org/I13-1062.

BRASIL (2022a). *Câmara dos Deputados - Dados Abertos*. URL: https://dadosabertos.camara.leg.br/index.html (acesso em 25/06/2022).

— (2022b). *Partidos Políticos Registrados no TSE - Tribunal Superior Eleitoral*. URL: https://dadosabertos.camara.leg.br/index.html (acesso em 25/06/2022).

Brum, Henrico Bertini e Maria das Graças Volpe Nunes (2017). "Building a Sentiment Corpus of Tweets in Brazilian Portuguese". *CoRR* abs/1712.08917. arXiv: 1712.08917. URL: http://arxiv.org/abs/1712.08917.

Chai, Zi Xu (2021). "Automatic parental guide ratings for short movies". Tese de dout. UTAR.

Classificação Brasileira de Ocupações (2022). *CBO - Busca por Título - 5.1.7*. (Acesso em 26/06/2022).

CNI (jan. de 2020). *Falta de Trabalhador Qualificado*. Português. Rel. técn. 76. Brasília, DF: Confederação Nacional da Indústria, p. 3. URL: http://www.portaldaindustria.com.br/estatisticas/sondesp-76-falta-de-trabalhador-qualificado/ (acesso em 14/01/2021).

Coibion, Olivier, Yuriy Gorodnichenko e Michael Weber (2020). *Labor markets during the covid-19 crisis: A preliminary view*. Rel. técn. MA, USA: National Bureau of Economic Research.

Conarq (2006). *Norma Brasileira de Descrição Arquivística*. Rio de Janeiro: Arquivo Nacional. URL: http://www.siga.arquivonacional.gov.br/images/publicacoes/nobrade.pdf.

CONIF (jun. de 2021). *Nota Oficial Sobre a Recomposição Do orçamento Da Educação*. pt-br. URL: https://portal.conif.org.br/br/component/content/article/84-ultimas-noticias/4017-nota-oficial-sobre-a-recomposicao-do-orcamento-da-educacao (acesso em 14/01/2021).

Devlin, Jacob et al. (jun. de 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". Em: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Domingues, Marcos Aurélio et al. (2014). "Exploiting Text Mining Techniques for Contextual Recommendations". Em: *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Vol. 2, pp. 210–217. DOI: 10.1109/WI-IAT.2014.100.

Dubin, David (2004). "The most influential paper Gerard Salton never wrote".

Easy2Recruit e GazzConecta (2020). *Mercado de trabalho: o dilema entre recrutamento, qualificação e desemprego em 2020*. Português. Rel. técn. 1. Curitiba, PR: Easy2Recruit. URL: https://easy2recruit.com/arquivos/Pesquisa-Easy2Recruit-GazzConecta.pdf (acesso em 14/01/2021).

Florescu, Corina e Cornelia Caragea (jul. de 2017). "PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents". Em: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1105–1115. DOI: 10.18653/v1/P17-1102. URL: https://aclanthology.org/P17-1102.

Fraga, Érica (dez. de 2020). *Pandemia acelera vagas de tecnologia e saúde e elimina de gerentes*. pt-BR. Section: Mercado. URL: https://www1.folha.uol.com.br/mercado/2020/12/pandemia-acelera-vagas-de-tecnologia-e-saude-e-elimina-de-gerentes.shtml (acesso em 11/01/2021).

Gatbonton, T.M.C. e B.E. Aguinaldo (2018). "Employability predictive model evaluator using part and JRIP classifier". English. Em: *ACM International Conference Proceeding Series*. Association for Computing Machinery, pp. 307–310. ISBN: 978-1-4503-6629-8. DOI: 10.1145/3301551.3301569.

Globo, Indianara Campos TV (s.d.). *Coren-SP vai apurar denúncia de Klara Castanho sobre vazamento de informações sigilosas*. Online; acessado em junho de 2022. Disponível no link.

