UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Análise em Larga Escala da Evolução Temporal de Tópicos obtidos do Twitter basado em Apache Spark

Braulio Valentin Sánchez Vinces

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP
SÃO CARLOS

Braulio Valentin Sánchez Vinces

Análise em Larga Escala da Evolução Temporal de Tópicos obtidos do Twitter basado em Apache Spark

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Ricardo Marcondes Marcacini

Versão original

São Carlos

2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Sánchez Vinces, Braulio Valentin
S211a Análise em Larga Escala da Evolução Temporal de Tópicos obtidos do Twitter basado em Apache Spark / Braulio Valentin Sánchez Vinces; orientador Ricardo Marcondes Marcacini. -- São Carlos, 2022.
41 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. PLN. 2. BERT. 3. Apache Spark. 4. Tópicos. 5. Twitter. I. Marcondes Marcacini, Ricardo, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Braulio Valentin Sánchez Vinces

Apache Spark-based Large-Scale Analysis of the Temporal Evolution of Twitter Topics

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Ricardo Marcondes Marcacini

Original version

São Carlos

2022

RESUMO

VINCES, Braulio V.S. **Análise em Larga Escala da Evolução Temporal de Tópicos obtidos do Twitter baseado em Apache Spark**. 2022. 41p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Nos últimos anos, os estudos relacionados à extração de tópicos do Twitter ganharam interesse dos campos acadêmicos e empresariais. A interconexão entre usuários e informações fez desta rede social uma plataforma para a propagação de eventos em tempo real. Aplicações como gerenciamento de desastres, detecção de surtos, análise de mercado e vigilância podem requerir o suporte da extração de tópicos de uma plataforma como o Twitter. Entretanto, esta tarefa é desafiadora devido ao conteúdo curto e textual das postagens (*tweets*), ao ambiente de plataforma altamente dinâmico e ao volume de *tweets* que a propagação de um evento pode desencadear. É por estas razões que propomos uma solução de extração e modelagem de tópicos do Twitter que processe um alto volume de *tweets*, além de usar algoritmos de inteligência artificial que permitam melhorar a extração de características do texto em comparação com os algoritmos clássicos de PLN. Os experimentos realizados demonstram que esta proposta, implementada no Apache Spark e utilizando modelos BERT pré-treinados, consegue não apenas ser viável em larga escala, mas também ser ligeiramente superior aos algoritmos comumente utilizados na modelagme de tópicos, como o LDA. Finalmente, realizamos experimentos de análise temporal sobre os tópicos obtidos por esta proposta.

Palavras-chave: PLN. BERT. Apache Spark. Tópicos. Twitter.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Estamos na chamada Era Digital, em que grande parte da população mundial interage com vários sistemas digitais, o que permite a geração de dados exponencialmente (KOLAJO; DARAMOLA; ADEBIYI, 2019). Isso facilita a existência de várias fontes de dados que nos permitem explorar nossa realidade, e.g., pensamento humano e as interações sociais. O Twitter é uma rede social e uma plataforma de microbloggs *on-line* em que as pessoas se comunicam por meio de mensagens curtas chamadas *tweets* [1].

Todas as informações que esta rede armazena permitem que os usuários sejam caracterizados em diferentes perfis com base na frequência de publicação, preferências de web sites, termos ou palavras mais utilizados, partidos políticos ou dirigentes mais referenciados, religião, etc. O interessante dessa rede social é que o fluxo de informações não é estocástica ou caótica, apresenta uma estrutura. Daí o interesse ao ter sido amplamente utilizada em pesquisas acadêmicas, na aplicação de diversas técnicas de mineração de dados, processamento de linguagem natural, educação, política, entre outras áreas.

