Capítulo 2

Processamento de Linguagem Natural em Textos de Mídias Sociais: Fundamentos, Ferramentas e Aplicações

Frances A. Santos[1], Jordan K. Kobellarz[2], Fábio R. de Souza[3], Leandro A. Villas[1] e Thiago H. Silva[2]

[1]Instituto de Computação, Unicamp, Campinas, Brasil

[2]Departamento Acadêmico de Informática, UTFPR, Curitiba, Brasil

[3]Faculdade de Filosofia, Letras e Ciências Humanas, USP, São Paulo, Brasil

{frances.santos,leandro}@ic.unicamp.br, thiagoh@utfpr.edu.br,

jordan@alunos.utfpr.edu.br, fabiorezende@usp.br

Abstract

Social media have proved to be an essential data source for analyzing social phenomena and their impacts in the real world due to the engagement and plurality of its users and the content shared by them. As the majority of content shared is textual, and we can find massive volumes of data, natural language processing techniques, jointly with machine learning models, are crucial to monitor the manifestation and evolution of these phenomena over time. This chapter presents an overview of the entire process of analyzing social media texts by computational means, from social media data extraction and main textual analysis techniques to examples of applications. Although this chapter focuses on social media texts and their particularities, most of the techniques presented can also be applied to other textual sources.

Resumo

As mídias sociais têm se mostrado uma importante fonte de dados para análise de fenômenos sociais e seus impactos no mundo real, devido ao engajamento e pluralidade de seus usuários e dos conteúdos compartilhados por eles. Como boa parte do conteúdo dessas fontes é textual e se tratando de um volume massivo de dados, torna-se fundamental o emprego de técnicas de processamento de linguagem natural e sua combinação com modelos de aprendizado de máquina para análises que permitam acompanhar a manifestação e mudança desses fenômenos ao longo do tempo. Este capítulo apresenta uma

visão geral de todo o processo de análise de textos de mídias sociais por meios computacionais, desde a extração de dados das mídias sociais e as principais técnicas de análise textual até exemplos de aplicações. Apesar deste capítulo ter foco em textos de mídias sociais e suas particularidades, boa parte das técnicas apresentadas podem ser também aplicadas em outras fontes textuais.

2.1. Introdução

Mídias sociais, tais como o Twitter, Reddit e Facebook são acessadas por aproximadamente 4,7 bilhões de pessoas em todo o planeta (i.e., 59% da população) [Kemp 2022]. Esses usuários, normalmente, engajam nas plataformas por muitas horas mensais e produzem e compartilham quantidades massivas de dados [Kemp 2022]. Por exemplo, anualmente os usuários do Twitter fazem cerca de 200 bilhões de postagens, o equivalente à 6 mil *tweets* por segundo [Stats 2022]. Aliado a isso, a maioria dessas mídias sociais disponibilizam meios para permitir o acesso programático a dados públicos em larga escala, seja por meio de APIs (*Application Program Interfaces*), aplicações de terceiros ou iniciativas de dados abertos (como descrito na Seção 2.2). Desta maneira, dados de mídias sociais têm sido amplamente utilizados por diversos estudos de fenômenos sociais, não somente no ambiente virtual onde são produzidos, mas também sendo traduzidos para o mundo real, uma vez que as mídias sociais implementam mecanismos que simulam interações sociais que acontecem *offline* [Silva et al. 2019]. Com isso, é possível inferir comportamentos *offline* a partir de "rastros" deixados de forma orgânica pelos usuários.

Nesse sentido, muito mais do que uma extensão do mundo real, tais mídias sociais podem ser consideradas como valiosas fontes de dados, sendo possível inferir comportamentos sobre sociedade urbanas em larga escala [Rogers 2009, Silva et al. 2019], contribuindo para diversas áreas de estudo, tais como a sociologia, psicologia, política, jornalismo, linguística, urbanismo, entre outros [Barbier and Liu 2011, Silva et al. 2019]. A indústria também tem tirado proveito de dados de mídias sociais para diversos propósitos, por exemplo, para conhecer melhor o perfil de clientes com maior potencial de compra, fazer comparação de desempenho com marcas concorrentes (*benchmarking*), inferir tendências de consumo (*forecasting*), personalizar recomendações de produtos e serviços, customizar e direcionar suas campanhas de comunicação de acordo com o perfil do cliente, entre várias outras possibilidades.

Dentre os possíveis tipos de mídia (texto, áudio, imagem, etc.) presentes em dados de mídias sociais, destaca-se a utilização de textos escritos em linguagem natural, por serem amplamente disponíveis na maioria delas. Por essa razão, neste capítulo apresentamos fundamentos teóricos de Processamento de Linguagem Natural (NLP, do acrônimo em Inglês de *Natural Language Processing*), aplicados aos desafios científicos e tecnológicos de lidar com textos gerados por usuários de mídias sociais, que podem ser explorados para desenvolver aplicações que se beneficiam do conhecimento extraído de tais textos. Como muitas técnicas, métodos e modelos de NLP são restritos de acordo com o idioma do texto, temos como foco neste capítulo textos de mídias sociais escritos em língua inglesa, por ser o idioma predominante na área de NLP e possuir o maior conjunto de recursos disponíveis. No entanto, sempre que possível, também são destacados recursos que são multilíngues, ou independentes de idioma, os quais podem ser explorados também em textos escritos em outros idiomas. Vale ressaltar também que, apesar do foco ser

em texto de mídias sociais, este material não é limitado somente a este tipo de conteúdo.

Com isso, este capítulo visa preparar o(a) leitor(a) para conhecer:

- As principais características de textos compartilhados por usuários em famosas mídias sociais, como coletá-los, assim como metainformações e respectivos desafios e limitações que cada uma dessas fontes de dados apresenta (Seção 2.2).
- Diversas técnicas para preparar os dados textuais, antes deles serem efetivamente utilizados por algum modelo de linguagem (Seção 2.3).
- Representações de textos utilizando vetores numéricos, chamados *embeddings*, capazes de capturar regularidades sintáticas e semânticas presentes nos textos (Seção 2.4).
- Diferentes métodos de modelagem e extração de conhecimento, incluindo técnicas de agrupamento de textos (clusterização), bem como a modelagem de estruturas semânticas latentes no texto, conhecida como extração de tópicos (Seção 2.5).
- Os conceitos de compreensão semântica e emocional, que abragem as tarefas de detecção de intenções, reconhecimento de entidades nomeadas, análises de sentimentos e emoções, ressaltando os desafios inerentes a elas (Seção 2.6).
- Aplicações reais, tais como a recomendação de rotas personalizadas para cidades inteligentes e o caso de uso na análise de situações politicamente polarizadas, que podem ser desenvolvidas com base no conhecimento semântico extraído a partir de dados de mídias sociais, considerando a utilização de diferentes técnicas apresentadas no minicurso (Seção 2.7).

Por fim, a Seção 2.8 traz as conclusões a respeito do tema.

2.2. Textos de mídias sociais: suas principais características e como coletá-los

Dados textuais em mídias sociais são conteúdos geralmente presentes em postagens e comentários, que podem tomar diferentes formas dependendo de como se dão as interações na respectiva plataforma em que foram produzidos, suas limitações e convenções. No Twitter[1], por exemplo, a plataforma permite um certo grau de anonimidade e as postagens são limitadas a 280 caracteres, por isso é comum que os usuários usem abreviações e adotem um tom informal, usando jargões e gírias comuns na *Internet*, incluindo muitas vezes construções ambíguas marcadas pelo sarcasmo [Tufekci 2014]. O Facebook[2], por sua vez, é uma mídia focada em interações entre pessoas reais, portanto, o não anonimato é uma característica marcante, ainda que haja incidência de contas falsas, muitas vezes focadas na manipulação da opinião pública [Weedon et al. 2017]. Já o Reddit[3], é uma mídia construída em torno de comunidades auto-organizáveis, onde os próprios membros ou moderadores são responsáveis por criar e executar as regras de moderação, portanto, a forma de interação depende diretamente da cultura de seus membros

[1]https://twitter.com. Último acesso em 05 de Agosto de 2022.

[2]https://facebook.com. Último acesso em 05 de Agosto de 2022.

[3]https://reddit.com. Último acesso em 05 de Agosto de 2022.

[Proferes et al. 2021]. Além de dados textuais, também é possível obter seus respectivos metadados, como informações pessoais do usuário, localização geoespacial de onde foi realizada a postagem, sinais sociais (como curtidas, votos ou reações), entre outros, que podem ser úteis em diferentes tipos de aplicações.

Nesta seção, são apresentadas as principais características de dados textuais disponíveis publicamente nas mídias sociais Twitter, Reddit e Facebook. Também são descritos os mecanismos de interação presentes em cada uma delas, os métodos para coletar dados em larga escala, o dicionário de dados e metadados disponíveis e, por fim, as limitações existentes.

2.2.1. Twitter

O Twitter é uma rede social *online* de *microblogging*, na qual os usuários podem postar mensagens com limite máximo de 280 caracteres, chamadas *tweets*. A plataforma do Twitter foi uma das primeiras a permitir acesso programático a dados públicos em larga escala por meio de API (acrônimo de *Application Program Interface*). Esse fato, tornou a utilização do Twitter bastante atrativo no meio acadêmico e na indústria [Tufekci 2014], dada a facilidade em se obter dados sobre eventos em tempo real ou, inclusive, dados históricos. Diferente de outras plataformas, como o Facebook, a maior parte dos dados do Twitter estão disponíveis abertamente na *web*, exceto informações de perfis privados (menos de 10% do total) ou mensagens privadas [Tufekci 2014].

Uma característica proeminente desta rede social é sua simplicidade, contando com algumas poucas funcionalidades, como (a) *hashtags*, palavras precedidas do símbolo #, usadas para demarcar um tópico de discussão, (b) *retweets*, ação de compartilhamento de mensagens, (c) menções, nome de usuários precedidos do símbolo @, quando um usuário marca outro em suas mensagens [Tufekci 2014], e (d) respostas, quando um usuário responde um *tweet*. Essas funcionalidades formam a base de como se dá a interação entre usuários dentro do Twitter e que possibilitam uma série de aplicações. Por exemplo, *hashtags* podem ser usadas para mapear assuntos importantes em um determinado local ou contexto, podendo representar a opinião de parte da população e, assim, sendo útil para planejamento de políticas públicas [Cody et al. 2015]. Outro exemplo, é a análise da rede de *retweets*, onde pode ser compreendido o comportamento de indivíduos em grupos homófilos e em uma situação politicamente polarizada, quando estes são expostos a conteúdos contrários ao seu viés político [Kobellarz et al. 2022]. A rede de menções, por sua vez, também pode ser uma importante fonte de informação em contextos políticos. Por exemplo, em [Conover et al. 2011], os autores identificaram que a rede de *retweets* em uma situação polarizada pode apresentar um alto grau de segregação entre grupos políticos opostos, enquanto a rede de menções não apresenta um padrão claro que possa ser ligado ao viés político.

Além dos dados textuais contidos nos *tweets*, também é possível obter seus metadados[4], tais como: o *timestamp* da postagem; os dados do perfil do usuário que fez a postagem, incluindo seu nome, localização, se é uma conta verificada pelo Twitter, quantidade de seguidores, amigos e postagens, língua do perfil, entre outros; as coordenadas

[4]https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-model/tweet. Último acesso em 05 de Agosto de 2022.

geográficas do *tweet* no formato GeoJson[5], que são reportadas pelo próprio usuário ou obtidas automaticamente pelo dispositivo que gerou o *tweet*; a associação do *tweet* a um determinado local (cidade, estado ou país); a quantidade de vezes que o *tweet* foi citado, respondido, compartilhado (*retweet*) ou curtido; e, as entidades presentes no *tweet*, como *hashtags*, *links*, usuários mencionados, endereço para mídia anexada na postagem, entre outros. Caso o *tweet* seja uma resposta a outro *tweet* ou um *retweet*, o identificador único da mensagem original e do seu respectivo autor são incluídos entre os metadados, permitindo obter facilmente os dados do *tweet* original programaticamente.

Data a vasta quantidade de pesquisas envolvendo dados do Twitter, suas limitações são bem conhecidas na literatura [Tufekci 2014]. A primeira é sobre o limite máximo de 280 caracteres por *tweet*, que pode restringir a capacidade argumentativa dos usuários em discussões. Essa limitação muitas vezes é contornada com a utilização de contrações de palavras, gírias da *Internet* e *emojis*, tornando complexa a tarefa de análise desses textos [Tufekci 2014]. Outra limitação é a representatividade dos dados, uma vez que a API do Twitter retorna dados parciais considerando sua relevância para uma dada consulta (como descrito na documentação[6]). Por isso, não é possível obter todos os *tweets* para uma consulta específica, comprometendo a replicabilidade do processo de obtenção de dados. Nesse sentido, a escolha do critério de filtragem deve considerar que a amostra não seja resultado da auto-seleção, ou seja, os critérios de filtragem não devem ser escolhidos com a intenção de forçar um resultado esperado por quem fez a seleção [Tufekci 2014]. Isto é uma implicação ética importante em estudos que usam dados do Twitter. Outra característica notável dessa fonte de dados é a incidência de contas robô, que em alguns casos podem comprometer a conclusão de resultados. Por exemplo, é reconhecido que a simulação de comportamentos simples no Twitter, como seguir contas e retuitar conteúdos de outros usuários, com o intuito de ser seguido de volta ou ser retuitado, são moedas sociais que podem tornar um robô tão influente quanto contas reais [Messias et al. 2013].

2.2.2. Reddit

O Reddit é uma mídia social composta por comunidades de discussão sobre temas específicos no formato de fóruns, chamados "*subreddits*". Em 2020, a plataforma tinha mais de 100 mil comunidades com 50 milhões de usuários ativos diariamente[7]. O uso de dados dessa plataforma se tornou bastante atrativo no meio acadêmico, uma vez que adotou um modelo similar ao do Twitter, disponibilizando dados abertamente por meio de uma API. Uma das características proeminentes são os mecanismos de moderação dos *subreddits*, que geralmente possuem regras claras que devem ser seguidas pelos seus membros, sob risco de serem penalizados pelos responsáveis pela comunidade, assim como por outros membros que podem votar nos comentários mais relevantes. Além da moderação, mecanismos de recompensa possibilitam a bonificação de membros que respeitam as regras e contribuem ativamente nas comunidades. Essas características, em conjunto, permitem uma maior riqueza nos dados tanto quantitativamente quanto qualitativamente [Proferes et al. 2021].

[5]https://geojson.org. Último acesso em 05 de Agosto de 2022.

[6]https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/overview. Último acesso em 05 de Agosto de 2022.

[7]https://www.redditinc.com/advertising/audience. Último acesso em 05 de Agosto de 2022.

A API do Reddit permite acesso à qualquer informação disponível publicamente na plataforma, incluindo postagens, comentários, perfis, comunidades e suas respectivas metainformações. É possível capturar dados históricos e em tempo real, similar a API do Twitter, mas com a vantagem de permitir a recuperação do histórico completo, diferentemente do Twitter que foca na relevância - essa característica torna o procedimento de coleta de dados no Reddit replicável, algo desejável em pesquisas acadêmicas. O dicionário de dados do Reddit também é bastante rico, por exemplo, além do conteúdo da postagem, é possível recuperar todos os comentários incluindo seu conteúdo completo e respectivos metadados, com informações do autor (nome, pontuação na plataforma, se possui e-mail verificado, entre outros[8]), informações sobre a pontuação do comentário, assim como a quantidade de votos positivos e negativos; *flag* apontando se o comentário foi editado pelo usuário ou destacado pelos moderadores; a árvore de respostas em cascata gerada por um comentário; entre outros dados.

Dentre os desafios ao trabalhar com dados do Reddit, está o fato da plataforma permitir um alto grau de anonimidade, possibilitando que usuários mal-intencionados se comportem de forma antiética em comunidades que tenham regras menos restritivas [Proferes et al. 2021]. Por exemplo, para fazer parte da plataforma, basta cadastrar um nome de usuário e e-mail, sem a necessidade de verificação, inclusive, é encorajado o uso de pseudônimo para manter a privacidade do perfil [Proferes et al. 2021]. Outro desafio está relacionado às comunidades serem regidas pelos seus próprios membros, consequentemente, culminando em práticas de moderação distintas. Isso torna complexa a comparação entre comunidades, mesmo utilizando os mesmos critérios para estudar um fenômeno em comum [Proferes et al. 2021]. Quanto aos conteúdos em texto, a liberdade dada no formato, que pode inclusive ter *tags Markdown* ou *html*, ao mesmo tempo que trazem riqueza de detalhes, também tornam a limpeza dos dados mais complexa. Outra característica limitante é o fato de *bots* terem uma grande representatividade em algumas comunidades, podendo atuar como criadores e curadores de conteúdos e, inclusive, como moderadores para automatizar a aprovação de postagens em *subbreddits*.

2.2.3. Facebook (Meta)

O Facebook, um dos produtos da Meta, é uma rede social pautada em interações entre pessoas reais. Diferentemente do Reddit e Twitter, o alto grau de controle de privacidade das contas nessa rede é uma característica marcante e o anonimato é desencorajado. Medidas restritivas são aplicadas para coibir comportamentos nocivos na plataforma, como os ligados à manipulação da opinião pública por meio do compartilhamento de informações falsas ou desinformação [Weedon et al. 2017]. Apesar da plataforma do Facebook disponibilizar APIs públicas para a criação de aplicações, o acesso aos dados é limitado àqueles sobre os quais o usuário autenticado tem permissões para gerenciar, não sendo possível a extração de dados públicos em larga escala.

Alternativamente, é possível coletar dados sem usar uma das APIs oficiais do Facebook, por exemplo, por meio do programa *Social Science One*[9], que permite estabelecer uma parceria entre instituições de ensino e o Facebook, para ter acesso direto à sua base

[8]https://praw.readthedocs.io/en/stable/code_overview/models/comment.html. Último acesso em 05 de Agosto de 2022.

[9]https://socialscience.one/grant-process. Último acesso em 05 de Agosto de 2022.

de dados. Apesar disso, a aplicação no programa demanda uma série de etapas legais, além de impor limitações significativas no nível de dados que podem ser acessados. Outra possibilidade, é realizar o recrutamento de voluntários para preencherem pesquisas ou cederem seus dados, utilizando aplicações de terceiros que se conectam às APIs oficiais. Esse método apresenta uma restrição considerável na abrangência do perfil e quantidade de usuários, limitando os dados aos usuários que foram recrutados e cederam seus dados para pesquisa. A raspagem de dados (ou, *scraping* em Inglês) também é uma das alternativas para coleta de dados, contudo esse procedimento é explicitamente proibido pelas políticas da plataforma, assim como é desencorajado por meio de sistemas *anti-bot* que limitam esse método.

Recentemente, uma iniciativa da Meta chamada CrowdTangle[10], permitiu a pesquisadores e empresas terem acesso a grande parte dos dados abertos do Facebook de forma gratuita, incluindo postagens de todas as páginas públicas com pelo menos 50 mil curtidas, grupos com pelo menos 95 mil membros, grupos dos Estados Unidos com pelo menos 2 mil membros e de todos os perfis verificados na plataforma. Por meio do CrowdTangle, também é possível coletar dados do Instagram, outra rede social da Meta, focada em postagens no formato de imagem e vídeo. Por brevidade, focaremos a seguir somente na rede social do Facebook. Dentre os dados que podem ser coletados do Facebook por meio do CrowdTangle estão: a postagem em texto, imagem ou vídeo, *timestamp* da postagem, tipo da postagem (vídeo, imagem ou texto), a página, perfil ou grupo em que foi feita a postagem, contagem de interações sociais (i.e., a quantidade de curtidas, reações segmentadas por tipo, comentários, compartilhamentos e visualizações), e quais páginas públicas ou perfis compartilharam a postagem. Mais informações podem ser encontradas na documentação da ferramenta[11].