Gomaa, Wael H, Aly A Fahmy et al. (2013). "A survey of text similarity approaches". *international journal of Computer Applications* 68.13, pp. 13–18.

Halvey, Martin J. e Mark T. Keane (2007). "An assessment of tag presentation techniques". Em: *Proceedings of the 16th international conference on World Wide Web - WWW '07*. ACM Press. DOI: 10.1145/1242572.1242826. URL: https://doi.org/10.1145/1242572.1242826.

Harvey, L. (2001). "Defining and measuring employability". English. *Quality in Higher Education* 7.2, pp. 97–109. ISSN: 13538322. DOI: 10.1080/13538320120059990.

Hasan, Kazi Saidul e Vincent Ng (2010). "Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art". Em: COLING '10. Beijing, China: Association for Computational Linguistics, pp. 365–373.

He, Kaiming et al. (2016). "Deep Residual Learning for Image Recognition". Em: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hearst, Marti A. et al. (2020). "An Evaluation of Semantically Grouped Word Cloud Designs". *IEEE Transactions on Visualization and Computer Graphics* 26.9, pp. 2748–2761. DOI: 10.1109/TVCG.2019.2904683.

Hillage, Jim et al. (1999). *Employability: developing a framework for policy analysis*. en. OCLC: 505053777. London: Dept. for Education e Employment. ISBN: 978-0-85522-889-7.

Inoubli, Wissem et al. (2018). "An experimental survey on big data frameworks". *Future Generation Computer Systems* 86, pp. 546–564.

Jareanpon, Chatklaw et al. (2018). "Automatic lyrics classification system using text mining technique". Em: *2018 International Workshop on Advanced Image Technology (IWAIT)*, pp. 1–4. DOI: 10.1109/IWAIT.2018.8369796.

Jones, Michael, Sandra Idrovo-Carlier e Alfredo J. Rodriguez (set. de 2021). "Automation in Colombia: assessing skills needed for the future of work". en. *Higher Education, Skills and Work-Based Learning* ahead-of-print.ahead-of-print. ISSN: 2042-3896. DOI: 10.1108/HESWBL-01-2021-0003. (Acesso em 29/09/2021).

Joulin, Armand et al. (2016). "Bag of Tricks for Efficient Text Classification". *arXiv preprint arXiv:1607.01759*.

Jurafsky, Dan e James H. Martin (2009). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall. URL: http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y.

Keller, James M., Michael R. Gray e James A. Givens (1985). "A fuzzy K-nearest neighbor algorithm". *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15.4, pp. 580–585. DOI: 10.1109/TSMC.1985.6313426.

Khan, Samee Ullah et al. (2019). "Cover the violence: A novel Deep-Learning-Based approach towards violence-detection in movies". *Applied Sciences* 9.22, p. 4963.

Koutrika, Georgia et al. (out. de 2008). "Combating spam in tagging systems". *ACM Transactions on the Web* 2.4, pp. 1–34. DOI: 10.1145/1409220.1409225. URL: https://doi.org/10.1145/1409220.1409225.

Landauer, Thomas K, Peter W. Foltz e Darrell Laham (jan. de 1998). "An introduction to latent semantic analysis". *Discourse Processes* 25.2-3, pp. 259–284. DOI: 10.1080/01638539809545028. URL: https://doi.org/10.1080/01638539809545028.

Lima, Bianca e Luiz Guilherme Gerbelli (nov. de 2020). *No Brasil, 40% dos jovens com ensino superior não têm emprego qualificado*. pt-br. URL: https://g1.globo.com/economia/concursos-e-emprego/noticia/2020/08/11/no-brasil-40percent-dos-jovens-com-ensino-superior-nao-tem-emprego-qualificado.ghtml (acesso em 14/01/2021).