Vale ressaltar que qualquer estudo com base neste tipo de dados deve lidar com dois desafios: a grande escala na geração de tweets e que eles não são rotulados. Considerando esse cenário, este projeto de pesquisa visa investigar métodos de modelagem de *tópicos* (NUGROHO *et al.*, 2020) (entenda-se tópico como um conjunto formado por vários tweets com características comuns) presentes nos tweets; desenvolvendo uma estratégia não supervisionada e escalável para grandes fluxos de dados, ou seja, sobre plataformas de processamento distribuído em grandes clusters de computadores, em particular Apache Spark [2].

1.2 Justificativa e Motivação

O estudo de identificação de tópicos extraídos do Twitter continua a ser uma área de interesse (AZIZI *et al.*, 2021). Como plataforma pública, o Twitter é uma fonte ideal para acumular um grande número de opiniões sobre uma ampla gama de tópicos. Todas essas informações oferecem grande potencial e podem ser aproveitadas para analisar a *transição* dos tópicos identificados, ao analizar vários instantes ou momentos ordenados em uma seqüência temporal. É claro que é necessária uma abordagem de análise automatizada e não supervisionada, porém que não se limite a ambientes centralizados, já que o análise se reduziría a alguns centos ou milhares de tweets. Lembrando que a medida que mais

[1] https://www.webopedia.com/definitions/tweet/
[2] https://spark.apache.org/

dados se tornam disponíveis, os modelos analíticos empregados tendem a ter um melhor desempenho.

Embora existam diferentes métodos para extração e modelagem de tópicos para Twitter (OSTROWSKI, 2015; CASALINO *et al.*, 2018), tais métodos não foram propostos considerando grandes bases de tweets. Para este cenário, é necéssario uma solução de modelagem de tópicos que permite processamente distribuido. Neste contexto, propomos utilizar a Apache Spark como base para a solução proposta, pois é uma das plataformas de processamento distribuído mais utilizadas. Possui também bibliotecas adicionais que estendem sua usabilidade a várias tarefas de mineração de dados e inteligência artificial, o que torna esta solução verdadeiramente escalável para um grande ambiente de dados. Além disso, utilizaremos os modelos pré-treinados existentes baseados no algoritmo BERT (DEVLIN *et al.*, 2018) a fim de extrair a representação numérica de um tweet com base no contexto das palavras que o compõem, melhorando notávelmente a extração de características do texto.

1.3 Questões de Pesquisa e Objetivos

LDA (BLEI; NG; JORDAN, 2003) é um dos algoritmos do estado-da-arte mais utilizados para inferir ou detectar tópicos de uma coleção de textos ou documentos, e até mesmo a biblioteca Apache Spark MLLib tem uma implementação desta técnica disponível [3]. Por outro lado, métodos mais recentes para processamento de linguagem natural e extração de tópicos, como o BERT, não estão disponíveis e foram poucos avaliados no cenário de processamento distribuido. Com base nisso, podemos elaborar as seguintes perguntas de pesquisa:

Q1 *"Os tópicos identificados com base nesta proposta, em comparação com aqueles obtidos usando LDA, representam melhor o conteúdo daqueles tweets que os compõem?"*

Q2 *"Esta proposta é comparável em escalabilidade à versão disponível do LDA em Apache Spark?"*

Com estas perguntas em mente, definimos os seguintes objetivos:

- Implementar uma abordagem usando modelos BERT juntamente com algoritmos de agrupamento para identificar tópicos coerentes. Este objetivo está relacionado à questão de pesquisa Q1.

- Investigar um modelo BERT viável para implementar usando Apache Spark e garantir a escalabilidade da solução proposta. Este objetivo está relacionado à questão de pesquisa Q2.

[3] https://spark.apache.org/docs/latest/ml-clustering.html#latent-dirichlet-allocation-lda

A partir do modelo proposto, espera-se que os resultados sejam equivalentes ou até melhores do que os obtidos aplicando LDA, além de atingir um nível de escalabilidade competente.