Os dados do CrowdTangle podem ser obtidos programaticamente via API ou por meio da interface gráfica da própria ferramenta, que também permite a criação de *dashboards* personalizados para análise de postagens em tempo real. Seu sistema de busca é bastante robusto, permitindo desde uma busca básica por *hashtags* ou ocorrência de palavras-chave em textos de postagens, até busca textual em imagens. Apesar disso, o nível de informação retornado é limitado: diferentemente do Twitter e Reddit, não é possível obter informações sobre quem reagiu ou visualizou uma postagem, assim como nenhuma informação demográfica sobre usuários, como idade ou localização. Comentários em postagens também não estão disponíveis. Outra limitação, é o fato dos dados que podem ser obtidos serem produzidos apenas por contas famosas ou verificadas, o que inviabiliza o estudo do comportamento de postagem de públicos menos representativos no Facebook. Todas essas limitações são esforços focados na proteção da identidade dos usuários, característica fundamental dessa fonte de dados, aliás, casos emblemáticos de violação da privacidade de usuários do Facebook ficaram bastante conhecidos, como o do *"Tastes, ties, and time"* ou 3T e o da *Cambridge Analytica*, duas situações nas quais, mesmo anonimizando a identidade dos usuários, foi possível reconstituir as informações e expor seus perfis [Zimmer 2020].

[10]https://crowdtangle.com. Último acesso em 05 de Agosto de 2022.

[11]https://help.crowdtangle.com/en/articles/4201940-about-us. Último acesso em 05 de Agosto de 2022.

2.3. Pré-processamento Textual

Modelos computacionais voltados à compreensão de linguagem natural não são capazes de reconhecer o significado abstrato que atribuímos às palavras que conhecemos, tampouco seus atributos semânticos, culturais ou históricos. Normalmente, tais modelos são projetados para inferir o significado das palavras e expressões por meio da *distribuição estatística* em que ocorrem em um texto, ou seja, as janelas contextuais e a frequência em que ocorrem.

Para que isso ocorra, cada item léxico é tratado como um *índice* contendo um valor, dentro de um *vetor* numérico. Este vetor, por sua vez, representa o contexto em que tal palavra está inserida (uma sentença ou documento, por exemplo), e o valor numérico atribuído a cada palavra pode indicar, dentre outros: a ocorrência ou não de determinadas palavras de interesse em um documento, a sua distribuição relativa em um conjunto de vários documentos ou, até mesmo, a relação léxica entre palavras e sentenças.

A etapa de pré-processamento é necessária para garantir que, ao ser atribuída uma representação numérica para cada palavra de uma sentença, mantenhamos apenas os conteúdos e características desejáveis para a análise computacional do *corpus* com o qual se trabalhará. Desta forma, a complexidade do domínio do problema é reduzida, eventuais ruídos podem ser removidos, aumentando as chances de sucesso do projeto como um todo. Não existe, no entanto, uma fórmula única e precisa que funcione em qualquer projeto ou contexto: as seções a seguir apresentam, como referência, as técnicas mais comuns e amplamente utilizadas pela academia e indústria para preparação dos dados textuais, antes de serem efetivamente analisados por algum modelo computacional. Deve-se considerar que o emprego ou não de cada técnica deve levar em conta uma análise profunda do conteúdo textual a ser investigado.

O pré-processamento, normalmente, é a segunda etapa em um projeto de NLP, vindo logo após a obtenção dos dados. Nessa etapa os dados textuais são estruturados considerando o domínio e método que será aplicado.

2.3.1. Compreensão do domínio

Antes de qualquer tarefa de manipulação de dados textuais, é importante que o domínio de origem do texto seja bem compreendido, e que se tenha clareza dos objetivos pretendidos com a sua análise através de um modelo computacional. Em relação ao domínio de origem: textos de notícias, por exemplo, tendem a adotar uma linguagem formal, com poucos marcadores de expressão (i.e., exclamações, interrogações, repetições, etc.), enquanto textos de redes sociais podem conter gírias particulares, *emojis* e *hashtags* que podem ser relevantes para o objetivo do projeto. Caso um modelo computacional seja aplicado sobre textos do primeiro caso, caracteres especiais poderiam ser removidos sem impacto na compreensão de seu conteúdo. No segundo caso, no entanto, a simples remoção de caracteres especiais poderia ser bastante prejudicial, pois acabaria removendo outros elementos que contribuem na compreensão desse tipo de texto.

Além da compreensão do domínio, o entendimento do método que será aplicado é de vital importância na escolha das técnicas de pré-processamento. Por exemplo, em um projeto no qual o objetivo é identificar postagens similares ou tópicos latentes em um conjunto de postagens, é importante que a etapa de pré-processamento preserve palavras

representativas para a identificação de tópicos e elimine aquelas que possam trazer ruído. Nesse caso, é importante destacar palavras que ajudem a inferir regularidades entre as postagens, e remover ou penalizar outras que ocorram com alta frequência no conjunto de dados, mas que não colaboram na tarefa.

Outro exemplo seria a análise de sentimentos ou identificação de toxicidade em postagens. Em casos mais simples, o sentimento ou a toxicidade pode ser inferida por meio do uso de dicionários que mapeiam palavras às suas respectivas valências (i.e., polaridade negativa ou positiva). Nesse tipo de aplicação, pontuações, palavras comuns na gramática da língua do texto e qualquer outra palavra que não possua ocorrência no dicionário, poderiam ser removidas sem impactar no resultado. Contudo, esse método não captura sutilezas que podem ser importantes em determinados tipos de aplicações, como, por exemplo, quando é necessário que o modelo contemple a relação entre palavras ou frases para identificar sentimentos ou características tóxicas em conteúdos compostos por sarcasmo ou figuras de linguagem. Nesse caso, boa parte das características textuais precisam ser preservadas para a aplicação de modelos de aprendizado de máquina modernos, que consigam capturar essas relações no texto (alguns modelos serão apresentados no decorrer das seções 2.4 e 2.5).

2.3.2. *Tokenização*

Até essa seção, nos referimos às unidades básicas presentes em textos usando o termo "palavras", "símbolos", entre outros. Apesar disso, existe uma designação apropriada no contexto de NLP para se referir de forma genérica a essas unidades: *tokens*.

O termo *token*, derivado das linguagens formais de programação, consiste em uma sequência de caracteres dentro de um documento textual que apresente algum tipo de unidade semântica [Schütze et al. 2008]. No contexto da NLP, um *token* pode ser representado por um caractere ou uma sequência de caracteres, ou mesmo um símbolo, pontuação, número, *emoji*, *hashtag*, menção, entre outros. O processo usado para transformar um documento textual em uma lista composta por esses elementos é chamado de *tokenização*.

Considere um *corpus* formado pelas últimas palavras de Sócrates, segundo relato de Platão no diálogo *Fédon* (retirado de [Pombo 2022]):

> *"Críton, somos devedores de um galo a Asclépio; pois bem pagai a minha dívida, pensai nisso"*

Após o processo de *tokenização* da sentença original obteríamos a seguinte lista de *tokens*:

> *["Críton", ",", "somos", "devedores", "de", "um", "galo", "a", "Asclépio", ";", "pois", "bem", "pagai", "a", "minha", "dívida", ",", "pensai", "nisso"]*

2.3.3. Normalização de texto

Após a transformação do *corpus* em uma lista de *tokens*, passamos por uma série de tarefas cujo objetivo é reduzir ao mínimo possível a complexidade do vocabulário com o qual trabalharemos. Isto é importante para aumentar a probabilidade de que o método aplicado nas etapas posteriores tenha maior chance de êxito na tarefa proposta.

Reduzir a variabilidade do *corpus* envolve, por um lado, remover termos que não possuem nenhum tipo de relevância para o domínio analisado (o que discutiremos nas etapas a seguir), e por outro, agrupar elementos idênticos ou similares, que estejam apresentados de forma distinta. Uma das formas mais simples e efetivas de fazê-lo é através da normalização da caixa das palavras presentes no vocabulário. Este processo consiste em transformar todas as palavras em minúsculas ou maiúsculas. Trata-se de uma funcionalidade computacionalmente muito simples, normalmente incorporada como uma função nativa das principais linguagens de programação (como o *.lower()* e *.upper()*, do *Python*, e o *.toLowerCase()* e *.toUpperCase()*, do *Javascript*, que uniformizam os *tokens* do *corpus* em palavras minúsculas ou maiúsculas, respectivamente).

Suponha que, analisando textos de redes sociais num domingo à tarde, nos deparemos com comentários a respeito de resultados de uma partida de futebol: um torcedor mais empolgado pode comemorar a pontuação de seu time escrevendo "GOL !!", enquanto outro, desapontado ao final da partida, escreve apenas "Perdemos por causa de um gol". Ou que ainda um outro, atendo-se apenas aos fatos, escreva "Gol aos 45 do segundo tempo". A mesma palavra, então, assume três grafias distintas ("GOL", "Gol", "gol"). Caso estes exemplos sejam submetidos a um algoritmo computacional da maneira como originalmente se apresentam, serão consideradas por este algoritmo como três *tokens* distintos. O processo de normalização fará com que este sistema computacional possa reconhecer, devidamente, as três como ocorrências de um único *token*, o que lhe permitirá atribuir a relevância devida deste *token* na compreensão do conteúdo textual analisado.

Há de se considerar, no entanto, que dada a natureza expressiva dos discursos presentes nas redes sociais, é possível que, além das exemplificadas, outras grafias possam ser adotadas ("GOOOOL!", por exemplo). Estas, no caso, não são resolvidas simplesmente pelo processo de normalização. Trata-se de outro desvio que pode ser contornado em outros níveis de análise. Uma destas formas, discutida a seguir, seria através da aplicação de expressões regulares.

Retornando ao nosso exemplo, após o processo de normalização para letras minúsculas, obteríamos o seguinte resultado:

> *["críton", ",", "somos", "devedores", "de", "um", "galo", "a", "asclépio", ";", "pois", "bem", "pagai", "a", "minha", "dívida", ",", "pensai", "nisso"]*

2.3.4. Expressões Regulares (Regex)

Para prosseguir na limpeza e pré-processamento de texto, garantindo que nenhuma informação relevante seja perdida, cabe agora uma análise do domínio semântico coberto pelo *corpus* utilizado. Algumas perguntas que podem ajudar nesta análise: Qual a natureza do *corpus* a ser investigado? É composto por textos formais (e.g., documentos, artigos, etc.), ou informais (e.g., textos de redes sociais, fóruns, *chats*)?

Caso sejam textos em linguagem formal, de notícias por exemplo, não é comum que se encontre *emojis* ou erros de digitação, e toda pontuação é utilizada de maneira ponderada. Uma ferramenta que permita remover todos os caracteres especiais e mantenha apenas o texto pode ser muito útil. Caso sejam textos de postagens de mídias sociais,

pode ser necessário manter alguns padrões de caracteres que correspondam aos *emojis*, removendo apenas os *links* por exemplo. Toda manipulação (e.g., contagem, substituição ou remoção de padrões) de texto apenas se faz possível a partir de uma funcionalidade que permita, de maneira preliminar, *encontrar* tais padrões no texto: ao conjunto de comandos que permitem este nível de manipulação de texto se dá o nome de Expressões Regulares.

As Expressões Regulares (do inglês, *Regular Expressions*, cujos acrônimos são *RE* ou *RegEx*), são comandos que especificam padrões de busca em texto [Jurafsky and Martin 2021]. As expressões regulares foram propostas inicialmente na década de 1950 por [Kleene et al. 1956] como uma notação algébrica para a representação de eventos nos primeiros modelos de redes neurais computacionais de McCulloch-Pitts, tendo sido adaptadas para uma ferramenta de busca de texto em linguagem natural por [Thompson 1968]. É ainda hoje incorporado aos editores de texto dos principais sistemas operacionais. A maior parte das linguagens de programação, do C++ ao Python, possuem suporte para manipulação de texto a partir de expressões regulares.

Voltando ao exemplo relacionado à palavra *gol*: suponha que queiramos encontrar todas as ocorrências desta palavra dentro de um *corpus* de textos de redes sociais (já normalizado). Para isso, basta indicar este padrão dentro de duas barras invertidas: $/gol/$. Para encontrarmos todas as variações em que a letra *o* dentro da palavra venha ser repetida (como no exemplo "goool !"), acrescentamos ao nosso padrão um quantificador $*$ à letra *o*: $/g[o]*l/$. Agora, suponha que estejamos lidando com um *corpus* de textos do *Twitter*, e queiramos encontrar todas as referências a *usernames* (iniciados por @), podemos utilizar o comando $/@(\backslash w\{1,15\})/$, que traduzindo, significaria: buscar elementos iniciados por @, seguidos por qualquer sequência entre 1 a 15 caracteres alfanuméricos. Uma vez que identificados, os *usernames* particulares podem ser substituídos por um *token* único.

As bibliotecas de manipulação de Expressões Regulares, como a *RE* do Python, permitem facilmente a manipulação de texto. No caso do *Twitter*, sabemos que as menções a usuários são sempre precedidas de "@", e que as *hashtags* são iniciadas por "#". Neste caso, a substituição dos *tokens* correspondentes a nomes de usuários por um *token* único (como, por exemplo, *@USUARIO*), permite melhorar a qualidade da análise computacional de texto, por permitir contar a quantidade de referências a usuários, independentemente de quem seja, analisar os contextos de sentenças em que referências a usuários aparecem e, ao mesmo tempo, anonimizando a identidade real destes, o que garante a privacidade dos dados analisados.

Existem algumas classes de comandos de Expressões Regulares, como os marcadores, os quantificadores e as âncoras. Os marcadores são aqueles responsáveis por encontrar determinados padrões num documento textual; os quantificadores, por sua vez, permitem que se especifique a quantidade de vezes que tais padrões devem ser buscados. Já as âncoras, são os delimitadores de início e fim de onde tais padrões devem ser encontrados.

Abaixo, alguns exemplos de comandos marcadores (adaptado de [Jurafsky and Martin 2021]). Em sequência, alguns exemplos de quantificadores e âncoras, que combinados aos marcadores, permitem que se especifique a quantidade e as condições em que os padrões devem ser encontrados:

Padrão	Descrição
\w	Qualquer símbolo alfanumérico
\W	Qualquer elemento não-alfanumérico
\s	Espaçamento (Espaço em branco)
\S	Espaço que não esteja em branco
\d	Qualquer dígito
\D	Qualquer símbolo diferente de dígito

Padrão	Descrição
()	Parênteses delimitando uma sub-expressão que deve ser encontrada
* + ? {}	Contadores (número de vezes em que a expressão anterior deve se repetir)
^$	Indicação de início e fim de sentença
|	Disjunção ("Ou"), indicando que um dos conjuntos deve ser encontrado

Expressões regulares, no entanto, podem se tornar muito complexas pelo mesmo motivo que são versáteis: o fato de poderem ser combinadas de maneira irrestrita e ilimitada. Não é necessário, contudo, decorar padrões de expressões regulares: diversas páginas, como o Regex101 [Dib 2022] possuem guias de referência e ambiente para montar e testar os padrões que possam ser úteis em cada caso

Retornando ao nosso exemplo da clássica enunciação em *Fédon*, suponhamos que queiramos manter apenas as palavras, removendo toda a pontuação e caracteres especiais. Utilizando o comando $[\backslash w+]$, obteríamos o seguinte conjunto de *tokens* como resultado:

> *[“críton”, “somos”, “devedores”, “de”, “um”, “galo”, “a”, “asclépio”, “pois”, “bem”, “pagai”, “a”, “minha”, “dívida”, “pensai”, “nisso”]*

2.3.5. Remoção de *stop-words*

Por causa da natureza da sintaxe da língua natural humana, quando nos comunicamos, fazemos uso de dois tipos de palavras: as palavras que carregam o conteúdo de determinada mensagem que queremos transmitir (os verbos, substantivos, adjetivos e advérbios); e as palavras de função, que não carregam por si significado relevante, mas são necessárias para a manutenção da gramaticalidade daquilo que se enuncia. Nesta segunda categoria, se encontram, por exemplo, os artigos, as conjunções e as preposições.

Uma análise da frequência (o número de ocorrência de palavras individuais em um dado texto), normalmente, aponta para o fato de que as palavras de função ocorrem com muita frequência, pelo fato de estarem presentes em quase todos os enunciados, independente de seu conteúdo. Trata-se de um padrão que pode ser observado em praticamente todos os idiomas da língua humana (inclusive, trata-se de um dado provado empiricamente, por aquela que ficou conhecida como Lei de Zipf - ler mais em [Zipf 2016]).

As palavras de conteúdo, por sua vez, são mais raras, e muito associadas à mensagem que se deseja transmitir. Num algoritmo para análise de sentimentos, por exemplo, pode ser interessante encontrar palavras que carreguem uma conotação positiva, que embora raras, podem estar associadas a elogios - enquanto palavras de conotação negativa podem estar associadas a críticas.

Remover ou não *stop-words* é uma decisão que deve levar em conta não mais o *corpus*, mas principalmente o algoritmo a ser utilizado para classificação ou análise. A remoção de *stop-words* é muito associada a algoritmos de aprendizado de máquina supervisionados mais tradicionais, como Naïve Bayes e Regressão Logística, que podem levar em conta a frequência dos termos presentes no *corpus* para determinar a classe a qual cada texto pertence, ou métodos de representação baseados em *Bag Of Words* (sobre os quais falaremos na próxima Seção).

Um exemplo de caso em que a remoção de *stop-words* não se faz necessária: para algoritmos baseados em redes neurais recorrentes, como as do tipo LSTM (*Long-Short Term Memory*), por exemplo, a remoção de *stop-words* pode não vir a se fazer necessária, pois o aprendizado é baseado na análise da sequência de palavras - e a termos muito frequentes são automaticamente atribuídos pesos baixos para a computação de significado.

Caso se decida pela remoção de *stop-words*, é necessário um cuidado em especial: a palavra "não" está geralmente contida em listas de *stop-words* a serem removidas (estando, por exemplo, nas *stop-words* nativas do português na biblioteca NLTK, do Python [Bird 2006]). No entanto, a remoção de um "não" em uma sentença pode, literalmente, inverter o seu significado. Portanto, uma boa prática é manter as negações ainda que se removam as outras *stop-words*.

Retornando ao nosso exemplo, caso queiramos remover as *stop-words* da sentença, manteríamos a seguinte lista de *tokens*:

> *["críton", "devedores", "galo", "asclépio", "pois", "bem", "pagai", "dívida", "pensai", "nisso"]*

Nesse caso, os *tokens* removidos foram:

> *["somos", "de", "um", "a", "a", "minha"]*

2.3.6. Lematização e Stemização

A lematização e a *stemização* (adaptados dos termos em inglês, *lemmatization* e *stemming*) são outros dois processos para diminuição da complexidade e variabilidade de um *corpus*. Ambos consistem em análises léxicas feitas com o auxílio de dicionários específicos da língua no *corpus* ao qual serão aplicadas.

A lematização consiste num processo para encontrar o *lema* de cada palavra de um *corpus*, ou seja, sua forma morfológica semântica e sintaticamente canônica [Indurkhya and Damerau 2010]. A lematização segue a premissa de que, ao se reduzir todas as inflexões possíveis que as palavras podem assumir (como, por exemplo, em relação a tempos verbais, gênero, número ou grau), o vocabulário do *corpus* é simplificado, facilitando a capacidade preditiva dos modelos de aprendizado de máquina a serem aplicados. Através da lematização, todas as ocorrências de variações do verbo *estar* (como, por exemplo, "estou", "estaremos", "estará"), seriam substituídas pela forma canônica do verbo.

Já o processo de *stemização*, trata-se de um algoritmo que reduz as palavras ao seu *radical* (ou *tema*). Em relação ao nome do processo, opta-se por manter a denominação derivada do inglês apenas para facilitar a referência a literatura relacionada em domínio computacional. No processo de *stemização*, são eliminados quaisquer afixos

agregados ao radical de uma palavra dentro de uma sentença (prefixos ou sufixos). A *stemização* é feita a partir de algoritmos de busca, que procuram pela ocorrência de determinados radicais que podem ocorrer em cada palavra. Trata-se, portanto, de um algoritmo mais simples, que apresenta algumas desvantagens em relação à lematização [Indurkhya and Damerau 2010], por reduzir a legibilidade, e, ao mesmo tempo, ser mais "agressivo" e, por vezes, excessivamente generalizador. Por exemplo, o *Porter Stemmer* [Porter 1980] reduz *organization* para *organ*) [Jurafsky and Martin 2021]. Esta imprecisão se deve, em parte, por desconsiderar que, devido ao caráter dinâmico e evolutivo das línguas, existam palavras que compartilhem um radical, mas ao longo do tempo ganhem significados próprios incompatíveis entre si.