Lima, Yuri et al. (mai. de 2021). "Understanding Technological Unemployment: A Review of Causes, Consequences, and Solutions". en. *Societies* 11.2, p. 50. ISSN: 2075-

4698. DOI: 10.3390/soc11020050. URL: https://www.mdpi.com/2075-4698/11/2/50 (acesso em 29/09/2021).

Lison, Pierre e Jörg Tiedemann (2016). "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". Inglês. Em: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. por Nicoletta Calzolari et al. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

Loureiro, Daniel et al. (2022). "TimeLMs: Diachronic Language Models from Twitter". *CoRR* abs/2202.03829. arXiv: 2202.03829. URL: https://arxiv.org/abs/2202.03829.

Manning, Christopher D., Prabhakar Raghavan e Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press. ISBN: 0521865719. URL: https://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Marinchev, Ivo (2006). "Practical Semantic Web – Tagging and Tag Clouds". *Journal Cybernetics and Information Technologies* 6.3, pp. 33–39.

Martinez, Victor R et al. (2019). "Violence rating prediction from movie scripts". Em: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 671–678.

McKinney, Wes (2010). "Data Structures for Statistical Computing in Python". Em: *Proceedings of the 9th Python in Science Conference*. Ed. por Stéfan van der Walt e Jarrod Millman, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

Me senti em casa: análise das revisões de experiências de hospedagem colaborativa no site Airbnb sob o prisma da confiança (dez. de 2021). *Rev. Bras. Pesq. Tur.*

Medelyan, Olena e Ian H. Witten (2008). "Domain-independent automatic keyphrase indexing with small training sets". *Journal of the American Society for Information Science and Technology* 59.7, pp. 1026–1040. DOI: 10.1002/asi.20790. URL: https://doi.org/10.1002/asi.20790.

Mihalcea, Rada e Paul Tarau (jul. de 2004). "TextRank: Bringing Order into Text". Em: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, pp. 404–411. URL: https://aclanthology.org/W04-3252.

Miller, George A. (nov. de 1995). "WordNet: a lexical database for English". *Communications of the ACM* 38.11, pp. 39–41. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

Ministério da Educação (2022). *Diretrizes Curriculares - Cursos de Graduação*. pt-br. URL: http://portal.mec.gov.br/component/content/article?id=12991 (acesso em 26/06/2022).

Mishra, T., D. Kumar e S. Gupta (2016). "Students' employability prediction model through data mining". English. *International Journal of Applied Engineering Research* 11.4. Publisher: Research India Publications, pp. 2275–2282. ISSN: 09734562. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84961746318&partnerID=40&md5=3640f70153cba34d5b2ab2297093ccaf.

Mitchell, Ryan (2018). *Web scraping with Python: Collecting more data from the modern web*. "O'Reilly Media, Inc."

Morgado, Fernando Fernandes (2010). "Representação de Documentos Através de Nuvens de Termos". Diss. de mestr. Brazil: Federal University of Rio de Janeiro. URL: https://www.cos.ufrj.br/index.php/pt-BR/publicacoes-pesquisa/details/15/2172.

Okolie, U.C., H.E. Nwosu e S. Mlanga (2019). "Graduate employability: How the higher education institutions can meet the demand of the labour market". English. *Higher Education, Skills and Work-based Learning* 9.4. Publisher: Emerald Group Publishing Ltd., pp. 620–636. ISSN: 20423896. DOI: 10.1108/HESWBL-09-2018-0089.

Orengo, V.M. e C.R. Huyck (2021). "A Stemming Algorithm for the Portuguese Language". Em: *8th International Symposium on String Processing and Information Retrieval (SPIRE)*. Laguna de San Raphael, Chile, pp. 183–193.

Pachet, F., G. Westermann e D. Laigre (2001). "Musical data mining for electronic music distribution". Em: *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pp. 101–106. DOI: 10.1109/WDM.2001.990164.

Paice, Chris D. (nov. de 1990). "Another Stemmer". *SIGIR Forum* 24.3, pp. 56–61. ISSN: 0163-5840. DOI: 10.1145/101306.101310. URL: https://doi.org/10.1145/101306.101310.