Nos capítulos seguintes desenvolvemos os conceitos teóricos envolvidos na elaboração deste trabalho (Capítulo 2), assim como a descrição da metodologia de trabalho proposta (Capítulo 3), os resultados experimentais obtidos (Capítulo 4) e as conclusões alcançadas com base nas questões de pesquisa formuladas acima (Capítulo 5).

2 FUNDAMENTOS E TRABALHOS RELACIONADOS

2.1 Modelagem de Tópicos em Twitter

O problema de caracterizar o texto com base em seu conteúdo é de grande importância em áreas como aprendizagem de máquinas e processamento de linguagem natural (NLP). Esta caracterização é freqüentemente utilizada como entrada para tarefas que envolvem a recuperação, classificação ou mesmo previsão de documentos de texto (AGGARWAL, 2018). No contexto das redes sociais, em posts e mensagens é crucial para muitas tarefas, tais como detecção de notícias de última hora, recomendação de amigos, análise de sentimentos, entre outras (GHANI *et al.*, 2019). O Twitter está entre as redes sociais mais utilizadas, na qual são postados cerca de 500 milhões de tweets por dia [1]. A popularidade do Twitter como uma vasta fonte de informação tem estimulado o desenvolvimento de novos métodos de pesquisa nos campos de pesquisa antes mencionados. Por exemplo, Sankaranarayanan *et al.* (2009) demonstrou como o Twitter pode ser usado para recuperar automaticamente notícias de última hora de tweets postados, como no caso da morte de Michael Jackson, onde o primeiro tweet foi postado 20 minutos antes da chamada para o 911. Sakaki, Okazaki and Matsuo (2010) utilizou o Twitter como fonte de informações rápida o suficiente para ajudar a identificar ou localizar a ocorrência de terremotos, considerando o volume de tweets postados relacionados com o evento. Singh *et al.* (2020) explorou como o Twitter pode ser usado como uma ferramenta importante para prever o resultado de eventos políticos futuros quando combinada com técnicas de análise de sentimentos, análise de conteúdo e modelagem de tópicos. Assim como uma das fontes mais utilizadas para análise dos sentimentos (ZIMBRA *et al.*, 2018).

Com base no anteriormente exposto, um tópico pode ser visto como um conjunto de histórias relacionadas a um evento real. No entanto, quando os documentos não possuem anotações ou etiquetas de categoria, como muitas vezes acontece com as fontes de microblogging, só abordagens não supervisionadas descobrem os tópicos diretamente das características do texto nos documentos. Considerando uma coleção de documentos, um tópico é formalmente definido como uma distribuição sobre um conjunto fixo de termos, i.e., vocabulário, onde cada documento da coleção é uma mistura de um conjunto de tópicos (WALLACH, 2006; BLEI, 2012; VAYANSKY; KUMAR, 2020). A seguir, explicamos brevemente as algumas das principais técnicas não supervisionadas utilizadas na modelagem de tópicos:

[1] https://www.dsayce.com/social-media/tweets-day/

2.1.1 *Latent semantic analysis*

LSA (DEERWESTER *et al.*, 1990) é uma técnica não-supervisionada de mineração de textos que permite identificar a estrutura semântica latente em uma coleção de documentos. A suposição básica deste algoritmo é que as palavras que têm o significado mais próximo aparecerão em um extrato de texto similar. O LSA utiliza a relação entre documentos e todos os termos únicos na coleção de documentos para contabilizar o conteúdo conceitual. Constrói uma matriz de $V_{t \times d}$, onde t é o número de termos únicos na coleção e d o número de documentos, e realiza uma decomposição nesta matriz para identificar k estruturas latentes. O LSA usa o método de *Singular Value Decomposition* para decompor V e inferir um conjunto de variáveis de indexação para determinar as estruturas latentes; estas estruturas latentes podem ser consideradas como os tópicos na coleção de documentos. É importante notar que a decomposição de V resulta em uma matriz de dimensionalidade reduzida, onde cada documento e termo é agora representado como um vetor k-dimensional no espaço derivado pelo método SVD.