Algoritmos de lematização e *stemização* partem de um processo heurístico, específico a cada linguagem. Ambas as tarefas possuem implementações no português brasileiro (como o lematizador LemPORT [Rodrigues 1997] e o RSLP Stemmer do NLTK [Bird 2006]), mas pode ser uma limitação quando aplicado a idiomas que possuam poucos recursos computacionais disponíveis.

Retornando ao exemplo, ao aplicarmos um lematizador, obteríamos o seguinte resultado sobre os *tokens* em questão:

> *[("críton", "críton"), ("devedores", "devedor"), ("galo", "galo"), ("asclépio", "asclépio"), ("pois", "pois"), ("bem", "bem"), ("pagai", "pagar"), ("dívida", "dívida"), ("pensai", "pensar"), ("nisso", "nisso")]*

2.3.7. N-gramas

Os *n-gramas* são, basicamente, sequências de *n* palavras. Por exemplo, caso queiramos capturar todas as sequências de duas palavras (2-gramas, ou "bigramas") em um enunciado como "eu quero água gelada", obteríamos as expressões:

> *<"eu, quero">, <"quero, água">, <"água, gelada">*

A análise de *n-gramas* pode ser bastante útil para entender as formas de expressão mais frequentes dentro de um *corpus*. Ferramentas de *n-gramas*, estão por trás das técnicas mais tradicionais para produção de modelos de linguagem, que permitem estimar a probabilidade de ocorrência de uma palavra em uma sequência, que é muito difundida em sistemas de reconhecimento de fala e para sugestão de digitação nos teclados de celulares.

Trata-se, portanto, de uma técnica bastante simples, que, ao mesmo tempo, facilita uma análise do comportamento agregado do conteúdo de um volume grande de texto. Em alguns casos, pode ser utilizado como uma tarefa adicional de limpeza de dados, onde a análise da frequência dos *n-gramas* de um determinado texto, pode ser utilizado como mais uma ferramenta de limpeza possível e, assim, sequências muito frequentes podem ser tratadas como *stop-words*, e sequências muito raras podem ser tratadas como ruído. O uso, mais uma vez, deve partir de uma análise do contexto do *corpus* no qual empregado.

2.3.8. *POS-Tagging*

As *POS-Tags* (do inglês *Part-of-Speech*, ou Partes da Fala em tradução livre), realizam a categorização da função gramatical de cada palavra na sintaxe de uma sentença. Em

outras palavras, qual o "papel" de cada palavra na construção de uma sentença. São as categorias como adjetivos, verbos e substantivos.

A lista de *POS-Tags* mais comuns são (Adaptado de [Jurafsky and Martin 2021]): (*i*) ADJ: Adjetivo; (*ii*) ADP: Preposição (do inglês *Adposition*); (*iii*) ADV: Advérbio; (*iv*) NOUN: Substantivo; (*v*) VERB: Verbo; e, (*vi*) PROPN: Nomes próprios.

Existem duas classes principais de *POS-Tags*: as classes fechadas, que contém um número fixo e geralmente fechado dentro de uma língua, como as preposições e artigos; e as classes abertas, como os nomes e verbos, os quais são geralmente os mais afetados pela criatividade da língua humana, sendo impossível mapear todas as possibilidades de ocorrências de termos que podem caber nestas classes.

A obtenção das *POS-Tags* pode ser um indicador útil dependendo do contexto onde aplicado. Ainda que sejam atributos puramente gramaticais, as *POS-Tags* estão geralmente relacionadas à função semântica dos termos em uma frase. Por exemplo, em um contexto em que se deseje conhecer a impressão dos usuários em relação a um determinado produto ou serviço, pode ser útil manter aquelas palavras cujas *POS-Tags* indiquem serem adjetivos (rotulados como ADJ), ao passo que os substantivos (indicados como NOUN) podem ser úteis para uma etapa posterior de compreensão e limpeza do texto (através do reconhecimento de entidades nomeadas, assunto que será apresentado na Seção 2.6) [Jurafsky and Martin 2021, Indurkhya and Damerau 2010].

Existem diversos algoritmos para geração de *POS-Tags*, como *Hidden Markov Models* e *Conditional Random Fields*, sobre os quais se pode ler mais em [Lafferty et al. 2001]. Assim como em relação aos lematizadores, citados anteriormente, já existem implementações de algoritmos para detecção de *POS-Taggings* no português brasileiro [Honnibal et al. 2020, Bird 2006].

Essas seriam as classes gramaticais retornadas para cada *token* em nosso exemplo ao ser aplicado um algoritmo de *POS-Tagging*:

> *[("críton", "PROPN"), (",", "PUNCT"), ("somos", "AUX"), ("devedores", "ADJ"), ("de", "ADP"), ("um", "DET"), ("galo", "NOUN"), ("a", "ADP"), ("asclépio", "PROPN"), (";", "PUNCT"), ("pois", "ADV"), ("bem", "ADV"), ("pagai", "ADJ"), ("a", "DET"), ("minha", "DET"), ("dívida", "NOUN"), (",", "PUNCT"), ("pensai", "VERB"), ("nisso", "PRON")]*

2.3.9. Codificação de Caracteres

A forma escrita das línguas de origem latina (como o português e o espanhol) e anglo-saxônica (como o inglês), é baseada no alfabeto de 26 letras, tal qual utilizado neste texto. Ainda assim, existem variações da escrita do português e espanhol que não estão presentes no inglês, como a acentuação e a cedilha. Uma vogal acentuada, como "ã", é por sua vez, reconhecida por um sistema computacional como um caractere diferente de "a", e assim ocorrendo para qualquer outro acento ou variação que sofra.

Indo um pouco além, como se sabe, existem inúmeras outras alternativas de escrita além do alfabeto tal qual nos acostumamos. Podemos citar o alfabeto cirílico, o grego, as vogais vazias do norueguês, e ainda as formas de escrita do árabe e hebraico. Indo além,

o mandarim e o japonês, algumas das línguas mais faladas do planeta, não são baseadas em um alfabeto, mas em ideogramas, cuja variabilidade e significação é muito ampla do que as letras do alfabeto quando analisadas por si só.

Computacionalmente, cada um dos caracteres que pode compor uma língua deve ser representado em uma codificação única, compatível com a tipografia adotada. Considerando que a grande parte das ferramentas de processamento de linguagem natural é ainda na atualidade baseada no inglês, é muito comum que, ao inserirmos caracteres comuns ao português brasileiro, ocorra um erro de processamento ou perda do caractere, caso a codificação adotada seja incompatível com a aparição de caracteres acentuados.

Existe uma série de codificações possíveis, como Unicode, ASCII e UTF-8, que garantem a conversão correta entre o termo utilizado e um endereço de memória no sistema computacional. Portanto, é necessário utilizar uma codificação compatível com a língua sendo processada, para garantir o processamento correto pelos algoritmos que serão utilizados em seguida e evitar a perda de informação. Para conhecer mais a respeito de codificação, recomendamos a leitura de [Indurkhya and Damerau 2010].

Os métodos e técnicas descritos acima são, em geral, os mais utilizados e difundidos para o pré-processamento de texto. No entanto, diversos outros, mais específicos, podem ser aplicados a depender do contexto. Por exemplo, métricas de distância de edição, como o algoritmo de Levenshtein, para encontrar palavras parecidas e, através disso, detectar possíveis erros de digitação - ler mais em [González-Bailón and De Domenico 2021]; alguma versão do *Flesch Reading Score* do inglês, para remoção de eventuais palavras sem sentido, comuns em redes sociais [Fleiss et al. 1981]; bem como corretores textuais, como os disponibilizados pela biblioteca *Spacy* [Honnibal et al. 2020]. O julgamento da necessidade do uso de técnicas adicionais para o pré-processamento, assim como para qualquer outra técnica apresentada, depende da avaliação de sua necessidade dentro do contexto empregado.

2.4. Representação de Textos Utilizando Vetores Numéricos

Como mencionado na seção anterior, os dados textuais devem ser previamente pré-processados antes de poderem ser utilizados por algum modelo de linguagem. Esse procedimento, no entanto, não gera representações que podem ser compreendidas por um modelo computacional. Tais modelos conseguem lidar apenas com dados numéricos e não com textos em seu formato original. Nesse sentido, uma das formas mais simples de criar uma representação numérica é por meio do chamado *one-hot encoding*, que nada mais é do que representar um texto por meio de uma matriz de valores binários, em que cada linha representa a ocorrência de um *token* no texto.

O método consiste em usar uma representação binária única de tamanho N para cada *token* no vocabulário, sendo N o tamanho do vocabulário. Dessa forma, os documentos são representados por uma matriz de N colunas por M linhas, representando a ocorrência de cada *token* no texto. Por exemplo, um vocabulário com três palavras "está", "tudo" e "bem", poderia ser representado pelos vetores $[1,0,0]$, $[0,1,0]$ e $[0,0,1]$, respectivamente. Dessa forma, a frase "está bem" seria representada pela matriz $[[1,0,0], [0,0,1]]$, enquanto a frase "tudo bem" seria representada pela matriz $[[0,1,0], [0,0,1]]$.

Apesar da simplicidade, existem duas grandes desvantagens em se utilizar tal re-

presentação: (*i*) os vetores formados são esparsos, sendo que se o vocabulário contiver *N* *tokens*, cada vetor terá *N* dimensões com apenas uma posição com valor 1, demandando uso intensivo de memória; (*ii*) os vetores numéricos resultantes não consideram características das palavras que podem ser importantes para o domínio, por exemplo, a relevância da palavra em relação às outras de acordo com sua incidência no *corpus*, a similaridade de uma palavra com outras no *corpus*, ou a relação de uma palavra com seu contexto, o que pode impactar negativamente na capacidade do modelo de linguagem em identificar regularidades léxicas e/ou semânticas presentes nos textos.

Por isso, nesta seção é introduzido o conceito de *Embeddings*, um tipo de função parametrizada usada para mapear textos em vetores de ponto flutuante de baixa dimensionalidade (ou, do termo em inglês, *low-dimensional floating-point vectors*), que resulta em representações mais poderosas, por serem significativamente mais compactas e preservarem o relacionamento léxico (também chamado de relacionamento geométrico) e/ou relacionamento semântico entre os vetores das palavras. Desta maneira, textos semelhantes também possuem representações semelhantes no espaço vetorial de *embeddings*, sendo possível, por exemplo, mensurar a similaridade entre textos distintos através da similaridade entre seus vetores.

2.4.1. *Bag of Words* (BoW)

A noção de *Bag-of-Words* (BoW) (em tradução livre, "saco de palavras") foi introduzida por Zellig Harris [Harris 1954], e trata-se da forma mais simples de representação de palavras para um algoritmo de aprendizado de máquina. No BoW, cada documento é representado por um vetor de tamanho *N*, onde *N* é a quantidade de *tokens* distintos no vocabulário. Para manter a consistência entre diversos documentos, é necessário obter o vocabulário completo, ocorrido em todos eles. Desta forma, cada *token* único é representado por uma posição no vetor, a ser preenchida com a quantidade de vezes que o *token* ocorre no documento. Caso não haja nenhuma ocorrência de um *token* no documento, sua respectiva posição recebe o valor 0.

A vantagem do BoW é a sua facilidade de implementação, porém, há uma série de desvantagens: trata-se de um modelo que pode apresentar um grande consumo de memória para vocabulários muito extensos, gerando representações esparsas e apresentando enviesamento em relação a termos muito frequentes. Como alternativas para solução do primeiro problema, podem ser aplicados algoritmos de redução de dimensionalidade como o PCA (acrônimo de *Principal Component Analysis*) [Tan et al. 2018] e, para o problema de desbalanceamento de algumas palavras no vocabulário, temos o TF-IDF, que será apresentado a seguir.

2.4.2. *TF-IDF*: Term Frequency-Inverse Document Frequency

A intuição por trás do modelo conhecido como *TF-IDF*, proposto em [Sparck Jones 1972], está em considerar que, para um conjunto de documentos textuais sendo analisados por um método computacional, palavras que ocorrem em muitos destes documentos provavelmente não serão relevantes para distinguir o conteúdo de cada um deles [Robertson 2004]. O *TF-IDF*, portanto, é um modelo que tem como principal característica ponderar a frequência com que um determinado termo ocorra em um conjunto de documentos, em relação à quantidade de vezes em que ocorre

em um documento em específico. Desta maneira, espera-se obter uma representação balanceada e ao mesmo tempo mais precisa em relação à relevância dos termos para cada documento, atribuindo um maior peso a termos que ocorram em poucos documentos [Jurafsky and Martin 2021].

O índice IDF é computado pela Equação 1, onde N corresponde ao número de documentos analisados e df_i corresponde ao número de documentos em que ocorre o termo em questão:

$$idf = log(\frac{N}{df_i}) \tag{1}$$

Multiplicando a frequência absoluta de cada termo (*TF*) por esta ponderação, podemos representar cada elemento de uma matriz termo $\times$ documento pela Equação 2:

$$w_{i,j} = tf_{i,j} idf_i \tag{2}$$

O *TF-IDF* representou um passo importante para o desenvolvimento de técnicas relacionadas à extração de informação, tendo depois se expandido para outras técnicas da área de processamento de linguagem natural, como extração de tópicos e classificação de texto. No entanto, conforme citado anteriormente, o TF-IDF tem como principal desvantagem o fato da representação vetorial resultante possuir o mesmo tamanho do vocabulário do texto, acarretando no mesmo problema de esparsidade encontrada nas codificações do tipo *one-hot encoding* e BoW. Tal problema pode ser minimizado com a aplicação de métodos de redução dimensional como o PCA. Outra limitação para aplicá-lo em tarefas mais complexas de NLP está no fato de que se trata de uma medida puramente estatística, baseada na frequência dos termos, que não considera as proximidades semânticas em que os termos ocorrem, ou seja, não verifica as palavras que ocorrem mais próximas ou mais distantes entre si. Métodos mais recentes de *word embeddings*, comentados nas próximas subseções, buscam resolver essa segunda limitação.

2.4.3. *Word Embeddings*

A seguir, são descritos os principais métodos utilizados para geração de vetores densos (*Embeddings*), gerados a partir de modelos baseados em redes neurais. São discutidos tanto os métodos de *Word Embeddings*, que geram uma representação vetorial para cada palavra, quanto os métodos de *Sentence Embeddings* (Seção 2.4.4), que, por sua vez, consideram porções mais longas de textos, como as frases, sentenças e parágrafos.

2.4.3.1. *Word2Vec*

Com o *Word2Vec*, proposto por [Mikolov et al. 2013] inaugurou-se um novo paradigma de representação semântica vetorial. Este tipo de representação tem como principal característica a atribuição de um vetor *denso*, de tamanho arbitrário, a cada palavra de um *corpus*, gerado a partir do treinamento de redes neurais. Esses vetores são gerados a partir da análise das *janelas semânticas* em que tais palavras venham a ocorrer. Este tipo de representação trouxe uma série de inovações e vantagens: os vetores não precisam

ser treinados apenas no *corpus* em que será feita a análise (é uma prática comum que se tomem vetores pré-treinados sobre um *corpus* com vocabulário mais extenso, como a *Wikipedia*, e apenas sejam otimizados sobre o *corpus avaliado*); além disso, é eliminado o problema da esparsidade de dados, obtendo uma representação vetorial mais rica. Resultados experimentais indicam que relações semânticas complexas podem ser capturadas, como a relação entre os nomes de países e suas respectivas capitais. Além disso, sinônimos obtém representações mais próximas, enquanto antônimos se distanciam de forma equivalente no espaço vetorial [Mikolov et al. 2013, Jurafsky and Martin 2021].

O *Word2Vec*, em si, é o nome dado a dois modelos conhecidos como *Skip-Gram* e *Continuous Bag of Words* (CBOW). No modelo *Word2Vec Skip-gram*, toma-se como entrada uma palavra, e a partir dela, tenta-se prever as palavras que venham a ocorrer em sua vizinhança. No modelo *Word2Vec CBOW*, toma-se como entrada uma janela de palavras, e tenta-se prever qual palavra ocorreria naquele contexto. A Figura 2.1 ilustra a arquitetura de ambos. Como podemos ver, além das camadas de entrada e saída, há apenas uma única camada de projeção, a qual chamamos de camada oculta (*hidden layer*).

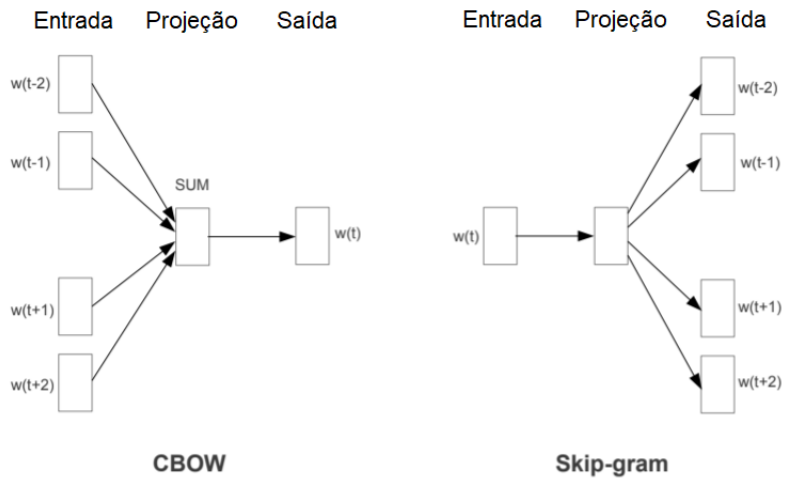

Figura 2.1. Esquema de representação dos métodos que compõem o *Word2Vec*, *CBOW* e *Skip-Gram*. Nela, o *t* indica a posição de uma palavra dentro de uma sentença. Adaptado de [Mikolov et al. 2013].

O algoritmo de aprendizado dos *embeddings* do *Word2Vec Skip-Gram* toma como entrada um *corpus* de texto, com um tamanho N de vocabulário. Inicialmente, são atribuídos valores aleatórios para cada um dos vetores das palavras do vocabulário: tais pesos são ajustados ao longo do treinamento para que palavras ocorrendo em contextos semelhantes obtenham representações vetoriais (*embeddings*) próximos, e palavras de significado distante, que não costumam ocorrer nos mesmos contextos, obtenham representações o mais distante possíveis entre si.

Este treinamento é análogo à tarefa de um classificador binário que vai recebendo instâncias *positivas* e *negativas* de treinamento: em cada palavra analisada por iteração, as instâncias positivas são palavras que realmente ocorrem em sua proximidade, e as negativas, uma seleção, de tamanho proporcional, de palavras que não ocorrem. O treinamento do algoritmo irá, então, minimizar a função de perda (*Loss Function*) representada pela

Equação 3, cujo objetivo é maximizar o produto escalar entre uma palavra e um exemplo *positivo* de contexto, e minimizar em relação aos exemplos *negativos* (as funções σ correspondem à distribuição sigmóide de probabilidade de cada um dos casos). O tamanho do contexto observado é arbitrário, sendo definido como um parâmetro de treinamento. Essa minimização é feita através de uma função de Gradiente Descendente Estocástico. Ao final, duas representações são aprendidas: uma matriz W contendo em cada vetor w_i um *word-embedding* para cada palavra do vocabulário, e uma matriz C em que cada vetor c_i é um *embedding* relativo ao contexto. [Jurafsky and Martin 2021].

$$L_{SG} = -[log\sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} log\sigma(-c_{neg} \cdot w)] \tag{3}$$

Para uma descrição detalhada do treinamento do modelo da arquitetura *CBOW*, consultar [Wen et al. 2016] e [Sivakumar et al. 2020].

2.4.3.2. *Global Vectors* (GloVe)

O modelo conhecido como *GloVe* (do inglês *Global Vectors*) [Pennington et al. 2014] é uma variação de matriz de co-ocorrência, cujo objetivo é produzir vetores contínuos para representação de palavras, de maneira a minimizar o erro quadrático entre o produto vetorial entre uma palavra w e uma palavra de contexto w'. Os vetores são obtidos pela minimização da Equação 4. Nela, $X_{i,j}$ corresponde ao número de vezes em que uma palavra j aparece no contexto de uma palavra i (sendo w'_j e w_i suas representações vetoriais). X_{max} corresponde a um hiperparâmetro de valor arbitrário pré-definido (sugerido experimentalmente pelos autores do modelo como $X_{max} = 100$), e os vetores b_i e b'_j correspondem aos seus respectivos vieses:

$$J = \sum_{i,j=1}^{V} f(X_{i,j})(w_i^T w'_j + b_i + b'_j - log(X_{i,j}))^2 \tag{4}$$

O peso de cada vetor é dado por:

$$f(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^{\alpha} & se \quad X_{i,j} < X_{max} \\ 1 & caso\ contrário. \end{cases}$$

O tamanho total dos vetores de representação de palavras, o fator escalar α (com valor experimentalmente sugerido pelos autores de $\alpha = \frac{3}{4}$) e janela de contexto, por sua vez, são arbitrários [Rendel et al. 2016].