Pan, Suhan, Zhiqiang Li e Juan Dai (2019). "An Improved TextRank Keywords Extraction Algorithm". Em: *Proceedings of the ACM Turing Celebration Conference - China*. ACM TURC '19. Chengdu, China: Association for Computing Machinery. ISBN: 9781450371582. DOI: 10.1145/3321408.3326659. URL: https://doi.org/10.1145/3321408.3326659.

Partido dos Trabalhadores (2022). *Nossa História - Partido dos Trabalhadores*. URL: https://pt.org.br/nossa-historia/ (acesso em 25/06/2022).

PESC (2022). *Programa de Engenharia de Sistemas e Computação*. URL: https://cos.ufrj.br/ (acesso em 04/08/2022).

Porter, M. F. (mar. de 1980). "An algorithm for suffix stripping". *Program: electronic library and information systems* 14.3, pp. 130–137. DOI: 10.1108/eb046814. URL: https://doi.org/10.1108/eb046814.

Porter, Martin (s.d.). *The English (Porter2) stemming algorithm*. Accessed: 2022-06-23. URL: http://snowball.tartarus.org/algorithms/english/stemmer.html.

— (2022). *The Porter Stemming Algorithm*. URL: https://tartarus.org/martin/PorterStemmer/ (acesso em 04/08/2022).

Raffel, Colin et al. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". *Journal of Machine Learning Research* 21.140, pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

Read, Jesse et al. (2009). "Classifier Chains for Multi-label Classification". Em: *Machine Learning and Knowledge Discovery in Databases*. Ed. por Wray Buntine et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 254–269. ISBN: 978-3-642-04174-7. DOI: 10.1007/978-3-642-04174-7_17.

Romer, Daniel et al. (2018). "Parental desensitization to gun violence in PG-13 movies". *Pediatrics* 141.6.

Saad, Khaled et al. (2022). "A Markov Model-Based Approach for Predicting Violence Scenes from Movies". Em: *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. IEEE, pp. 21–26.

Sakiyama, Kenzo Miranda, Andre Quintiliano Bezerra Silva e Edson Takashi Matsubara (2019). "Twitter breaking news detector in the 2018 Brazilian presidential election using word embeddings and convolutional neural networks". Em: *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. DOI: 10.1109/IJCNN.2019.8852394.

Salton, Gerard (1989). "Automatic text processing: The transformation, analysis, and retrieval of". *Reading: Addison-Wesley* 169.

Santos, Herbert Salazar dos (dez. de 2021). "LABORe IES: UM FRAMEWORK PARA AVALIAÇÃO DA EMPREGABILIDADE DE CURSOS DE GRADUAÇÃO BRASILEIROS". Portuguese. Dissertation. Rio de Janeiro: Universidade Federal do Rio de Janeiro.

Shafaei, Mahsa et al. (2019). "Rating for parents: Predicting children suitability rating for movies based on language of the movies". *arXiv preprint arXiv:1908.07819*.

Shevenock, Sarah (2022). *The International Content Boom Has Made Subtitlers and Dubbers the Lifeblood of Streaming*. Acesso em: 18 jun.2022. Morning Consult. URL: https://morningconsult.com/2022/04/25/subtitles-dubbing-streaming/.

Simões, Alberto, Anália Lourenco e José Almeida (jan. de 2007). "Using Text Mining Techniques for Classical Music Scores Analysis". *New Trends in Artificial Intelligence*, pp. 791–799.

Smith, J., A. McKnight e R. Naylor (2000). "Graduate employability: Policyand performance in higher education in the UK". English. *Economic Journal* 110.464. Publisher: Blackwell Publishing Ltd, F382–F411. ISSN: 00130133. DOI: 10.1111/1468-0297.00546.

Souza Medeiros, Paola Cristine de et al. (set. de 2020). "Prevalência de sintomas de depressão e ansiedade em trabalhadores durante a pandemia da Covid-19". *Epidemiol. Serv. Saúde* 29.4.