O LSA continua a sendo utilizado hoje em dia para análise de conteúdo textual no Twitter. Por exemplo, Santilli, Nota and Pilato (2017) explorou seu uso para analisar textos obtidos a partir de uma pesquisa onde os participantes definiram o conceito *corage* e um conjunto de tweets contendo sinônimos de palavras relacionadas com o mesmo conceito. E como, comparando os tópicos obtidos nos dois corpus, é possível realizar um estudo semântico diferente do clássico utilizado em psicologia. Por sua vez, Nugrahaningsih, Lestari and Karolita (2020) faz uso do LSA para a identificação automática do discurso de ódio no Twitter na Indonésia.

2.1.2 *Nonnegative matrix factorizaction*

NMF é um método não supervisionado no qual uma matriz V é decomposta em duas matrizes, i.e., $V \approx WH$, cada uma com entradas não negativas, mediante algoritmos de atualização multiplicativos (FÉVOTTE; IDIER, 2011). Para uma coleção de documentos, a NMF pode identificar as estruturas temáticas ocultas da coleção encontrando os fatores de aproximação da matriz para uma matriz documento-termo. A matriz documento-termo representa a relação de cada documento com cada termo único na coleção de documentos. Seja $V_{m \times n}$ uma matriz documento-termo onde m é o número de documentos e n o número de termos únicos, o produto das matrizes $W_{m \times k}$ e $H_{k \times n}$ é a aproximação da matriz V. Portanto, $k < min(m, n)$ pode ser considerado como o número de tópicos latentes esperados.

Casalino *et al.* (2018) propõe um *framework* para extração e análise de tópicos obtidos do Twitter, na qual a decomposição NMF representa um componente crucial para a visualização dos tópicos identificados com a possibilidade de o usuário calibrar *a priori* quantos tópicos se espera encontrar. Lahoti, Garimella and Gionis (2018) usa a NMF para

modelar o problema de polarização ideológica presente nas redes sociais como o Twitter para identificar o que ele chama de espaços ideológicos latentes e poder detectar grupos de usuários por seu perfil ideológico.

2.1.3 *Latent Dirichlet Allocation*

LDA (BLEI; NG; JORDAN, 2003) é o método de modelagem de tópicos mais amplamente utilizado (NUGROHO *et al.*, 2020). É um método generativo probabilístico não supervisionado para modelagem de uma coleção de documentos. O LDA assume que cada documento pode ser representado como uma distribuição probabilística sobre tópicos latentes, e que a distribuição de tópicos em todos os documentos compartilha um processo Dirichlet (TEH, 2010) comum. Cada tópico latente no modelo LDA também é representado como uma distribuição probabilística sobre as palavras e as distribuições de palavras dos tópicos também compartilham um processo Dirichlet comum. A Figura 1 mostra o processo generativo do LDA em notação simples. Suponha que temos M documentos em uma coleção, α é o processo Dirichlet para descrever a distribuição de tópicos no documento θ, e β é o processo Dirichlet para descrever a distribuição de palavras no tópico ϕ onde k é o número de tópicos latentes, n o número de palavras no documento, e z o tópico atribuído à palavra w na iteração atual. LDA modela M de acordo com o seguinte processo generativo:

1. Para cada documento da coleção, selecionar $\theta \sim$Dir(α). Onde Dir(α) é uma distribuição Dirichlet com parâmetro α.
2. Para cada tópico, selecionar $\phi \sim$Dir(β).
3. Para cada palavra w_i no documento atual:
   - Selecionar um tópico $z_i \sim$ Multinomial(θ)
   - Selecionar uma palavra $w_i \sim$ Multinomial(ϕ_{z_i})

Neste processo generativo, as palavras nos documentos são apenas variáveis observadas, enquanto outras são variáveis latentes (ϕ e θ) e hiperparâmetros (α e β).