2.4.3.3. *FastText*

Buscando solucionar uma limitação do *Word2Vec* - o fato de que este ignora a estrutura sintática das palavras, considerando cada uma delas como um *token* unitário - o modelo *FastText* propõe uma flexibilização desta representação, que permite mensurar, além da

proximidade semântica das palavras, sua proximidade léxica. Por esta razão, o *FastText* mostra-se robusto em reconhecer palavras que apresentem algum desvio em relação à grafia original (devido a erros de digitação, por exemplo), por permitir a representação de palavras fora do vocabulário original de treinamento.

No *FastText*, cada palavra é representada por uma sequência de *N*-gramas de caracteres. Por exemplo, dada uma palavra como "<farol>", onde os caracteres especiais "<" e ">" indicam o início e o final da palavra, respectivamente, e uma janela (de tamanho arbitrário, escolhido pelo usuário) $n = 3$, o *FastText* toma como entrada os 3-gramas: "<fa", "far", "aro", "rol", "ol>"; além da palavra completa "<farol>". A cada um destes *N*-gramas, é associada uma representação vetorial única. Então, a representação vetorial final da palavra, consistirá na soma dos vetores de representação de todos os seus *N*-gramas [Joulin et al. 2016, Bojanowski et al. 2017].

2.4.4. *Sentence Embeddings*

Similar a *Word Embeddings*, mas considerando porções mais longas de textos, i.e., sentenças, frases ou parágrafos, ao invés de apenas uma palavra, os métodos de *Sentence Embeddings* se destacam em relação a *Word Embeddings* por resultarem em modelos mais eficientes, ao transferir o aprendizado para outras tarefas de NLP, requerendo conjuntos de dados de treinamento menores, e obtendo desempenho superior em diversas tarefas de NLP [Cer et al. 2018]. A seguir, são descritos os principais métodos de *sentence embeddings*, desde os primeiros propostos, tais como *SkipThought*, *InferSent* e *USE*, até os modelos estado-da-arte *SBERT*, e modelos multilíngues (*cross-lingual*) como o *LASER*, o mUSE e o *LaBSE*.

2.4.4.1. SkipThought

O modelo *SkipThought* [Kiros et al. 2015], possui uma arquitetura composta por três redes neurais treinadas por meio de aprendizado não supervisionado, que geram representações vetoriais de tamanho fixo para as sentenças, sendo uma delas para codificar uma sentença de entrada e as outras duas para decodificar a sentença anterior e posterior à entrada. A premissa do modelo é de que, dada uma sentença de entrada, ele consiga identificar regularidades na sentença que permitam inferir as sentenças que vêm antes e depois dela [Kiros et al. 2015]. Por isso, durante o treinamento, cada ponto de dado é composto por três sentenças contíguas, representadas pela tripla (S_{i-1}, S_i, S_{i+1}). A camada de codificação recebe a sentença S_i, enquanto as outras duas camadas de decodificação recebem a sentença anterior (S_{i-1}) e posterior (S_{i+1}). Durante o treinamento, o erro propagado pelas camadas de decodificação é utilizado para refinar o treinamento da camada de codificação. Assim que o modelo converge, somente a camada de codificação é mantida, a qual é usada para gerar os *embeddings* de tamanho fixo, que podem ser aplicados em diferentes tarefas de NLP.

Uma das vantagens do *SkipThought* é o fato de ser um modelo não supervisionado, necessitando apenas do *corpus* de treinamento para criar o modelo de *embeddings*. Como o próprio artigo base do *SkipThought* apresenta na seção de conclusão, essa arquitetura abriu várias possibilidades para novos modelos, por exemplo, baseados em redes

convolucionais e até a expansão para trabalhar com parágrafos inteiros em vez de sentenças [Kiros et al. 2015]. Uma das limitações do *SkipThought* é a necessidade de contexto, uma vez que cada sentença depende da sentença anterior e posterior a ela em uma ordem coerente para treinar o modelo. Nesse sentido, mensagens no Twitter, por exemplo, não seriam adequadas para o treinamento desse modelo.

2.4.4.2. InferSent

O *InferSent* [Conneau et al. 2017], diferentemente do *SkipThought*, é um modelo baseado em aprendizado de máquina supervisionado, capaz de gerar representações semânticas de sentenças escritas em língua inglesa. O InferSent é treinado com o *corpus Stanford Natural Language Inference* (SNLI) [Bowman et al. 2015], que é composto por triplas contendo: (i) uma premissa (uma sentença qualquer); (ii) uma hipótese inferida a partir da premissa; e (iii) o julgamento de voluntários sobre a relação entre a premissa e a hipótese. O julgamento atribuído pelos voluntários pode assumir uma de três possíveis classes, sendo "vínculo", se a premissa possui relação com a hipótese, "contradição", se a hipótese contradiz a premissa, ou "neutro", se não é possível estabelecer uma relação entre a premissa e a hipótese.

Esse *corpus* faz parte de um campo dentro da grande área de NLP chamado *Natural Language Inference* (NLI), ou inferência de linguagem natural. A NLI compreende a tarefa de determinar se uma hipótese e respectiva premissa possuem vinculação lógica, são contraditórias, ou neutras (indeterminadas) entre si. Por exemplo, considerando a premissa "todos os dias nessa localidade são ensolarados" e a hipótese "essa localidade está nublada", a tarefa de NLI é identificar se a hipótese possui vínculo, contradição ou é neutra em relação à premissa. Conjuntos de dados usados para treinar modelos de NLI são comumente usados para treinar modelos de *embeddings*, dada a sua capacidade de generalização, como é o caso do próprio *InferSent* e do *SentenceBERT*, que será apresentado em um dos tópicos desta seção.

A arquitetura do *InferSent* possui dois codificadores idênticos para as sentenças de entrada, um para a premissa e outro para a hipótese, que geram os vetores u e v, respectivamente [Conneau et al. 2017]. Em seguida, esses vetores são concatenados para extrair a relação entre si de três formas distintas: (i) concatenando as duas representações ((u, v)); (ii) multiplicando os vetores ($u * v$); e (iii) calculando o valor absoluto da diferença entre os vetores ($|u - v|$) [Conneau et al. 2017]. A representação resultante, que captura a relação entre a premissa e a hipótese, alimenta um classificador com múltiplas camadas totalmente conectadas, além de uma camada de saída com função de ativação *Softmax* [Conneau et al. 2017]. O InferSent possui 3 saídas para representar cada uma das classes: "vínculo", "contradição" e "neutro" [Conneau et al. 2017]. Diferentes arquiteturas de redes neurais, incluindo células do tipo *Long Short-Term Memory* (LSTM) ou *Gated Recurrent Units* (GRU), e suas variações, foram testadas para os codificadores nessa arquitetura [Conneau et al. 2017]. O modelo com maior acurácia nos testes foi o que utilizou redes neurais com células LSTM bidirecionais e camada de *max pooling* (referida no artigo como BiLSTM-max) [Conneau et al. 2017]. Em testes realizados pelos autores, o modelo BiLSTM-max obteve desempenho ligeiramente superior em várias tarefas de NLP, em comparação ao *SkipThought* original, sendo um modelo capaz de generalizar

melhor em várias tarefas de *Semantic Textual Similarity* (STS), que é um campo dentro de NLP relacionado a tarefas como tradução, sumarização e geração de textos, perguntas e respostas (QA), busca semântica e sistemas conversacionais [Cer et al. 2017].

Uma vantagem da arquitetura *InferSent*, é o fato dela conseguir trabalhar com sentenças e textos com maior dimensão, além de não necessitar que o *corpus* utilizado no treinamento esteja em uma sequência lógica, como é o caso do *SkipThought*. Apesar de a arquitetura poder ser adaptada para treinamento com diferentes conjuntos de dados, a versão original treinada com os dados do SNLI está disponível apenas para língua inglesa, limitando sua aplicação a esse idioma.

2.4.4.3. *Universal Sentence Encoder* (USE)

O *Universal Sentence Encoder* (USE) [Cer et al. 2018] oferece duas maneiras para gerar *sentence embeddings* de textos escritos em inglês: (i) utilizando o subgrafo de codificação (*encoding*) da arquitetura *Transformers* [Vaswani et al. 2017]; e (ii) utilizando uma *Deep Averaging Network* (DAN) [Iyyer et al. 2015]. Em comum, ambos os métodos recebem como entrada um a sequência de palavras, em minúsculo e *tokenizada* com *Penn Treebank* (PTB) *tokenizer*, e produzem um *sentence embedding* com 512 dimensões. Apesar dessa semelhança, a forma como cada método constrói os *embeddings* das sentenças é muito distinta, como veremos a seguir.

No primeiro caso, é utilizado o mecanismo de autoatenção (*self-attention*), introduzido pela arquitetura *Transformers* [Vaswani et al. 2017]. Na arquitetura *Transformers*, o mecanismo de *self-attention* é o responsável por calcular o relacionamento entre as palavras da sentença de entrada e, enquanto ele faz isso para uma determinada palavra, ele permite que o modelo tenha foco nas outras palavras da senteça que são mais similares a esta, por isso o nome do mecanismo é atenção, por permitir que o modelo concentre-se mais no que realmente importa [Vaswani et al. 2017]. Desta forma, ao utilizar o mecanismo de *self-attention*, o modelo USE consegue computar as representações das palavras com contexto de uma dada sentença de entrada, considerando tanto a ordem como a identidade das palavras. Tais representações são convertidas em um único vetor numérico de tamanho fixo, obtido através da soma dos elementos de mesmas posições dos vetores. A principal vantagem em se utilizar esse método é o alto desempenho em diferentes tarefas de NLP, mesmo quando os conjuntos de dados possuem poucas amostras para treinamento.

Por exemplo, em [Cer et al. 2018], os autores conduziram diversos experimentos para avaliar o desempenho dos modelos em diferentes tarefas de NLP, tais como a SST (classificação binária de sentimento em nível de frase [Socher et al. 2013]) e STS *Benchmark* (similaridade textual semântica entre pares de frases [Cer et al. 2017]). Para isso, foi utilizado apenas *word embeddings* (sendo utilizado o *Word2Vec*), ou apenas *sentence embeddings*, ou a combinação de *word* e *sentence embeddings*, em relação ao *baseline*, que inicializa os pesos da rede de maneira aleatória e os *word embeddings* são aprendidos ao longo do treinamento com base nos dados da tarefa avaliada. Dentre os resultados apresentados, chama a atenção o desempenho do USE com o mecanismo de *self-attention* (chamado de USE_T) para a tarefa SST, onde foram avaliados o desempenho dos mode-

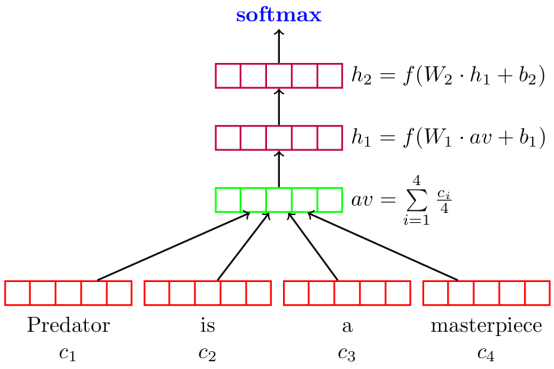

Figura 2.2. Ilustração da arquitetura do modelo DAN. Imagem de [Iyyer et al. 2015].

los considerando diferentes tamanhos do conjunto de treinamento (1.000, 2.000, 4.000, 8.000, 16.000, 32.000 e 67.300). Como mostrado, com apenas 1.000 amostras de treinamento, o modelo USE_T obteve um resultado superior em comparação com vários outros modelos, mesmo eles tendo sido treinados com o conjunto de dados de treinamento completo (i.e., 67.300 amostras). Para as demais tarefas de NLP avaliadas, os melhores resultados também foram obtidos com o USE_T, seja ele sozinho ou em conjunto com algum modelo de *word embeddings*.

Diante desses resultados, fica claro o grande potencial do USE_T para transferir seu aprendizado para muitas tarefas de NLP. No entanto, como o USE_T utiliza o *transform encoding model*, sua complexidade de tempo é $O(n^2)$, sendo N o tamanho da sentença de entrada, bem como sua complexidade de espaço também é quadrática, ou seja, $O(n^2)$. Desta forma, caso seja necessário ter uma melhor eficiência em termos de tempo, memória, ou ambos, sem degradar significativamente o desempenho, você pode optar pelo modelo USE com o *encoding* gerado com a DAN, chamado USE_D, já que ele tem complexidade de tempo linear, $O(n)$, e sua complexidade de espaço é constante no tamanho da entrada, $O(1)$.

Para combinar os múltiplos *word embeddings* das palavras de uma dada sentença de entrada em um único vetor (i.e., o *sentence embedding*), a DAN primeiro computa a média dos *word embeddings* e, então, utiliza o vetor resultante para alimentar uma ou mais camadas de uma rede neural profunda (DNN, do Inglês *Deep Neural Network*) [Iyyer et al. 2015]. A Figura 2.2, apresenta um exemplo do processo da DAN para uma frase contendo 4 palavras, utilizando duas camadas ocultas (i.e., *hidden layers*) e a classificação (linear) na última camada (i.e., *softmax*). Note que ao calcular a média dos *word embeddings* para utilizar como entrada da DNN, a informação sintática, i.e., a posição em que as palavras aparecem na frase, se perde e, por isso, a DAN é considerada uma função de composição não ordenada. Apesar disso, como é possível observar nos resultados apresentados em [Cer et al. 2017], o USE_D apresenta bom desempenho em tarefas de classificação de texto.

2.4.4.4. *SentenceBERT* (SBERT)

O *SentenceBERT* (SBERT) [Reimers and Gurevych 2019], é um modelo estado-da-arte baseado no *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin et al. 2018] – um modelo considerado "divisor de águas" na área de NLP. O SBERT permite a geração de *embeddings* com mesma capacidade de generalização, mas com complexidade computacional ordens de magnitude menor em comparação com seu predecessor e outros modelos até então estado-da-arte, como o *InferSent* e USE [Reimers and Gurevych 2019]. Por isso, para entender essas vantagens, primeiramente é necessário compreender a arquitetura de seu modelo predecessor [Reimers and Gurevych 2019].

O BERT é um modelo de aprendizado profundo criado para ser facilmente adaptado a diferentes tarefas de similaridade semântica sem necessitar grandes alterações em sua estrutura (agnóstico de tarefa) [Devlin et al. 2018]. Esse modelo gera representações usando uma lógica similar à aplicada pelo *SkipThought*, onde o contexto anterior e posterior de uma palavra ou sentença é usado para identificar as características que melhor representam sua relação com seu entorno. Para isso, o BERT usa o mecanismo de atenção introduzido pela arquitetura *Transformers* [Vaswani et al. 2017]. Sua arquitetura é composta por codificadores com redes neurais de aprendizado profundo, treinados percorrendo o *corpus* tanto na ordem natural em que as sentenças ocorrem, quanto na ordem inversa, por isso o uso do termo "bidirecional" [Zhang et al. 2021]. O fato do aprendizado ocorrer em ambas as direções, permite que esse modelo possa ser pré-treinado mais rapidamente e com uma quantidade menor de dados em comparação com modelos estado-da-arte que usam aprendizado unidirecional, como o *Generative Pre-trained Transformer* (GPT) [Radford et al. 2018, Zhang et al. 2021].

A arquitetura do SBERT, por sua vez, se assemelha a do *InferSent*, contendo dois codificadores BERT com uma estrutura de redes siamesas - compartilhando os pesos entre si e necessitando apenas uma entrada - esse é o diferencial que torna esse modelo muito mais rápido em comparação ao BERT [Reimers and Gurevych 2019]. Na saída de cada um dos codificadores BERT, uma camada de *pooling* é responsável por gerar duas representações vetoriais, u e v, de tamanhos fixos [Reimers and Gurevych 2019]. Para tarefas de classificação, os vetores u e v, então, são transformados em uma tripla contendo (i) u; (ii) v; e (iii) o valor absoluto da subtração entre o vetor u e v ($|u - v|$) [Reimers and Gurevych 2019]. Essa tripla é passada para um modelo de classificação usando a função de ativação *Softmax* [Reimers and Gurevych 2019]. Já para tarefas de regressão, a similaridade do cosseno entre os vetores u e v é computada após a camada de *pooling* e enviada para a saída do modelo [Reimers and Gurevych 2019].

Com essa arquitetura, o SBERT, além de ter apresentado melhor desempenho em relação ao *InferSent* e USE, também resolve os problemas de complexidade computacional apresentados pelo BERT [Reimers and Gurevych 2019]. Além disso, o SBERT possui outras características que o tornam bastante atrativo, não apenas na área acadêmica, mas em aplicações na indústria. Por exemplo, modelos SBERT monolíngues podem ser aumentados para tarefas de NLP multilíngues com boa capacidade de generalização, partindo da premissa de que uma sentença traduzida deve ser mapeada para a mesma posição no espaço vetorial que a sentença original [Reimers and Gurevych 2020]. Esse é o pro-

cesso usado por um famoso modelo de análise de toxicidade em comentários na *web* chamado Perspective API [Jigsaw 2022], que será apresentado em uma das seções desse capítulo.

Outra vantagem do SBERT é o fato deste modelo estar disponível por meio de uma biblioteca de código aberto feita em Python, chamada *Sentence-Transformes*[12], que disponibiliza vários modelos pré-treinados para diferentes tarefas, incluindo modelos multilíngues. O projeto tem uma documentação bem estruturada e expõe alguns métodos fáceis de usar para a geração de *embeddings*, bastando apenas indicar o nome do modelo pré-treinado que será usado e aplicá-lo para gerar os *embeddings*. A transferência de aprendizado também pode ser facilmente feita com essa biblioteca, permitindo adaptar qualquer modelo disponibilizado a diferentes tarefas de NLP.

2.4.4.5. Modelos Multilíngues

Modelos multilíngues, também conhecidos como independente de idioma (do termo em Inglês *language-agnostic*), têm a capacidade de gerar *embeddings* similares para palavras com o mesmo significado mas escritas em idiomas diferentes. Ou seja, no espaço vetorial de *embeddings*, as palavras bom, *good*, *bueno*, *bien*, *gut* e *bene*, terão representações vetoriais muito semelhates, caso o modelo multilíngue suporte seus respectivos idiomas, Português, Inglês, Espanhol, Francês, Alemão e Italiano.

Note que essa caraterística é muito interessante, uma vez que pode ser muito difícil ter *corpus* disponíveis em diferentes idiomas para treinar os modelos de linguagem, sendo a maioria deles escritos em língua Inglesa. Além disso, ao lidar com textos de mídias sociais, é muito comum se deparar com textos escritos em outros idiomas. Por exemplo, ao coletar *tweets* compartilhados por pessoas que estão em uma cidade cosmopolita como Nova Iorque, EUA, além de textos escritos em Inglês, como é esperado, também é comum obter *tweets* escritos em Português, Espanhol, Mandarim, entre outros idiomas. Desta forma, utilizar modelos multilíngues para geração dos *embeddings* podem ser uma boa opção.

Como vimos na seção anterior, o SBERT pode ser treinado para tarefas de NLP multilíngues. Além dele, são apresentados a seguir outros três modelos multílingues amplamente utilizados pela academia e indústria.

- *Language-Agnostic SEntence Representations* (LASER) [Artetxe and Schwenk 2019], foi o primeiro modelo a explorar a representação de sentenças multilíngues para propósito geral, ou seja, sem ter uma tarefa de NLP específica. Para isso, o LASER utiliza uma arquitetura *sequence-to-sequence encoder-decoder*, como mostra a Figura 2.3. Como podemos ver, o *encoder* é composto por uma rede LSTM bidirecional (ou simplesmente, BiLSTM), que recebe como entrada uma sequência de palavras (representadas pelas variáveis $x_1, x_2, ...$), seguida do símbolo que indica o final da sentença (i.e., $</s>$). Cada palavra é primeiro codificada por um vocabulário de *Byte-Pair encoding* (BPE), que foi construído considerando a concatenação de todos os corpora de treinamento

[12]https://www.sbert.net. Último acesso em 11 de Setembro de 2022.