THE - Times Higher Education (2022). *Federal University of Rio de Janeiro*. URL: https://www.timeshighereducation.com/world-university-rankings/federal-university-rio-de-janeiro (acesso em 04/08/2022).

The Economist (ago. de 2020). *The absent student - Covid-19 will be painful for universities, but also bring change*. English. URL: https://www.economist.com/leaders/2020/08/08/covid-19-will-be-painful-for-universities-but-also-bring-change (acesso em 11/01/2021).

Thijssen, Johannes G.L., Beatrice I.J.M. Van der Heijden e Tonette S. Rocco (jun. de 2008). "Toward the Employability—Link Model: Current Employment Transition to Future Employment Perspectives". en. *Human Resource Development Review* 7.2, pp. 165–183. ISSN: 1534-4843, 1552-6712. DOI: 10.1177/1534484308314955. URL: http://journals.sagepub.com/doi/10.1177/1534484308314955 (acesso em 18/08/2021).

UFRJ (mai. de 2021). *Governo libera parte de orçamento, mas previsão é que UFRJ só consiga funcionar até setembro – Universidade Federal do Rio de Janeiro*. pt-

BR. URL: https://ufrj.br/2021/05/governo-libera-parte-de-orcamento-mas-previsao-e-que-ufrj-so-consiga-funcionar-ate-setembro/ (acesso em 21/10/2021).

— (2022). *Conect UFRJ*. URL: https://www.conecta.parque.ufrj.br/ (acesso em 04/08/2022).

Van Der Heijde, C.M. e B.I.J.M. Van Der Heijden (2006). "A competence-based and multidimensional operationalization and measurement of employability". English. *Human Resource Management* 45.3, pp. 449–476. ISSN: 00904848. DOI: 10.1002/hrm.20119.

Vaswani, Ashish et al. (2017). "Attention is All you Need". Em: *Advances in Neural Information Processing Systems*. Ed. por I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Waltmann, Ben et al. (fev. de 2020). *The impact of undergraduate degrees on lifetime earnings*. en. Rel. técn. London, UK: Institute For Fiscal Studies - IFS. DOI: 10.1920/re.ifs.2020.0167. URL: https://www.ifs.org.uk/publications/14729 (acesso em 14/01/2021).

Wan, Xiaojun e Jianguo Xiao (ago. de 2008). "CollabRank: Towards a Collaborative Approach to Single-Document Keyphrase Extraction". Em: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, pp. 969–976. URL: https://aclanthology.org/C08-1122.

Wikimedia Foundation, Inc. (2022). *Precision and recall*. Accessed: 2022-06-22. URL: https://en.wikipedia.org/wiki/Precision%5C_and%5C_recall (acesso em 22/06/2022).

Witten, Ian H. et al. (1999). "KEA: Practical Automatic Keyphrase Extraction". Em: *Conference: Proceedings of the Fourth ACM conference on Digital Librarie*. Berkeley, CA, USA. URL: https://arxiv.org/abs/cs/9902007.

Wolf, Thomas et al. (jan. de 2020). "Transformers: State-of-the-Art Natural Language Processing". Em: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

Xexéo, Geraldo (abr. de 2022). *Tópicos em Busca e Aprendizado de Máquina em Texto com exemplos em Python e KNIME - Notal de Aula do Curso de Busca e Mineração de Texto*.

Xexéo, Geraldo, Fernando Morgado e Patrícia Fiuza (2009). "Differential Tag Clouds: Highlighting Particular Features in Documents". Em: *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 3, pp. 129–132. DOI: 10.1109/WI-IAT.2009.247.

YAKE! Keyword extraction from single documents using multiple local features (2020). *Information Sciences* 509, pp. 257–289. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.09.013. URL: https://www.sciencedirect.com/science/article/pii/S0020025519308588.

Zobel, Justin e Alistair Moffat (2006). "Inverted files for text search engines". *ACM computing surveys (CSUR)* 38.2, 6–es.