Podemos ver o modelo LDA como dividindo a coleção de documentos em tópicos, representando o documento como uma mistura de tópicos com uma distribuição de probabilidade representando a importância do tópico para aquele documento. Os tópicos, por sua vez, são representados como uma combinação de palavras com uma probabilidade que representa a importância da palavra para aquele tópico.

Deve-se notar que estes métodos são projetados para serem aplicados a documentos que são suficientemente longos para extrair o máximo de informações corretas possíveis por documento. Quando aplicados diretamente a textos em plataformas como o Twitter, os quais são geralmente curtos e freqüentemente com ruído, estes métodos geram resultados

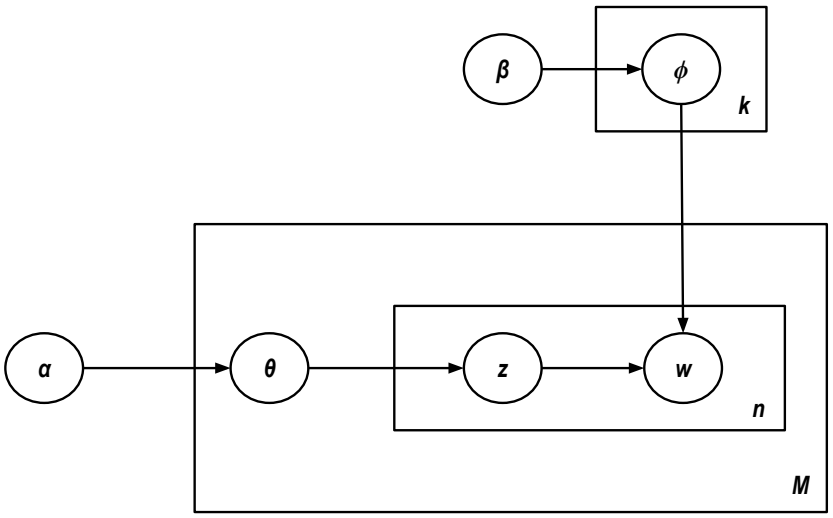

Figura 1 – Modelo LDA.

com tópicos pouco informativos e difíceis de interpretar (NEGARA; TRIADI; ANDRYANI, 2019).

O trabalho de Ostrowski (2015) utiliza LDA para extrair tópicos de tweets a fim de analisar tendências e seu significado para uso na gestão de relacionamento com o cliente. Por sua vez, Yang and Zhang (2018) combina LDA com técnicas de análise de sentimentos para propor um método computacionalmente eficiente para identificar emoções e sentimentos mistos em cada tweet, com a finalidade de que eles possam ser resumidos e claramente compreendidos.

2.2 Apache Spark para Processamento Distribuido em Larga Escala

Para enfrentar o desafio de lidar com uma fonte de dados como o Twitter, pensamos em utilizar a plataforma Apache Spark. Spark permite o processamento de dados com base no modelo *MapReduce* (DEAN; GHEMAWAT, 2008), que é composto de duas categorias principais de tarefas chamadas *Map* e *Reduce*. Dado um conjunto de dados de entrada, uma função *Map* processa os dados e retorna pares chave-valor. Há um processo intermediário chamado *Shuffle*, que agrupa os pares chave-valor e cada grupo é enviado para a tarefa *Reduce* correspondente. Dependendo da finalidade da aplicação, o usuário pode definir suas próprias funções (*User Defined Functions*). Este modelo tira a complexidade de ter que definir como o programa é executado nos nós de computação distribuída, ou lidar com questões de tolerância a falhas, gerenciamento de memória, entre outros. A Figura 2 mostra a arquitetura do modelo MapReduce.