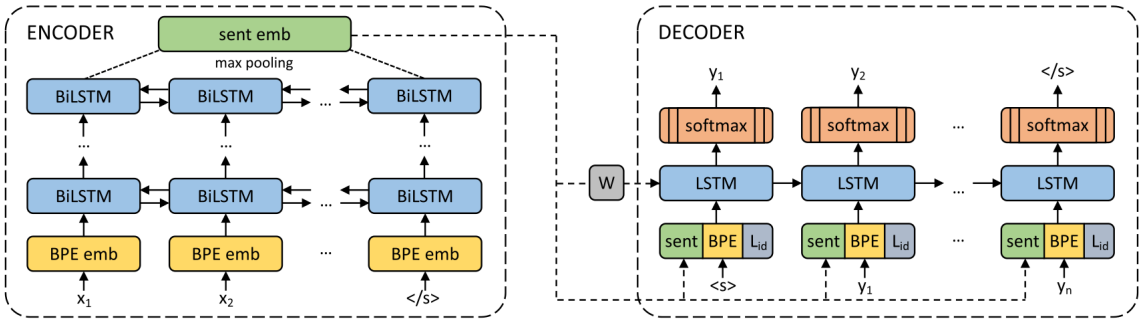

Figura 2.3. Ilustração da arquitetura do modelo LASER. Imagem de [Iyyer et al. 2015].

dos 93 idiomas suportados pelo modelo, antes de alimentar a rede BiLSTM. Com isso, não é necessário informar explicitamente o idioma de entrada para o *encoder*, o que auxília o modelo a aprender as representações independente do idioma. Por fim, o *sentence embedding* é obtido no *encoder* ao aplicar a operação *max pooling* sobre a BiLSTM. O *decoder* por sua vez, que é composto por apenas uma rede LSTM, recebe como entrada o *sentence embedding* gerado pelo *encoder*, concatenado em cada etapa do tempo com a codificação BPE da sentença alvo (i.e., as palavras $<s>, y_1, ..., y_n$) e a codificação que específica qual idioma deve ser gerado (representado pela variável L_{id}).

Assim, considerando dados anotados em apenas dois idiomas alvos, onde foram utilizados Inglês e Espanhol, o LASER foi treinado de maneira *end-to-end* para tarefa de tradução. Após o treinamento, o *encoder* torna-se capaz de gerar *sentence embeddings* em qualquer um dos 93 idiomas do conjunto de treinamento, de modo que sentenças semelhantes em diferentes idiomas também estejam próximas no espaço de *embeddings*.

- *Multilingual Universal Sentence Encoder* (mUSE) [Yang et al. 2019] é uma extensão do modelo USE [Cer et al. 2018] (descrito na Seção 2.4.4.3), onde são adicionados dois modelos multilíngues multitarefas, sendo um baseado em Rede Neural Convolucional (*Convolutional Neural Network*, CNN) e outro na arquitetura *Transformers*, além de um modelo *Transformers* multilíngue para uso em recuperação de perguntas e respostas (*Retrieval Question Answering*, ReQA). Em comum, os três novos modelos são treinados utilizando a abordagem *multi-task dual-encoder* [Chidambaram et al. 2018], que é capaz de aprender representações semelhantes para frases com significados idênticos, pórem, escritas em idiomas distintos, por meio de *bridging translation task* [Chidambaram et al. 2018]. Ao todo, 16 idiomas distintos são suportados pelo mUSE.

- *Language-agnostic BERT Sentence Embedding* (LaBSE) [Feng et al. 2020], também utiliza a abordagem *dual-encoders*, semelhante ao mUSE, para aprender representações multilíngues. Nesta abordagem, pares de sentenças de origem e alvo são codificadas separadamente, onde os *embeddings* de cada uma são obtidos por *encoders* distintos, mas que compartilham seus parâmetros. Os *embeddings* são então treinados considerando a tarefa de tradução. Como podemos ver na Figura 2.4,

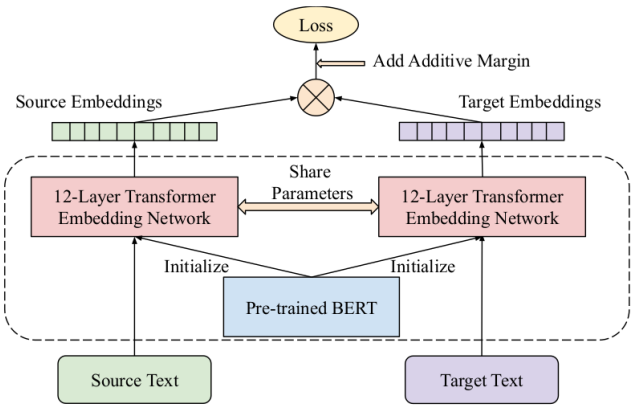

Figura 2.4. Ilustração da arquitetura do modelo LaBSE. Imagem de [Feng et al. 2020].

o principal diferencial do LaBSE, em relação aos demais modelos multilíngues que utilizam essa abordagem, é a combinação de um *encoder* pré-treinado baseado no modelo de linguagem BERT com os *dual-encoders*, evitando assim a necessidade de treiná-los do zero. Com isso, o LaBSE consegue reduzir drasticamente a quantidade de treinamento necessário e, ainda assim, alcançar um desempenho superior. Ao todo, 109 idiomas distintos são suportados pelo LaBSE. Além disso, o LaBSE também foi capaz de produzir bons resultados para mais de 30 idiomas além dos que já são suportados, mesmo não havendo quaisquer dados de treinamento de tais idiomas.

2.5. Modelagem e Extração de Conhecimento

Nesta seção, serão apresentadas técnicas de modelagem que podem ser aplicadas para extrair conhecimento a partir de representações vetoriais de textos. Dentre os tópicos que serão apresentados, estão as técnicas para agrupamento de textos similares, incluindo o agrupamento com *k-means*, agrupamento hierárquico e agrupamento usando algoritmos de detecção de comunidades em grafos. Por último, serão apresentadas técnicas para modelagem de tópicos e seus respectivos desafios.

2.5.1. Agrupamento de Textos

O problema da clusterização ou agrupamento, pode ser definido como o de identificar grupos de objetos similares em um conjunto de dados [Aggarwal and Zhai 2012]. Em NLP, o agrupamento de textos pode ser aplicado em diferentes granularidades, por exemplo, a nível de documento, parágrafos, sentenças ou termos [Aggarwal and Zhai 2012]. Além do simples agrupamento de textos, também é possível realizar a organização hierárquica, sumarização e classificação de documentos – no último caso, transformando a tarefa de clusterização em uma tarefa de aprendizado supervisionado [Aggarwal and Zhai 2012]. Várias técnicas podem ser utilizadas para realizar clusterização, incluindo abordagens clássicas, como o *k-means* e o agrupamento hierárquico, assim como a transformação de textos em grafos para aplicação de algoritmos de detecção de comunidades.

2.5.1.1. *k*-means

Um dos mais famosos métodos de agrupamento é o *k-means*, dado seu funcionamento simples e intuitivo. Inicialmente, deve-se escolher o número de agrupamentos (ou, *clusters*), em que o conjunto de documentos deve ser particionado, dado pelo parâmetro k, que pode ser definido com o auxílio de heurísticas [Tan et al. 2018]. Então, são definidos os centróides iniciais para cada um dos k *clusters* em um espaço multidimensional, podendo inicialmente serem aleatórios ou definidos por alguma heurística [MacQueen 1967]. Depois, é calculada a similaridade da representação vetorial de cada documento com todos os centróides dos *clusters* usando, geralmente, a distância Euclidiana ou alguma outra métrica de similaridade [MacQueen 1967]. Em seguida, cada documento é alocado ao *cluster* com maior similaridade com seu centróide [MacQueen 1967]. Feito isso, o centróide de cada *cluster* é atualizado pela média da representação vetorial dos documentos no *cluster* (por isso o nome *k-means*) e repetido o processo de cálculo de similidade até que o algoritmo atinja um critério de parada (i.e., um número máximo pré-definido de iterações ou quando não houver alterações significativas nos centróides) [MacQueen 1967].

Esse método pode ser aplicado sobre representações vetoriais ingênuas de documentos, como o *one-hot-encoding*, apresentando resultados satisfatórios para algumas aplicações. Contudo, representações simples como essa, em geral, resultam em vetores esparsos que dificultam o cálculo da similaridade entre documentos. Esse é o caso, por exemplo, de textos curtos em um *corpus* com vocabulário extenso, como no caso de mensagens do Twitter, ou comentários em postagens de mídias sociais em geral, onde a complexidade no vocabulário é maior devido à presença de gírias, erros gramaticais e linguagem da Internet. Por isso, representações vetoriais mais sofisticadas, como *word/sentence embeddings*, que são capazes de capturar melhor as regularidades léxicas e/ou semânticas presentes nos textos, são preferíveis nesse tipo de aplicação [Aggarwal and Zhai 2012].

Em análise de textos de mídias sociais, a velocidade com que um algoritmo consegue produzir um resultado pode ser determinante para sua escolha, uma vez que nesses ambientes digitais são produzidos grandes volumes de dados para serem processados, sendo comum algumas aplicações demandarem que isso seja feito, inclusive, em tempo real. Nesse sentido, o algoritmo *k-means* original tem um custo computacional alto de $O(kns)$, onde n é o número de documentos e s é o tamanho da representação vetorial de cada documento [Sculley 2010]. Uma forma de reduzir o custo computacional é removendo termos pouco representativos no *corpus*, consequentemente reduzindo o valor s, por exemplo, usando o próprio valor de TF-IDF para filtrar termos muito comuns ou muito raros no vocabulário. Apesar disso, ainda assim, essa redução pode não ser suficiente para grandes conjuntos de dados. Por isso, foram propostas diversas variações do algoritmo *k-means*, uma delas, dentre as mais famosas, chamada *mini-batch k-means* [Sculley 2010]. Essa abordagem trabalha com amostragens aleatórias de documentos para serem comparados com os centróides, dado por um parâmetro, que torna o algoritmo ordens de magnitude mais rápido que a versão original, possibilitando clusterização de documentos praticamente em tempo real em diversas aplicações [Sculley 2010].

2.5.1.2. Agrupamento Hierárquico

Um dos desafios em grandes *corpus* de domínios do mundo real, como no caso de textos de mídias sociais, é não saber de antemão informações quantitativas sobre como documentos estão organizados [Ushioda 1996]. O agrupamento com *k-means*, como apresentado na seção anterior, colabora nessa tarefa. Contudo, em algumas aplicações, pode ser difícil escolher o valor de k e compreender o motivo de um determinado agrupamento ter sido identificado, principalmente em grandes *corpus*. Para minimizar esse problema, existe uma classe especial de métodos de agrupamento, chamada agrupamento hierárquico. O agrupamento hierárquico permite agrupar as representações dos textos em diferentes níveis de granularidade, criando organizações dos *clusters* e *subclusters* na forma de dendrogramas. Essa é uma representação conveniente, que ajuda a responder perguntas como "Quantos grupos úteis existem nestes dados?" e "Quais são as inter-relações mais importantes?" [Murtagh and Contreras 2012].

Existem dois métodos para realizar o agrupamento hierárquico: (i) por meio de algoritmos divisivos, e (ii) por meio de algoritmos aglomerativos. Algoritmos hierárquicos divisivos [Zhao and Karypis 2002], iniciam o processo de particionamento dividindo o conjunto de documentos inicial em dois *clusters*. Em seguida, os *clusters* formados (contendo mais de um documento), são divididos ao meio, baseando-se em uma heurística dada por uma função de critério de clusterização. Esse processo continua $n-1$ vezes, sendo n o número de documentos. Ao final, cada *cluster* irá conter apenas um documento, chamado de nós folhas. Nessa abordagem, o dendrograma é construído de cima para baixo, partindo de um único *cluster* com todos os documentos e terminando com cada documento em seu próprio *cluster* [Zhao and Karypis 2002]. De maneira inversa, os algoritmos hierárquicos aglomerativos [Zhao and Karypis 2002], começam com um documento em cada *cluster* e, progressivamente, une os pares de *clusters* entre si, até resultar em um único *cluster* com todos os documentos inclusos.

Os algoritmos de agrupamento divisivo, em geral, definem o problema de agrupamento como o cálculo de uma solução de agrupamento tal que o valor de uma função de critério de clusterização seja otimizado [Zhao and Karypis 2002]. As funções de critério de clusterização podem ser dividas em 4 grupos: (a) internas, (b) externas, (c) híbridas e (d) baseadas em grafos. As funções internas (a), consideram na similaridade entre os documentos do *cluster* a ser particionado, desconsiderando documentos fora dele; as funções externas (b), se baseiam em quanto o *cluster* que será particionado se difere de outros *clusters*; as funções baseadas em grafos (c), modelam os documentos como grafos e usam métricas de clusterização; e as funções híbridas (d) otimizam múltiplas funções de critério de clusterização [Zhao and Karypis 2002].

Os algoritmos de agrupamento aglomerativo, por sua vez, podem aplicar diferentes métodos para identificação dos pares de *clusters*, a serem agrupados em cada passo [Tan et al. 2018]. Existem vários métodos feitos para aplicações específicas, dentre eles, o de *linkagem* única e o de *linkagem* completa. O esquema de *linkagem* única, agrupa dois *clusters* que contêm o par de documentos com a menor distância entre si, dentre todos os pares de documentos no *corpus* [Omran et al. 2007]. Já o esquema de *linkagem* completa, agrupa dois *clusters* quando a distância entre o par de documentos mais distantes entre si for a menor dentre todos os pares de documentos no *corpus* [Omran et al. 2007]. Há

também um método muito similar ao aplicado pelo *k-means*, chamado método centróide, onde o agrupamento é feito entre dois *clusters* que tenham a menor distância entre seus pontos médios (centróides) [Tan et al. 2018].

Em relação à complexidade computacional, os algoritmos de agrupamento hierárquico aglomerativo possuem dois passos que aumentam seu custo computacional, sendo o primeiro deles a comparação da similaridade entre os pares de todos os documentos, repetindo esse procedimento em cada passo ao subir na árvore de resultados. Por isso, esses algoritmos possuem complexidade polinomial, geralmente de $O(n^2)$ [Zhao and Karypis 2002]. Já os algoritmos de agrupamento divisivos, dada a natureza de "dividir para conquistar", têm complexidade de $O(nlog(n))$ ou, em muitos casos, linear – quantos mais balanceados os *clusters*, mais rapidamente o algoritmo converge [Zhao and Karypis 2002].

2.5.1.3. Detecção de Comunidades em Grafos

Grafos no contexto de NLP podem ser interessantes pelo poder de expressarem relações complexas entre as entidades. Comunidades, também chamadas *clusters* ou agrupamentos em grafos, são estruturas formadas por vértices que apresentam características em comum. No contexto de NLP, a detecção de comunidades pode ser utilizada para identificar grupos de textos similares. Para isso, o primeiro passo consiste em modelar um grafo que representa um dado *corpus*. Isso pode ser feito de diferentes formas, por exemplo, construir um grafo ponderado e não direcionado, onde os vértices representam os *tokens* presentes no *corpus* (i.e., o vocabulário), as arestas indicam alguma relação entre os *tokens* e, o peso das arestas, reflentem a intensidade dessa relação.

Para fins ilustrativos dessas ideias, considere o grafo construído em [Santos et al. 2020a]. Nesse trabalho, os autores primeiro realizaram *Part-Of-Speech* (POS) *tagging* em todas as sentenças de um *corpus*, o qual é composto por aproximadamente 40 mil comentários de usuários das mídias sociais Google Places e Foursquare (Tips). Com isso, foi possível identificar quais *tokens* eram um adjetivo, por ser a classe de palavras que designa qualidade e, por isso, foi a escolhida pelos autores. Ao todo, foram identicados 271 *tokens* como sendo adjetivos. O próximo passo, foi identificar a similaridade sintática e semântica entre esses *tokens*, para ponderar as arestas de um grafo. Para isso, o mesmo *corpus* foi utilizado para treinar o modelo Word2Vec [Mikolov et al. 2013] e, assim, ser possível computar a similaridade entre os 271 *tokens*. Adicionalmente, também foi considerado a polaridade do sentimento (podendo ser positiva, negativa ou neutra, como é discutido em mais detalhes na Seção 2.6.3), para compor a pontuação final de similaridade entre os *tokens*. A utilização da polaridade do sentimento nesse caso é bastante relevante, para aumentar a pontuação entre *tokens* de mesma polaridade e, por outro lado, penalizar quando os *tokens* têm polaridades opostas. Isto é devido a palavras como "ótimo" e "péssimo", ou "alegre" e "triste", que podem ter contextos muito parecidos, mas sentidos opostos. Por exemplo, "meu dia está sendo ótimo" e "meu dia está sendo péssimo", ou, "recebi uma notícia que alegrou o meu dia" e "recebi uma notícia que entristeceu o meu dia". Em ambos os exemplos, as expressões são quase iguais, exceto pelos adjetivos.

De posse dos *tokens* e das pontuações de similaridade, os autores construíram um grafo não-direcionado, completo e ponderado, denotado por $K_n = (V, E, \omega)$, onde V é o conjunto de vértices que representa os 271 *tokens* (i.e., $|V| = n = 271$), E é o conjunto de arestas entre todos os vértices, onde cada aresta $e \in E$ tem o peso $\omega_e \in \omega$, definido pela pontuação de similaridade. Antes de realizar a detecção de comunidades no grafo K_n, os autores removeram as arestas consideradas "leves", aquelas cuja a pontuação é menor que um certo *threshold*, afim de remover conexões entre vértices pouco similares. Então, após a remoção de tais arestas, os vértices que ficaram isolados também foram removidos, resultando em um grafo com 261 vértices e 3485 arestas, que foi utilizado para detecção de comunidades.

O método escolhido pelos autores para detecção de comunidades foi o *clique percolation* [Palla et al. 2005], que identifica k-clique comunidades como sendo a união de todas as cliques de tamanho k que podem ser alcançadas através de cliques adjacentes (i.e., que compartilham $k-1$ vértices). Como resultado, em [Santos et al. 2020a], foram identificadas 8 comunidades, considerando $k = 6$.

O método *clique percolation* é particularmente interessante, porque possibilita a remoção de possíveis ruídos, uma vez que vértices fracamente conectados não são colocados em nenhuma comunidade. Além disso, também permite a existência de comunidades sobrepostas, ou seja, um mesmo vértice pode pertencer a várias comunidades e, assim, evita a divisão de grandes comunidades em pequenas, mantendo comunidades com contextos semelhantes unidas. No entando, este método possuí duas principais desvantagens: definir o valor adequado de k; e, como o problema k-clique é $\mathscr{NP}$-difícil, no pior caso, sua complexidade é exponencial no número de vértices.

Alternativamente, outro método bastante conhecido para detecção de comunidades em grafos é o *Label Propagation Algorithm* (LPA) [Zhu and Ghahramani 2002] que, diferente do método *clique percolation*, é rápido (complexidade de tempo quase linear no tamanho da entrada – o grafo) e considera apenas a estrutura do grafo para identificar as comuninades, sem necessitar definir a priori nenhuma função objetivo ou parâmetro. Para isso, o LPA considera inicialmente que cada objeto é uma comunidade (ou *cluster*). Então, de maneira interativa, o LPA atualiza a *label* de cada vértice do grafo, de acordo com a *label* majoritária entre os seus vizinhos e, em caso de empate entre duas ou mais *labels*, uma é escolhida de maneira aleatória. Quando o algoritmo converge, i.e., cada nó já possui a *label* majoritária de seus vizinhos, então ele se encerra e os vértices que terminarem esse processo com a mesma *label* são considerados como sendo da mesma comunidade. Também é possível definir um número máximo de iterações, para encerrar o processo antes da convergência. Além disso, é possível utilizar o LPA de maneira semi-supervisionada, onde se atribuí algumas *labels* preliminarmente, i.e., algumas comunidades são definidas no início, restringindo assim as possíveis soluções resultantes.

2.5.2. Modelagem de Tópicos

Um objetivo comum em NLP é identificar estruturas semânticas compartilhadas entre textos para determinar quais eventos, conceitos ou assuntos estão sendo discutidos em um conjunto de documentos [Vayansky and Kumar 2020]. Esse é o papel da modelagem de tópicos, um processo que envolve a aplicação de métodos estatísticos para descoberta de estruturas semânticas latentes, também chamadas de tópicos, em um conjunto de textos

[Huh and Fienberg 2012]. Para isso, parte-se da premissa de que cada documento pode ser representado por uma mistura de tópicos, sendo cada tópico composto por palavras que melhor o define no *corpus* analisado [Huh and Fienberg 2012]. Por exemplo, em um *corpus* de notícias de várias fontes, os tópicos poderiam representar eventos que ocorreram (e.g., confronto entre Rússia e Ucrânia, Copa do Mundo, etc.) ou delinear grandes temas (e.g., economia, política, educação, etc.). A principal tarefa na modelagem de tópicos, portanto, é descobrir padrões de uso de palavras e relacionar documentos que compartilham padrões semelhantes [Alghamdi and Alfalqi 2015].

Quando aplicada a modelagem de tópicos, para a maioria dos casos, não é necessário considerar a ordem em que as palavras ocorrem nos documentos, sendo geralmente usada uma representação vetorial de *Bag of Words* (BoW) para os textos [Huh and Fienberg 2012]. De uma forma simplificada, em um modelo de tópicos cada documento pode ser representado por um histograma com a contagem de cada termo contido nele [Huh and Fienberg 2012]. A forma desse histograma é proveniente de uma distribuição entre k tópicos pré-definidos, os quais são distribuídos entre os termos no vocabulário do *corpus* [Huh and Fienberg 2012]. O objetivo da modelagem de tópicos, então, é aprender essas distribuições para conseguir inferir estruturas semânticas latentes no *corpus* [Huh and Fienberg 2012]. Para isso, as técnicas de modelagem de tópicos geralmente incluem dois componentes principais: uma matriz que relaciona os termos do vocabulário aos k tópicos e outra matriz que relaciona os k tópicos aos documentos [Huh and Fienberg 2012].

Existem várias técnicas conhecidas para modelagem de tópicos, tais como o *Latent Semantic Analysis* (LSA) [Landauer et al. 1998] – não probabilístico –, e sua respectiva versão probabilística (pLSA) [Hofmann 2001], e o *Latent Dirichlet Allocation* (LDA) [Blei et al. 2003]. Elas podem ser facilmente aplicadas por meio das implementações disponíveis na biblioteca GenSim [Řehůřek and Sojka 2010], que além da implementação de todas essas técnicas, também possui uma versão otimizada do LDA com processamento paralelizado e métricas para avaliação de desempenho. Outra biblioteca mais recente, chamada OCTIS [Terragni et al. 2021], permite a criação de modelos de tópicos sem a necessidade de otimizar os hiperparâmetros manualmente, os quais são estimados de forma automática por meio de uma abordagem Bayesiana. Essa biblioteca usa a mesma implementação do GenSim [Řehůřek and Sojka 2010], portanto, para efeitos de comparação, são exatamente as mesmas implementações. Outra vantagem da biblioteca OCTIS, é o fato dela possuir uma interface gráfica, que permite a prototipação rápida de modelos, sem a necessidade de criar código, facilitando com que pesquisadores de diferentes domínios possam criar modelos de tópicos para desafios específicos em suas áreas [Terragni et al. 2021].

Uma característica que deve ser considerada ao modelar tópicos em grandes *corpus* é sua evolução ao longo do tempo. Essa característica, em alguns casos, pode comprometer o processo de descoberta de tópicos, uma vez que os métodos tradicionais, como o LSA, pLSA e LDA, não capturam a relação entre os tópicos mais novos com seus predecessores [Alghamdi and Alfalqi 2015]. Um exemplo prático deste caso, seria o de um pesquisador cujo objetivo é identificar temas dentro de um determinado campo de pesquisa ao longo de décadas, de forma que seja possível obter conjuntos de artigos que tratam do mesmo tema, mesmo que algumas palavras-chave tenham sido alteradas ao longo do

tempo [Alghamdi and Alfalqi 2015]. Algumas técnicas conhecidas para modelagem de tópicos no decorrer do tempo são o *Topic Over Time* (TOT) [Wang and McCallum 2006] e *Dynamic Topic Models* (DTM) [Huh and Fienberg 2012]. Essas técnicas adicionam um novo componente no modelo de tópicos, que é a relação entre a metainformação de *timestamp* dos documentos com os tópicos. Dessa forma, os modelos temporais têm o objetivo adicional de aprender a distribuição dos tópicos ao longo do tempo em comparação com os modelos atemporais [Wang and McCallum 2006, Huh and Fienberg 2012].

Alguns desafios em modelagem de tópicos devem ser destacados. A caracterização de tópicos é uma tarefa complexa, uma vez que é necessário um avaliador humano para identificar qual é o assunto a partir das *top* N palavras mais representativas de um tópico [Ramage et al. 2009]. Nesse sentido, dar um nome significativo para um tópico é difícil, e depende de um profundo entendimento do domínio e vocabulário, além das nuances representadas pelas *top* N palavras mais representativas do tópico, que nem sempre delimitam de forma clara um tema específico [Ramage et al. 2009]. Em diferentes contextos, tópicos têm significados diferentes, por exemplo, em um *corpus* que evoluiu ao longo do tempo, pode acontecer de alguns tópicos inicialmente ligados entre si serem separados em algum momento, e essa dinâmica pode não ser capturada pelo modelo [Ramage et al. 2009]. Por fim, a modelagem de tópicos depende do componente humano, devendo este, preferencialmente, ser um especialista no domínio, para ser possível identificar um número razoável k de tópicos, além de ser capaz de avaliar se eles possuem relação com o que se pretende estudar [Ramage et al. 2009]. Considerando que não existe forma estritamente objetiva para escolher os k tópicos, sendo a qualidade desta tarefa diretamente relacionada à habilidade do avaliador, que deve saber de antemão quais tópicos podem existir no *corpus* e suas respectivas interpretações, aumenta a probabilidade de enviesar as análises posteriores [Ramage et al. 2009]. No entanto, existe heurísticas, como por exemplo, a medida de coerência proposta em [Röder et al. 2015], que permite avaliar a qualidade dos tópicos identificados por modelos de tópicos e, assim, auxiliar na descoberta do número de tópicos em um *corpus*.

2.6. Compreensão Semântica e Emocional

A Compreensão de Linguagem Natural, ou *Natural Language Understanding* (NLU), como o próprio nome indica, auxilia na compreensão semântica de textos em linguagem natural. Para isso, a NLU, comumente, realiza duas tarefas: detecção (ou reconhecimento) de intenção e reconhecimento de entidades nomeadas. Assim, para cada texto em linguagem natural (também chamado de expressão ou, ainda, *utterance*), a NLU retorna como resultado de seu processamento a intenção do texto, isto é, o objetivo que a pessoa tinha em mente ao enviar a expressão, e as entidades nomeadas que foram reconhecidas. Além disso, também pode-se incluir a detecção de sentimentos e emoção a esse processo para ser possível obter também a compreensão emocional. Nesta seção são apresentados mais detalhes sobre as técnicas que compõem a NLU.

2.6.1. Detecção de intenção

A detecção de intenções é a parte responsável por fornecer uma interpretação geral do significado de uma expressão. Em outras palavras, a intenção nos informa o que uma expressão quer dizer. Normalmente, é uma tarefa abstraída em um processo de classifi-

cação, na qual se treina um modelo de classificação supervisionado com um conjunto de expressões rotuladas, onde para cada expressão é atribuída uma intenção correspondente.

Para dados de mídias sociais, normalmente, não temos a disposição a intenção (podendo ser uma categoria, ou assunto, ou tema, etc) dos textos compartilhados, que poderia nos auxiliar no processo de construção do conjunto de dados para treinamento do modelo de classificação. Além disso, os usuários podem compartilhar conteúdos diversos, desde de política, esporte, cultura, religião, até situações pessoais cotidianas, como suas lembranças (conhecido como *tbt*, do termo "*Throwback Thursday*"), declarações amorosas e momentos especiais. Então, uma possível pergunta seria: quais são as possíveis intenções em *corpus* de textos de mídias sociais?

Para responder a essa questão, podemos utilizar a modelagem de tópicos (como descrito na Seção 2.5), para determinar quais assuntos estão presentes no *corpus*, como mostra [Churchill and Singh 2021, Aiello et al. 2013], onde são analisados várias abordagens para mineiração de textos de mídias sociais e identificação de tópicos. Desta maneira, após identificar os tópicos presentes no *corpus* e atribuir um assunto para cada um deles, de acordo com as palavras que o melhor representam, torna-se mais simples a criação do conjunto expressões rotuladas.

De posse do conjunto de expressões rotuladas, o próximo passo é treinar algum modelo de classificação, para que ele seja capaz de inferir a intenção mais provável de uma expressão inédita (i.e., uma expressão diferente das expressões utilizadas durante o treinamento do modelo). Para isso, existe várias possibilidades, como utilizar uma arquitetura similar ao modelo LASER [Artetxe and Schwenk 2019], sendo uma rede BiLSTM, seguida de uma rede LSTM e uma camada totalmente conectada (função ativação) para realizar a classificação, que é similar ao modelo utilizado em [Yang et al. 2017]. Ou, ainda, utilizar algum modelo de *sentence embeddings* baseado na arquitetura *Transformers* (como descrito na Seção 2.4.4), que é o estado-da-arte para várias tarefas de NLP, entre elas, a classificação de textos. Assim, pode-se utilizar o modelo LaBSE [Feng et al. 2020] para obtenção dos *embeddings*, seguido de algum algoritmo de classificação (*Random Forest*, *Logistic Regression CV*, etc). Em [Ladeira et al. 2022], são avaliadas diferentes combinações de modelos de *sentence embeddings* e algoritmos de classificação para detecção de intenções, considerando 5 conjuntos distintos de expressões rotuladas.

2.6.2. Reconhecimento de entidades nomeadas

O reconhecimento de entidades nomeadas (do inglês *Named Entity Recognition*, ou NER), corresponde à tarefa de reconhecimento e categorização de termos relevantes em um texto, para enriquecimento da interpretação das informações contidas neste texto. Esses termos relevantes são os substantivos correspondentes a nomes próprios, por exemplo, pessoas, empresas, lugares, produtos, organizações, cores, valores monetários, idioma, GPE (países, estados, cidades), dentre outros.

Desta forma, a tarefa do NER é reconhecer dentre os termos de um texto escrito em linguagem natural, quais podem ser caracterizados como nomes próprios e quais categorias de nome provavelmente se referem, o que é chamado de entidades nomeadas. Há portanto, diferentes técnicas para identificar as entidades nomeadas, por exemplo, o *slot*

filling.

No *slot filling*, cada palavra (ou termo) de um texto é marcada com um *slot*. Para isso, é utilizado algum tipo de notação, por exemplo, a popular notação *in/out/begin* (IOB), onde "B" denota o início de uma entidade, "I" significa "dentro" e é usado para todas as palavras que compõem a entidade, exceto a primeira, e "O" significa ausência de entidade.

A Tabela 2.1 mostra um exemplo de *slot filling* inspirado em [Mesnil et al. 2014], que utiliza o bem conhecido *benchmark* chamado *Airline Travel Information System* (ATIS). Como podemos ver, as palavras "Boston" e "New York" foram identificadas como sendo os locais de partida e chegada, respectivamente. Além disso, a palavra "today" foi identificada como uma data. A partir deste resultado, é possível compreender com base nos *slots* identificados, quais são as cidades de origem e destino e, também, a data da viagem.

Frase	*show*	*flights*	*from*	*Boston*	*to*	*New*	*York*	*today*
Slots	O	O	O	B-dept	O	B-arr	I-arr	B-date

Tabela 2.1. Exemplo de *slot filling* inspirado em [Mesnil et al. 2014].

Para realizar esta tarefa, a abordagem mais popular é o *Conditional Random Field* (CRF) [Lafferty et al. 2001] e suas variantes. O modelo CRF de cadeia linear, é um método de classificação discriminativa capaz de prever a sequência mais provável de rótulos (ou *slots*) para uma sequência de palavras. Recentemente, as abordagens de aprendizado profundo se tornaram mais populares para preenchimento de *slots* devido à seu desempenho superior ao modelo CRF, por exemplo, utilizando RNNs (*Recurrent Neural Networks*) [Mesnil et al. 2014] ou modelos Seq2Seq (*sequence-to-sequence*) com o mecanismo de *self-attention* [Zhao and Feng 2018].

Caso você não deseje implementar seu modelo de *slot filling* do princípio, você pode se aproveitar de bibliotecas públicas, como o spaCy[13] e o Duckling[14], que já oferecem algoritmos robustos para reconhecimento de entidades em diversos idiomas.

2.6.3. Análise de sentimentos

A Análise de Sentimentos (AS), ou Mineração de Opiniões, é o estudo computacional das opiniões, atitudes e emoções de pessoas em relação a uma entidade, por exemplo, um indivíduo, evento ou tópico [Medhat et al. 2014]. Esse é um campo em NLP, que visa identificar a polaridade do sentimento (negativo, positivo ou neutro) ou emoção (e.g., raiva, antecipação, nojo, medo, alegria, tristeza, surpresa, confiança, etc.) presente em um texto [Mohammad and Turney 2013]. Além dessas tarefas principais, a AS também é usada para detecção de toxicidade, sarcasmo, análise multilíngue de sentimentos, entre outras aplicações complexas [Zhang et al. 2018]. Vale ressaltar que neste capítulo nosso foco é em conteúdo texual, mas essa tarefa vem sendo desenvolvida para outros tipos de conteúdo, como os visuais, por exemplo, na análise de sentimentos em fotos de mídias sociais [Oliveira et al. 2020].

[13]https://spacy.io/api/entityrecognizer. Último acesso em 11 de Setembro de 2022.

[14]https://github.com/facebook/duckling. Último acesso em 11 de Setembro de 2022.

Com a crescente quantidade de dados produzidos na *web* e o aumento do poder computacional, a AS tornou-se uma das áreas de pesquisa mais ativas em NLP, permeando outras áreas além da ciência da computação, como administração e ciências sociais, uma vez que opiniões são centrais para quase todas as atividades humanas e são as principais influenciadoras do comportamento [Zhang et al. 2018]. Nesse sentido, identificar o sentimento (ou opinião) em textos é muito útil para diversas aplicações, principalmente, para aplicações com tomada de decisão em larga escala. Por exemplo, seleção de comentários agressivos em artigos de notícias para serem moderados [Jigsaw 2022], extração da opinião pública sobre um candidato ou partido político [Pang et al. 2008], priorização de respostas a avaliações negativas de produtos, para diminuir possíveis impactos negativos no comportamento de consumo do produto ou marca [Bougie et al. 2003] ou estudar o impacto do sentimento em revisões (*reviews*) de hotéis visando guiar usuários nas suas decisões [Santos et al. 2020b]. Em tarefas como essas, a AS tem duas vantagens em destaque: permite análises em larga escala e reduz a subjetividade provocada por avaliadores humanos.

A AS pode ser realizada em três níveis diferentes: (i) nível de documento; (ii) nível de frase; ou (iii) nível de aspecto [Medhat et al. 2014]. No primeiro caso, a unidade básica de onde a emoção é extraída é o documento, partindo da premissa de que todo o documento expressa a opinião sobre uma única entidade [Zhang et al. 2018]. No segundo caso, a extração é feita a nível de sentença dentro de um documento [Medhat et al. 2014], partindo da premissa de que a sentença possui uma opinião sobre uma entidade [Zhang et al. 2018]. Nos dois primeiros níveis, não é possível obter detalhes importantes para alguns tipos de aplicações em que opiniões sobre diferentes entidades coexistem em um mesmo documento ou sentença. Nesse caso, entra a AS a nível de aspectos, que visa compreender não apenas a opinião sobre a entidade como um todo, mas também sobre partes específicas da entidade, chamados de aspectos ou de alvos [Zhang et al. 2018]. Esse tipo de análise mais refinada pode ser utilizada em casos em que uma entidade pode ter opiniões diferentes para aspectos distintos. Por exemplo, considere essa avaliação de produto: "A qualidade de voz deste telefone não é boa, mas a vida útil da bateria é longa", note que ela combina um sentimento positivo sobre o aspecto "vida útil da bateria", com um sentimento negativo sobre o aspecto "qualidade de voz" [Feldman 2013, Medhat et al. 2014].

As abordagens para AS podem ser divididas em três categorias, podendo ser baseadas em (i) léxicos, (ii) aprendizado de máquina, ou (iii) híbridas [Medhat et al. 2014, Zhang et al. 2018]. Na abordagem baseada em léxicos, são construídos dicionários com termos conhecidos e relacionados a sentimentos, obtidos por meio de processos estatísticos em grandes *corpus*, possibilitando a identificação de emoções em entidades por meio da ocorrência de palavras do léxico no texto a ser analisado [Medhat et al. 2014]. Já na abordagem baseada em aprendizado de máquina, são aplicados diferentes algoritmos de aprendizado supervisionado ou não supervisionado [Medhat et al. 2014, Zhang et al. 2018]. Por último, a abordagem híbrida, que é uma combinação das duas primeiras, por exemplo, em modelos que usam aprendizado de máquina para classificação de termos em larga escala, usando como base de treinamento léxicos anotados manualmente, eliminando parte do esforço manual envolvido na criação de léxicos de grande escala [Medhat et al. 2014]. A vantagem da abordagem (i), está relacionada à in-

dependência de domínio, mas com a desvantagem de ser menos precisa e demandar maior investimento de tempo na criação do léxico [Gamon and Aue 2005]. Já a abordagem (ii), possui a vantagem de geralmente ser mais precisa, uma vez que modelos de aprendizado de máquina podem identificar regularidades no *corpus*, que a simples contagem de ocorrências de categorias emocionais no texto em um léxico não conseguiria identificar [Zhang et al. 2018].

Dentre as tarefas de AS, duas são mais comuns: a detecção de polaridade do sentimento e a detecção de emoções. No primeiro caso, são identificadas as palavras em um dado texto que refletem sentimentos positivos, negativos ou neutros. Assim, é atribuído a cada palavra "relevante", uma pontuação flutuante dentro do intervalo $[-1.0; 1.0]$, sendo -1 para negativo, 0 para neutro e 1 para positivo. Já a detecção de emoção, consiste em identificar a ocorrência ou intensidade de determinadas classes de emoção em um dado texto [Medhat et al. 2014]. Para ambos os casos, dois léxicos são comumente utilizados: (i) EmoLex (*the NRC Emotion Lexicon*) [Mohammad and Turney 2013]; e (ii) LIWC (*Linguistic Inquiry Word Count*) [Pennebaker et al. 2001], que são descritos a seguir.

2.6.3.1. EmoLex

O EmoLex é um léxico em inglês que possui associações de palavras com os sentimentos e com as oito emoções primárias da teoria de Plutchik [Plutchik 1980], sendo elas: raiva, medo, antecipação, confiança, surpresa, tristeza, alegria e desgosto [Mohammad and Turney 2013]. O EmoLex foi criado em 2013 para suprir uma lacuna da época, ainda não haviam léxicos que conseguissem mapear uma grande quantidade de termos em Inglês [Mohammad and Turney 2013]. Por isso, os autores compilaram unigramas e bigramas comuns na língua Inglesa, incluindo termos do *General Inquire* [Stone et al. 1966] e do *WordNet Affect Lexicon*, em um único léxico. Então, usaram uma plataforma *crowdsourcing* para classificação manual dos termos em larga escala, construindo um dos maiores léxicos disponíveis em língua Inglesa, feito especificamente para análise de sentimentos e emoções [Mohammad and Turney 2013].

Muito mais que o léxico, esse trabalho apresentou um arcabouço usando boas práticas para a classificação de termos em larga escala, além de considerar aspectos éticos e respectivas implicações em novos trabalhos [Hipson and Mohammad 2021], servindo como base para a criação de outros léxicos derivados, que podem ser encontrados no site de um dos autores[15].

2.6.3.2. LIWC

O LIWC é uma ferramenta amplamente aplicada para identificar características linguísticas, psicológicas e sociais em textos [Pennebaker et al. 2001]. A operação básica do LIWC se resume em percorrer cada uma das palavras em um texto, procurando sua ocorrência em um dicionário (léxico) interno, para identificar a qual categoria ela pertence e, por último, calculando o percentual de ocorrências de cada categoria no texto

[15]http://saifmohammad.com/WebPages/lexicons.html. Último acesso em 29 de Agosto de 2022.

[Pennebaker et al. 2001]. Inicialmente, o dicionário do LIWC era composto apenas pelas categorias de emoções negativas e positivas em Inglês, mas com sua evolução, foram incluídas classes gramaticais e processos psicológicos, como auto-reflexão, pensamento causal, entre outras [Tausczik and Pennebaker 2010]. Também foram criados dicionários para outras línguas, incluindo português do Brasil [Balage Filho et al. 2013][16]. Quando o LIWC foi desenvolvido, o objetivo era criar um sistema eficiente que pudesse explorar tanto os processos psicológicos quanto o conteúdo [Tausczik and Pennebaker 2010]. Com isso, duas categorias amplas de palavras foram incluídas, sendo elas as de conteúdo, que exploram características psicométricas, como emoções positivas e negativas, afeto, cognição, entre outras, e as de estilo ou função, que exploram as características gramaticais, como pronomes, preposições, artigos, conjunções, verbos auxiliares, entre outras [Tausczik and Pennebaker 2010]. Por exemplo, na frase "Foi uma noite escura e tempestuosa", as palavras "noite", "escuro" e "tempestade", são palavras de conteúdo, enquanto "isso", "era", "um" e "e" são palavras de função [Tausczik and Pennebaker 2010].

Dentre as categorias de conteúdo disponíveis no LIWC, estão as emoções positivas (e.g. amor, legal, doce, etc.) e negativas (e.g. ferido, feio, desagradável, etc.), processos sociais (filha, marido, vizinho, adulto, bebê, etc.), processos cognitivos (e.g. pensar, conhecer, causa, etc.), processos perceptivos (e.g. observar, escutar, sentir, etc.), processos biológicos (e.g. comer, sangue, dor, etc.), relatividade (e.g. chega, vai, embaixo, ontem, até, fim, etc.), preocupações sociais (e.g. auditar, igreja, cozinhar, trabalhar, mestrado, etc.), consentimento (e.g. concordar, ok, etc.), não-fluências e palavras de preenchimento (e.g. hm, er, umm, certo, etc.), entre outras [Tausczik and Pennebaker 2010]. Além dessas categorias, algumas contagens estão disponíveis, como quantidade de palavras, quantidade de palavras por sentença, percentual de palavras que foram identificadas no dicionário interno, entre outras contagens bastante úteis em diferentes aplicações [Tausczik and Pennebaker 2010].

Com esse repertório, o LIWC consegue trazer um alto detalhamento sobre o texto analisado, por isso, é um léxico amplamente usado em estudos na área da psicologia e sociologia. Inclusive, a versão em inglês do dicionário oficial do LIWC foi utilizada em projetos derivados em AS, como o *SentiStrength* [Thelwall et al. 2010], para construção de sua lista interna de palavras [Balage Filho et al. 2013]. Em relação aos dicionários em português brasileiro, existem duas versões, uma criada em 2007 (*LIWC_2007pt*) e outra, baseada na primeira, criada em 2015 (*LIWC_2015pt*), ambas avaliadas comparando com outros léxicos e entre versões, respectivamente [Balage Filho et al. 2013, Carvalho et al. 2019]. Atualmente a última versão *LIWC_2015pt* é a recomendada para trabalhos em produção [Carvalho et al. 2019].

2.6.3.3. *Perspective* API

Além de léxicos, que se baseiam na contagem de palavras em categorias específicas, métodos baseados em aprendizagem de máquina podem ser aplicados para indicar o grau em que diferentes categorias de emoções incidem em textos [Zhang et al. 2018].

[16]http://143.107.183.175:21380/portlex/index.php/pt/projetos/liwc. Último acesso em 24 de Agosto de 2022.

Uma aplicação bastante útil é a identificação de toxicidade em textos de mídias sociais, focando na incidência de categorias negativas como insulto, profanidade, ameaça, racismo, etc. [Jigsaw 2022]. Comportamentos *online* não civilizados, como assédio e abuso, desencorajam interações saudáveis, levando a conflitos e experiências desagradáveis [Kumar et al. 2018]. Um caso especial é representado por comentários grosseiros, desrespeitosos ou irracionais que podem levar os usuários a não se engajarem em discussões benéficas para o entendimento mútuo sobre um determinado assunto, esse é o conceito que define o termo "toxicidade" [Jigsaw 2022].

Dada a velocidade com que as discussões por meios de comentários crescem na *web* [Kumar et al. 2018], diferentes modelos de identificação de toxicidade em larga escala foram criados para resolver desafios específicos [Srivastava et al. 2018, Almerekhi et al. 2020]. Dentre eles, está o fato desse tipo de texto conter variados graus de sutileza inerentes à linguagem, aspectos culturais, especificidades de contexto, presença de sarcasmo e uso de figuras de linguagem, que podem mascarar a real toxicidade de um comentário. Outros desafios incluem o fato de os comentários em mídias sociais serem geralmente textos curtos, contendo erros de ortografia que ocorrem de forma esparsa no conjunto de dados [Srivastava et al. 2018]. Além disso, modelos baseados em aprendizado de máquina, tais como os usados para análise de toxicidade, tendem a ser suscetíveis a ataques adversários, nos quais um usuário mal-intencionado pode ajustar seu comentário, trocando palavras tóxicas por uma variante facilmente reconhecida por um humano, mas indetectável pelo modelo, por exemplo, substituindo a palavra "estúpido" por "st.Up1d0" [Hosseini et al. 2017]. Tal mudança na entrada pode causar um distúrbio na saída do modelo, fazendo com que um comentário tóxico seja reconhecido como não tóxico [Hosseini et al. 2017]. Um último desafio está relacionado à toxicidade ter variações consideráveis entre línguas distintas, considerando aspectos culturais e da própria forma de uso da língua [Srivastava et al. 2018].

Nesse sentido, um modelo multilíngue foi disponibilizado gratuitamente por meio de uma iniciativa da empresa Google, chamada Perspective API [Jigsaw 2022]. Esse modelo pode ser acessado por meio de uma API pública, que permite identificar diferentes categorias de toxicidade em comentários *online*. Por ser uma ferramenta amplamente adotada por alguns dos principais veículos jornalísticos internacionais para moderar comentários em seus portais, além de estar disponível de forma aberta, o Perspective API é um potencial candidato para aplicações envolvendo o estudo do comportamento abusivo em mídias sociais.

O Perspective API atribui uma pontuação contínua entre 0 e 1 para diferentes categorias de toxicidade em um comentário [Jigsaw 2022]. Uma pontuação mais alta para uma determinada categoria (ou atributo), indica uma maior probabilidade de um leitor perceber que o comentário possui este atributo [Jigsaw 2022]. Por exemplo, conforme apresentado na documentação da API, um comentário como "Você é um idiota" pode receber uma pontuação de $0,8$ para o atributo TOXICIDADE, indicando que 8 entre 10 pessoas perceberiam esse comentário como tóxico [Jigsaw 2022]. Com isso, é possível usar essa pontuação, por exemplo, para priorizar a análise manual de comentários ou até remover automaticamente comentários com pontuação acima de um determinado limite [Jigsaw 2022]. A arquitetura do Perspective API é composta por modelos multilíngues baseados em *sentence embeddings* BERT treinados em dados de fóruns *online*, que são

separados em redes neurais convolucionais (CNNs) para cada um dos idiomas suportados pelo modelo - essa separação garante que os modelos para diferentes línguas possam ser treinados separadamente de forma rápida, além de permitir maior velocidade no processamento da entrada [Jigsaw 2022]. Um dos idiomas suportados é o português, inclusive, um estudo recente mostra que utilizar textos de comentários em português do Brasil atinge melhores resultados que a sua tradução para inglês nessa API [Kobellarz and Silva 2022].

O Perspective API tem os chamados atributos de produção, testados em vários domínios e treinados em quantidades significativas de comentários anotados por humanos – esses atributos estão disponíveis para Inglês, Português e dezenas de outros idiomas [Jigsaw 2022], inclusive idiomas expressivamente mais complexos para essa tarefa, como o Mandarim, Sueco e Coreano. Dentre os atributos de produção estão [Jigsaw 2022]: TOXICITY - "Um comentário rude, desrespeitoso ou irracional que provavelmente fará com que as pessoas deixem uma discussão"; SEVERE_TOXICITY - "Um comentário que é muito odioso, agressivo, desrespeitoso, ou muito provável de fazer um usuário sair de uma discussão, ou desistir de compartilhar sua perspectiva. Este atributo é muito menos sensível a formas mais leves de toxicidade, como comentários que incluem usos positivos de palavrões"; IDENTITY_ATTACK - "Comentários negativos ou de ódio direcionados a alguém por causa de sua identidade"; INSULT - "Comentário ofensivo, inflamatório ou negativo para uma pessoa ou grupo de pessoas"; PROFANITY - "Xingamentos, palavrões ou outras linguagens obscenas, ou profanas"; THREAT - "Descreve a intenção de infligir dor, lesão ou violência contra um indivíduo, ou grupo". Além desses atributos, há outros ainda experimentais, somente em inglês, que não são recomendados para uso em produção [Jigsaw 2022].

2.7. Possíveis Aplicações

Nesta seção, são apresentadas possíveis aplicações que podem ser desenvolvidas com base no conhecimento semântico extraído a partir de dados de mídias sociais.

2.7.1. Recomendação de rotas personalizadas para Cidades Inteligentes

Muitos motoristas vêm adotando sistemas de recomendação de rotas durante seus deslocamentos diários, principalmente em grandes centros urbanos, onde a mobilidade tende a ser um desafio devido à constantes congestionamentos, interdições para obras, baixa qualidade no transporte público, entre outros problemas. Ao utilizar tais sistemas, o motorista é capaz de verificar possíveis rotas com tempos de viagem mais rápidos, levando em consideração principalmente a fluidez do trânsito nas vias que compõem as rotas sugeridas. Dois sistemas muito conhecidos são o Waze e o Google Maps, que consideram as condições de tráfego históricas e atuais, a distância a ser percorrida e, até mesmo, eventos ocasionais (como acidentes e bloqueios) reportados por seus usuários para recomendar rotas mais rápidas. No entanto, pessoas diferentes têm preferências diferentes para planejar suas rotas com base em vários aspectos, como distância percorrida, tempo de viagem, consumo de combustível, qualidade do asfalto, segurança e beleza das vias e seu entorno. Aspectos relacionados a mobilidade são normalmente estimados e apresentados aos motoristas, para que eles possam escolher a rota desejada. Enquanto outros aspectos das cidades geralmente são mais difíceis de coletar e interpretar, como o nível de segurança dos locais ou a beleza das vias, e, por isso, normalmente não são levados em consideração

para a recomendação.

Considerando o aspecto de segurança, é comum vermos notícias de motoristas que ao seguir orientações do sistema de recomendação acabaram transitando por regiões consideradas perigosas e sofreram algum tipo de incidente. Por exemplo, um casal que seguia uma rota recomendada pelo Waze foi baleado ao entrar em uma área perigosa no Rio de Janeiro, Brasil [Phillips 2017]. Episódios como esse mostram a importância de considerar o nível de segurança associado às rotas além da mobilidade.

Normalmente, o nível de segurança de uma determinada área da cidade, como um bairro ou distrito, é estimada com base nas ocorrências de crime registradas pelos órgãos oficiais, como as forças policias e secretarias de segurança pública. Para muitos municípos, é possível obter acesso a tais dados via portais da iniciativa de dados abertos. Apesar desses dados oficiais representarem uma "impressão digital" da insegurança das àreas urbanas e, por isso, serem imprescindíveis para elaboração e execução de políticas públicas, eles podem não agregar muito valor aos sistemas de recomendação de rotas por uma série de motivos, tais como:

- **Periodicidade**: a periodicidade em que os dados são disponibilizados pode não atender a necessidade do sistema de recomendação. Por exemplo, alguns municípios atualizam mensalmente os dados com novos registros de ocorrência, mas um crime que aconteceu um mês atrás pode não ter impacto em uma rota a ser sugerida no momento atual. Para o sistema de recomendação, quanto menor for o intervalo de tempo (minutos ou horas) entre a ocorrência de um crime e a obtenção dessa informação, melhor será o serviço prestado por ele, uma vez que ele pode evitar sugerir rotas no entorno da área onde o crime esteja ocorrendo, trazendo assim mais segurança aos seus usuários.
- **Isolamento de áreas**: regiões onde historicamente há mais insegurança podem ser completamente evitadas pelos sistemas de recomendação, o que potencialmente impactará tanto os motoristas, por terem que percorrer distâncias maiores para contornar tais áreas, mas também a comunidade local, que deixará de ter acesso a um fluxo maior de pessoas transitando pelo seu bairro e, assim, reduzindo sua capacidade de gerar mais renda por meio de seus serviços e produtos, e consequentemente, podendo agravar problemas econômicos e sociais dessas regiões.

Nesse sentido, melhor do que ter o conhecimento histórico sobre ocorrências de crime, para os sistemas de recomendação é mais importante detectar eventos de insegurança enquanto eles acontecem ou, pelo menos, pouco tempo depois deles terem acontecido. Desta maneira, é possível ajudar seus usuários a evitarem tais áreas enquanto elas estão inseguras. Para isso, algumas mídias sociais, como o Twitter e Facebook, podem ser uma rica fonte de dados, uma vez que seus usuários costumam postar relatos de crimes e incidentes sofridos e/ou testemunhados por eles para alertar seus contatos [Lal et al. 2020, von Nordheim et al. 2018].

Diferente dos dados oficiais, que são estruturados, manualmente gerados por especialistas e referente apenas a ocorrências de crime, o conteúdo proveniente de mídias sociais é diversificado, abrangendo diversos temas, como política, esportes, cultura, religião

e etc, tornando a tarefa de extração de conteúdo sobre um determinado tema muito desafiadora. Para endereçar esse desafio, em [Santos et al. 2020a] é apresentado um *framework* que combina diversas técnicas de NLP descritas aqui, indo desde o pré-processamento textual e geração de *word embeddings*, até detecção de comunidades em grafos, agrupamento de textos e análise de sentimentos.

Por meio do *framework*, os autores conseguiram mapear e extrair de dados de *Location-Based Social Networks* (LBSNs) algumas percepções coletivas de áreas urbanas, por exemplo, quais bairros eram considerados mais agressivos e violentos, em contraste com outros que eram considerados mais alegres, respeitosos e espetaculares. Para analisar a establidade da percepção ao longo do tempo, os autores conduziram uma análise temporal, onde foi possível identificar que a maioria das perceções se mantêm com poucas alterações independentemente do período avaliado. Também foi realizada uma análise comparativa com base em um conjunto de dados público, que contém as percepções dos voluntários sobre áreas urbanas expressas em um experimento controlado. Foi possível observar que ambos os resultados apresentam um nível de concordância muito semelhante.

Desta maneira, [Santos et al. 2020a] mostram que é possível utilizar o conhecimento extraído por esse *framework*, para realizar recomendações personalizadas considerando as preferências e necessidades dos usuários. Ou, ainda, pode-se considerar outras técnicas descritas neste trabalho além das utilizadas em [Santos et al. 2020a], para obter resultados mais relevantes para os sistemas de recomendação, uma vez que os autores não consideraram a utilização para uma aplicação em específico.

2.7.2. Polarização Política

Em situações politicamente polarizadas, a compreensão do papel dos principais atores, seu grau de influência e o tipo de conteúdo que espalham em mídias sociais é fundamental para o estudo da dinâmica deste fenômeno no ambiente *online* [Kobellarz et al. 2022]. Nesse sentido, a análise de tópicos e intenção de conteúdos disseminados por atores influentes e a respectiva reação de quem consome tais conteúdos, seja por meio de compartilhamentos ou comentários, são alguns exemplos de casos em que a análise textual pode trazer contribuições importantes para a compreensão de comportamentos polarizantes em mídias sociais.

Nesse âmbito, a hipótese dos "filtros-bolha" sugere uma intensificação da polarização gerada por mecanismos que levam indivíduos a serem expostos de forma seletiva a informações de seu interesse, com objetivo de otimizar o engajamento, em detrimento daquelas que expõe indivíduos à diversidade de opiniões [Pariser 2011]. Uma possível forma de mitigar seus efeitos, seria por meio da criação de conexões cruzadas entre grupos politicamente opostos, criando oportunidades de contato intergrupal e troca de informações [Burt 2003]. Esse é o papel dos "intermediadores", usuários capazes de criar pontes entre bolhas em uma rede social [Burt 2003, Yan 2019].

Dada a importância dos intermediadores, a pesquisa de Kobellarz *et al.* (2022) [Kobellarz et al. 2022] teve como objetivo compreender como usuários de *sites* de redes sociais se engajam com conteúdos compartilhados por intermediadores, sua fonte (domínio) e respectivos tópicos. Essa pesquisa teve como objetos de estudo a eleição presiden-

cial brasileira de 2018 e a eleição federal canadense de 2019, duas situações políticas com diferentes níveis de polarização. Os dados usados nas análises foram obtidos por meio da API de *streaming* do Twitter usando como critério de filtragem *hashtags* que contemplassem de forma equilibrada os diferentes candidatos e visões políticas. Foram coletados dados durante 6 semanas, incluindo o período antes e depois do dia de votação em cada situação eleitoral.

A polaridade de cada usuário foi inferida de acordo com a orientação política das *hashtags* aplicadas em seus *tweets* usando uma métrica chamada $P(H)$, cujo resultado é um valor no intervalo contínuo $[-1,0;+1,0]$, em que valores próximos de $-1,0$ ou $+1,0$ representam uma inclinação maior para a esquerda ou direita, respectivamente. Em seguida, foram construídas redes de retuítes para cada uma das semanas, sobre as quais foram identificados os intermediadores mais representativos por meio de uma métrica de centralidade criada pelos autores chamada "*intergroup bridging centrality*" ou, em português, "centralidade de ponte intergrupo". Essa métrica permitiu selecionar em cada semana os 100 usuários com contas verificadas que tiveram maior efetividade em distribuir conteúdos até grupos com posicionamentos políticos diferentes. Para efeitos de convenção, estes usuários foram denominados *bubble reachers*.

Em seguida, foram mantidos apenas os *retweets* feitos de conteúdos compartilhados por *bubble reachers* que contivessem *links* para *sites* de notícias, cujo conteúdo foi extraído por meio do uso de uma biblioteca de raspagem de dados chamada *news-please*[17]. O conteúdo textual das notícias foi pré-processado na seguinte ordem: (1) remoção de *stop-words* de listas específicas para os idiomas de cada conjunto de dados, sendo português para o caso brasileiro e inglês para o caso canadense, usando a biblioteca NLTK[18]; (2) remoção de *tokens* que não fossem substantivos, nomes próprios, adjetivos ou verbos usando modelos de *POS-tagging* pré-treinados específicos para os idiomas de cada conjunto de dados usando a biblioteca Spacy[19]; (3) *stemming* de *tokens* utilizando modelos pré-treinados para os idiomas de cada país através da biblioteca Spacy; (4) transformação de *tokens* em minúsculas e remoção de caracteres especiais e acentos; (5) extração de bi e trigramas usando a biblioteca Gensim[20].

Após a limpeza dos dados, os tópicos foram extraídos por meio da aplicação do algoritmo Latent Dirichlet Allocation (LDA) [Blei et al. 2003] usando a implementação da biblioteca Gensim. Este algoritmo permite identificar tópicos em um conjunto de textos, considerando que cada texto é composto por uma mistura de tópicos [Blei et al. 2003]. Como a escolha de tópicos é feita manualmente para o LDA, foram criados múltiplos modelos para cada conjunto de dados com o número de tópicos indo de 1 até 50. Dentro desse intervalo, foi escolhido o modelo cuja quantidade de tópicos teve o maior grau de similaridade semântica entre os textos por meio da métrica de coerência gerada com o modelo C_v[22], que é baseado na similaridade indireta do cosseno das palavras mais representativas de cada tópico em todos os documentos do conjunto de dados. Para todos os casos, foi utilizada a mesma semente para que os resultados da identificação dos tópicos pudessem ser replicados. Com esse método, foram identificados 48 tópicos no Brasil e 26

[17]https://pypi.org/project/news-please. Último acesso em 11 de Setembro de 2022.
[18]https://www.nltk.org. Último acesso em 11 de Setembro de 2022.
[19]https://spacy.io. Último acesso em 11 de Setembro de 2022.
[20]https://radimrehurek.com/gensim. Último acesso em 11 de Setembro de 2022.

no Canadá dentre as 338 e 484 notícias obtidas em cada uma desses países. Após a extração de tópicos com este método, foi analisada a dominância de tópicos de cada conteúdo, a partir do qual foi verificado que 87% e 68% dos conteúdos nos conjuntos de dados brasileiro e canadense, respectivamente, eram dominados por apenas um tópico, com um domínio de pelo menos 80% sobre outros tópicos. Considerando esse resultado, foi extraído apenas o tema dominante de cada conteúdo, verificando-se a quantidade média de conteúdos ligados a cada tema dominante. No caso brasileiro, encontrou-se uma quantidade média de $6,5$ ($\sigma = 2,5$) conteúdos por tema dominante e, no caso canadense, uma média de $15,0$ ($\sigma = 4,1$) conteúdos por tema dominante. Esses resultados indicam que a maioria dos tópicos estava bem definida (menos nebulosa) e distribuída uniformemente entre os conteúdos. Por fim, os autores avaliaram manualmente todos os tópicos identificados e chegaram a um consenso sobre se os tópicos estavam diretamente relacionados às respectivas situações políticas.

Para seguir com as análises, foram definidas três entidades de interesse: (i) o conteúdo, representado pelo *link* da notícia, (ii) o domínio, representando a fonte que publicou o conteúdo e (iii) o tópico, representado pelo tópico latente dominante ligado ao conteúdo. Assim como os usuários, as entidades também tiveram sua polaridade estimada. Para isso foi calculada a polaridade ($P(H)$) média ponderada dos usuários que compartilharam a respectiva entidade, gerando uma métrica chamada $RP(H)$ (polaridade relativa), cuja interpretação é a mesma da métrica $P(H)$. Dentre os resultados foi identificado que dentre os *bubble reachers* mais representativos, estavam contas de veículos jornalísticos, jornalistas, personalidades políticas e ativistas políticos, mas somente os veículos jornalísticos neutros conseguiram sustentar essa posição durante mais de uma semana. Isso indica que esses usuários conseguiram alcançar grupos homófilos distintos com maior eficiência na rede, ao passo que sustentaram essa posição ao longo das semanas no período eleitoral. Quanto ao comportamento de usuários no compartilhamento de notícias apontadas por *links* em tweets criados por *bubble reachers*, usando testes de correlação, foi identificado que domínios com RP(H) neutro representando veículos jornalísticos conseguiram distribuir seus conteúdos de forma balanceada para usuários em grupos politicamente opostos. Apesar disso, mesmo expostos a conteúdos politicamente diversificados, usuários polarizados preferem se engajar mais frequentemente com conteúdos que favorecem seu posicionamento político. Em relação aos tópicos, não foi identificada correlação entre a polaridade de tópicos e o posicionamento político dos usuários, sendo importante apenas a polaridade do conteúdo nesse sentido.

Os resultados indicaram que, mesmo contornando os mecanismos de filtro-bolha, a criação de conexões cruzadas entre grupos distintos pode não ser suficiente para reduzir a polarização de uma rede. A evidência de que veículos de notícias tendem a distribuir com mais eficiência conteúdos a grupos polarizados pode ser interpretada no sentido de que a mídia jornalística tenta manter seu propósito de imparcialidade, pelo menos quando considerado o "viés de seleção" (*gatekeeping bias*) [D'Alessio and Allen 2000]. Esse é um ponto importante no momento atual, em que a mídia jornalística sofre constantes ataques por líderes políticos e seus seguidores, sob a alegação de que produz notícias falsas ou tendenciosas para enfraquecer seu poder [21].

[21] https://fenaj.org.br/violencia-contra-jornalistas-cresce-10577-em-2020-com-jair-bolsonaro-liderando-ataques e https://cpj.org/wp-content/uploads/2020/04/cpj. Último acesso em 11 de Setembro de 2022.

2.8. Conclusão

As mídias sociais são valiosas fontes de informação, devido ao expressivo volume de dados gerados por seus usuários e a facilidade em acessá-los de maneira programática em larga escala. E, como boa parte do conteúdo dessas fontes são textos escritos em linguagem natural, torna-se fundamental a utilização de técnicas de processamento de linguagem natural em conjunto com modelos de aprendizado de máquina para processar, analisar e extrair *insights* relevantes a partir de tais textos, que podem auxiliar no desenvolvimento de novas aplicações e serviços. Diante disso, este capítulo além de apresentar as características de famosas mídias sociais e como coletar seus dados, também introduziu os fundamentos e ferramentas das principais etapas do processo de análise de textos, fornecendo, assim, os meios necessários para desenvolver aplicações que possam tirar proveito do conhecimento semântico e emocional extraídos a partir de dados de mídias sociais. Apesar do foco em textos de mídias sociais e suas particularidades, boa parte das técnicas apresentadas também podem ser aplicadas em outras fontes textuais.

2.9. Reconhecimentos

Este capítulo foi parcialmente financiado pela CAPES - Finance Code 001, projeto GoodWeb (Bolsa 2018/23011-1 da Fundação de Amparo à Pesquisa de São Paulo - FAPESP), e CNPq (bolsa 310998/2020-4).

Referências

[Aggarwal and Zhai 2012] Aggarwal, C. C. and Zhai, C. (2012). A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer.

[Aiello et al. 2013] Aiello, L. M., Petkos, G., Martin, C., Corney, D., Papadopoulos, S., Skraba, R., Göker, A., Kompatsiaris, I., and Jaimes, A. (2013). Sensing trending topics in twitter. *IEEE Transactions on multimedia*, 15(6):1268–1282.

[Alghamdi and Alfalqi 2015] Alghamdi, R. and Alfalqi, K. (2015). A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(1).

[Almerekhi et al. 2020] Almerekhi, H., Kwak, H., Salminen, J., and Jansen, B. J. (2020). *Are These Comments Triggering? Predicting Triggers of Toxicity in Online Discussions*, page 3033–3040. Association for Computing Machinery, New York, NY, USA.

[Artetxe and Schwenk 2019] Artetxe, M. and Schwenk, H. (2019). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

[Balage Filho et al. 2013] Balage Filho, P., Pardo, T. A. S., and Aluísio, S. (2013). An evaluation of the brazilian portuguese liwc dictionary for sentiment analysis. In *Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology*.

[Barbier and Liu 2011] Barbier, G. and Liu, H. (2011). Data mining in social media. In *Social network data analytics*, pages 327–352. Springer.

[Bird 2006] Bird, S. (2006). Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics.

[Blei et al. 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

[Bojanowski et al. 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Bougie et al. 2003] Bougie, R., Pieters, R., and Zeelenberg, M. (2003). Angry customers don't come back, they get back: The experience and behavioral implications of anger and dissatisfaction in services. *J. of the acad. of mark. science*, 31(4):377–393.

[Bowman et al. 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

[Burt 2003] Burt, R. S. (2003). The social structure of competition. *Networks in the knowledge economy*, 13:57–91.

[Carvalho et al. 2019] Carvalho, F., Rodrigues, R. G., Santos, G., Cruz, P., Ferrari, L., and Guedes, G. P. (2019). Evaluating the brazilian portuguese version of the 2015 liwc lexicon with sentiment analysis in social networks. In *Anais do BRASNAM*.

[Cer et al. 2017] Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

[Cer et al. 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

[Chidambaram et al. 2018] Chidambaram, M., Yang, Y., Cer, D., Yuan, S., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836*.

[Churchill and Singh 2021] Churchill, R. and Singh, L. (2021). The evolution of topic modeling. *ACM Comput. Surv.*

[Cody et al. 2015] Cody, E. M., Reagan, A. J., Mitchell, L., Dodds, P. S., and Danforth, C. M. (2015). Climate change sentiment on twitter: An unsolicited public opinion poll. *PloS one*, 10(8):e0136092.

[Conneau et al. 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proc of EMNLP*, pages 670–680, Copenhagen, Denmark.

[Conover et al. 2011] Conover, M., Ratkiewicz, J., Francisco, M., Gonçalves, B., Menczer, F., and Flammini, A. (2011). Political polarization on twitter. In *Proceedings of the international aaai conference on web and social media*, volume 5, pages 89–96.

[D'Alessio and Allen 2000] D'Alessio, D. and Allen, M. (2000). Media bias in presidential elections: A meta-analysis. *Journal of communication*, 50(4):133–156.

[Devlin et al. 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dib 2022] Dib, F. (2022). Regular expressions 101.

[Feldman 2013] Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

[Feng et al. 2020] Feng, F., Yang, Y., Cer, D., Arivazhagan, N., and Wang, W. (2020). Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*.

[Fleiss et al. 1981] Fleiss, J. L., Levin, B., Paik, M. C., et al. (1981). The measurement of interrater agreement. *Stat meth rat and prop*, 2(212-236).

[Gamon and Aue 2005] Gamon, M. and Aue, A. (2005). Automatic identification of sentiment vocabulary: Exploiting low association with known sentiment terms. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 57–64, Ann Arbor, Michigan. Association for Computational Linguistics.

[González-Bailón and De Domenico 2021] González-Bailón, S. and De Domenico, M. (2021). Bots are less central than verified accounts during contentious political events. *Proceedings of the National Academy of Sciences*, 118(11).

[Harris 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hipson and Mohammad 2021] Hipson, W. E. and Mohammad, S. M. (2021). Emotion dynamics in movie dialogues. *PloS one*, 16(9):e0256153.

[Hofmann 2001] Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1):177–196.

[Honnibal et al. 2020] Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

[Hosseini et al. 2017] Hosseini, H., Kannan, S., Zhang, B., and Poovendran, R. (2017). Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

[Huh and Fienberg 2012] Huh, S. and Fienberg, S. E. (2012). Discriminative topic modeling based on manifold learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):1–25.

[Indurkhya and Damerau 2010] Indurkhya, N. and Damerau, F. J. (2010). *Handbook of natural language processing*. Chapman and Hall/CRC.

[Iyyer et al. 2015] Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proc. of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing*, pages 1681–1691.

[Jigsaw 2022] Jigsaw, G. (2022). Perspective api. Acessed May 31, 2022.

[Joulin et al. 2016] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[Jurafsky and Martin 2021] Jurafsky, D. and Martin, J. H. (2021). *Speech and language processing 3. Ed.*, volume 3. Pearson London.

[Kemp 2022] Kemp, S. (2022). Digital 2022: July global statshot report. https://datareportal.com/reports/digital-2022-july-global-statshot. Accessed: 2022-09-08.

[Kiros et al. 2015] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. *Advances in neural information processing systems*, 28.

[Kleene et al. 1956] Kleene, S. C. et al. (1956). Representation of events in nerve nets and finite automata. *Automata studies*, 34:3–41.

[Kobellarz and Silva 2022] Kobellarz, J. and Silva, T. H. (2022). Should we translate? evaluating toxicity in online comments when translating from portuguese to english. In *Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)*, Curitiba, Brasil.

[Kobellarz et al. 2022] Kobellarz, J. K., Brocic, M., Graeml, A. R., Silver, D., and Silva, T. H. (2022). Reaching the bubble may not be enough: news media role in online political polarization. *arXiv*.

[Kumar et al. 2018] Kumar, S., Hamilton, W. L., Leskovec, J., and Jurafsky, D. (2018). Community interaction and conflict on the web. In *Proc of WWW*, WWW '18, page 933–943, Republic and Canton of Geneva, CHE.

[Ladeira et al. 2022] Ladeira, L. Z., Santos, F., Cléopas, L., Buteneers, P., and Villas, L. (2022). Neo-nda: Neo natural language data augmentation. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 99–102. IEEE.

[Lafferty et al. 2001] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ACM International Conference on Machine Learning*, pages 282–289.

[Lal et al. 2020] Lal, S., Tiwari, L., Ranjan, R., Verma, A., Sardana, N., and Mourya, R. (2020). Analysis and classification of crime tweets. *Procedia Computer Science*, 167:1911–1919. International Conference on Computational Intelligence and Data Science.

[Landauer et al. 1998] Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

[MacQueen 1967] MacQueen, J. (1967). Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297.

[Medhat et al. 2014] Medhat, W., Hassan, A., and Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

[Mesnil et al. 2014] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., et al. (2014). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

[Messias et al. 2013] Messias, J., Schmidt, L., Oliveira, R., and Benevenuto, F. (2013). You followed my bot! transforming robots into influential users in twitter. *First Monday*.

[Mikolov et al. 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mohammad and Turney 2013] Mohammad, S. M. and Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

[Murtagh and Contreras 2012] Murtagh, F. and Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97.

[Oliveira et al. 2020] Oliveira, W. B. d., Dorini, L. B., Minetto, R., and Silva, T. H. (2020). Outdoorsent: Sentiment analysis of urban outdoor images by using semantic and deep features. *ACM Trans. Inf. Syst.*, 38(3).

[Omran et al. 2007] Omran, M. G., Engelbrecht, A. P., and Salman, A. (2007). An overview of clustering methods. *Intelligent Data Analysis*, 11(6):583–605.

[Palla et al. 2005] Palla, G., Derényi, I., Farkas, I., and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814.

[Pang et al. 2008] Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in information retrieval*, 2(1–2):1–135.

[Pariser 2011] Pariser, E. (2011). *The Filter Bubble: What The Internet Is Hiding From You*. Penguin Books Limited.

[Pennebaker et al. 2001] Pennebaker, J. W., Francis, M. E., and Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

[Pennington et al. 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

[Phillips 2017] Phillips, D. (2017). How directions on the waze app led to death in brazil's favelas. https://goo.gl/QxHdKv. Accessed: 2022-08-29.

[Plutchik 1980] Plutchik, R. (1980). A general psychoevolutionary theory of emotion. In *Theories of emotion*, pages 3–33. Elsevier.

[Pombo 2022] Pombo, O. (2022). Morte de sócrates.

[Porter 1980] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*.

[Proferes et al. 2021] Proferes, N., Jones, N., Gilbert, S., Fiesler, C., and Zimmer, M. (2021). Studying reddit: A systematic overview of disciplines, approaches, methods, and ethics. *Social Media+ Society*, 7(2):20563051211019004.

[Radford et al. 2018] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. *OpenAI Blog*.

[Ramage et al. 2009] Ramage, D., Rosen, E., Chuang, J., Manning, C. D., and McFarland, D. A. (2009). Topic modeling for the social sciences. In *NIPS 2009 workshop on applications for topic models: text and beyond*, volume 5, pages 1–4.

[Řehůřek and Sojka 2010] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proc. of LREC, Workshop*, pages 45–50, Valletta, Malta. ELRA.

[Reimers and Gurevych 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

[Reimers and Gurevych 2020] Reimers, N. and Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*.

[Rendel et al. 2016] Rendel, A., Fernandez, R., Hoory, R., and Ramabhadran, B. (2016). Using continuous lexical embeddings to improve symbolic-prosody prediction in a text-to-speech front-end. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE.

[Robertson 2004] Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*.

[Röder et al. 2015] Röder, M., Both, A., and Hinneburg, A. (2015). Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408.

[Rodrigues 1997] Rodrigues, N. (1997). *Flor de obsessão: as 1000 melhores frases de Nelson Rodrigues*, volume 12. Companhia

das Letras.

[Rogers 2009] Rogers, R. (2009). *The end of the virtual: Digital methods*, volume 339. Amsterdam University Press.

[Santos et al. 2020a] Santos, F. A., Silva, T. H., Loureiro, A. A., and Villas, L. A. (2020a). Automatic extraction of urban outdoor perception from geolocated free texts. *Social Network Analysis and Mining*, 10(1):1–23.

[Santos et al. 2020b] Santos, G., Mota, V. F. S., Benevenuto, F., and Silva, T. H. (2020b). Neutrality may matter: sentiment analysis in reviews of Airbnb, Booking, and Couchsurfing in Brazil and USA. *Social Network Analysis and Mining*, 10(1):45.

[Schütze et al. 2008] Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

[Sculley 2010] Sculley, D. (2010). Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178.

[Silva et al. 2019] Silva, T. H., Viana, A. C., Benevenuto, F., Villas, L., Salles, J., Loureiro, A., and Quercia, D. (2019). Urban computing leveraging location-based social network data: A survey. *ACM Comput. Surv.*, 52(1):17:1–17:39.

[Sivakumar et al. 2020] Sivakumar, S., Videla, L. S., Kumar, T. R., Nagaraj, J., Itnal, S., and Haritha, D. (2020). Review on word2vec word embedding neural net. In *Proc of ICOSEC*, pages 282–290. IEEE.

[Socher et al. 2013] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc of EMNLP*, pages 1631–1642.

[Sparck Jones 1972] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

[Srivastava et al. 2018] Srivastava, S., Khurana, P., and Tewari, V. (2018). Identifying aggression and toxicity in comments using capsule network. In *Proc of TRAC*, pages 98–105, Santa Fe, New Mexico, USA.

[Stats 2022] Stats, I. L. (2022). Twitter usage statistics. https://www.internetlivestats.com/twitter-statistics/. Accessed: 2022-09-08.

[Stone et al. 1966] Stone, P. J., Dunphy, D. C., and Smith, M. S. (1966). The general inquirer: A computer approach to content analysis. *Journal of Regional Science*.

[Tan et al. 2018] Tan, P.-N., Steinbach, M., and Kumar, V. (2018). *Introduction to data mining*. Pearson Education, 2nd edition.

[Tausczik and Pennebaker 2010] Tausczik, Y. R. and Pennebaker, J. W. (2010). The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

[Terragni et al. 2021] Terragni, S., Fersini, E., Galuzzi, B. G., Tropeano, P., and Candelieri, A. (2021). OCTIS: Comparing and optimizing topic models is simple! In *Proc. of the 16th Conference of the European Chapter of the ACL*, pages 263–270.

[Thelwall et al. 2010] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American society for information science and technology*, 61(12):2544–2558.

[Thompson 1968] Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422.

[Tufekci 2014] Tufekci, Z. (2014). Big questions for social media big data: Representativeness, validity and other methodological pitfalls. In *Eighth international AAAI conference on weblogs and social media*.

[Ushioda 1996] Ushioda, A. (1996). Hierarchical clustering of words and application to nlp tasks. In *Fourth Workshop on Very Large Corpora*.

[Vaswani et al. 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Vayansky and Kumar 2020] Vayansky, I. and Kumar, S. A. (2020). A review of topic modeling methods. *Information Systems*, 94:101582.

[von Nordheim et al. 2018] von Nordheim, G., Boczek, K., and Koppers, L. (2018). Sourcing the sources. *Digital Journalism*, 6(7):807–828.

[Wang and McCallum 2006] Wang, X. and McCallum, A. (2006). Topics over time: a non-markov continuous-time model of topical trends. In *Proc. of the 12th ACM SIGKDD*, pages 424–433.

[Weedon et al. 2017] Weedon, J., Nuland, W., and Stamos, A. (2017). Information operations and facebook. *Retrieved from Facebook: https://fbnewsroomus. files. wordpress. com/2017/04/facebook-and-information-operations-v1. pdf*.

[Wen et al. 2016] Wen, Z., Li, Y., and Tao, J. (2016). The parameterized phoneme identity feature as a continuous real-valued vector for neural network based speech synthesis. In *INTERSPEECH*.

[Yan 2019] Yan, P. (2019). Information bridges: Understanding the informational role of network brokerages in polarised online discourses. In *Proc. of ICIS*, pages 377–388. Springer.

[Yang et al. 2017] Yang, X., Chen, Y.-N., Hakkani-Tür, D., Crook, P., Li, X., Gao, J., and Deng, L. (2017). End-to-end joint learning of natural language understanding and dialogue manager. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5690–5694. IEEE.

[Yang et al. 2019] Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G. H., Yuan, S., Tar, C., Sung, Y.-H., et al. (2019). Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.

[Zhang et al. 2021] Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2021). Dive into deep learning. *arXiv preprint arXiv:2106.11342*.

[Zhang et al. 2018] Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

[Zhao and Feng 2018] Zhao, L. and Feng, Z. (2018). Improving slot filling in spoken language understanding with joint pointer and attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 426–431.

[Zhao and Karypis 2002] Zhao, Y. and Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524.

[Zhu and Ghahramani 2002] Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. *Technical Report*.

[Zimmer 2020] Zimmer, M. (2020). "but the data is already public": on the ethics of research in facebook. In *The Ethics of Information Technologies*, pages 229–241. Routledge.

[Zipf 2016] Zipf, G. K. (2016). *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books.