Spark estende este modelo, além de apresentar melhorias sobre soluções existentes, e.g., Hadoop [2]: (i) uma forma mais eficiente de agrupar tarefas entre nós de computação e

[2] https://hadoop.apache.org/

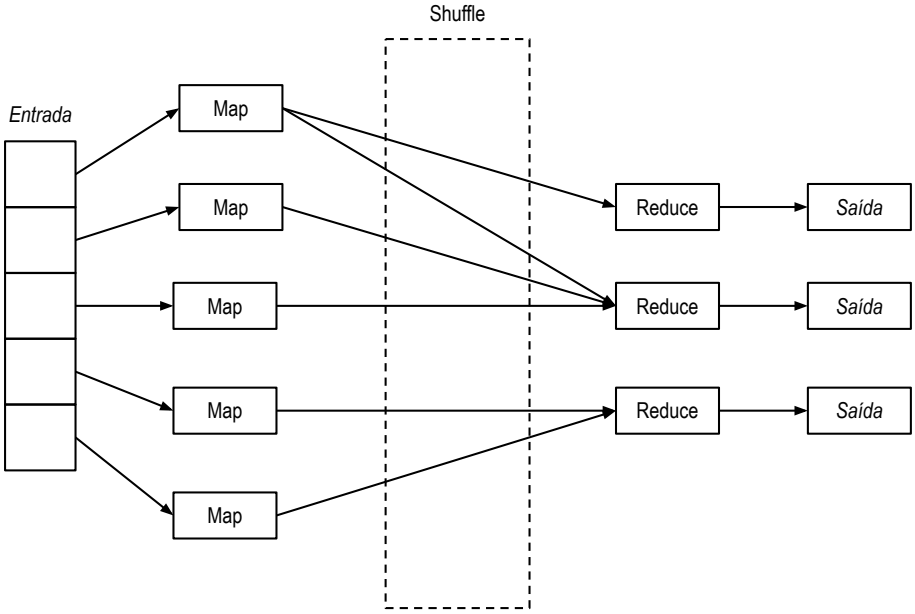

Figura 2 – Arquitetura do modelo *MapReduce*.

agendar a execução de funções, e (ii) a integração de recursos de linguagem de programação, tais como iterações e bibliotecas de funções. A abstração de dados central de Spark é chamada de *Resilient Distributed Dataset* (ZAHARIA *et al.*, 2012), ou RDD. Um RDD é um arquivo de objetos de tipo único, e.g., arquivos de pares chave-valor. O RDD é distribuído no sentido de que normalmente é dividido em fragmentos ou *chunks* que podem ser armazenados em diferentes nós de computação. O RDD é resiliente no sentido de que temos a garantia de poder nos recuperar da perda de alguns ou todos os seus fragmentos. Entretanto, ao contrário da abstração par chave-valor do MapReduce, não há restrição quanto ao tipo de elementos que compõem um RDD. Já um programa Spark consiste de uma seqüência de etapas, cada uma das quais aplica uma *UDF* a um RDD para produzir outro RDD, ou para produzir um resultado que é passado de volta para um aplicativo que foi chamado dentro de um programa Spark. Essas operações são chamadas de *tranformations* e *actions*, respectivamente.

Apache Spark é usado em uma ampla gama de aplicações, em áreas que vão desde serviços web até biotecnologia e finanças (ZAHARIA *et al.*, 2016). Por tanto, é possível aproveitar esta generalidade oferecida pela plataforma para desenvolver uma solução que permita utilizar bibliotecas existentes na modelagem de tópicos e adaptá-las para seu uso em ambientes de processamento distribuído. Para modelagem de tópicos, Spark oferece uma implementação distribuída do modelo LDA[3].

[3] https://spark.apache.org/docs/latest/ml-clustering.html#latent-dirichlet-allocation-lda

2.3 Modelagem de Tópicos usando *Word Embeddings*

2.3.1 *Word Embeddings*

Em 2013, um grupo de pesquisadores do Google desenvolveram a rede neural *word2vec* (MIKOLOV *et al.*, 2013) para gerar representações numéricas de palavras em uma coleção de documentos. As representações numéricas, i.e., *word embeddings*, criadas pela rede *word2vec* são vetores de números reais que representão a posição da palavra no espaço vetorial, sendo o tamanho dos vetores um parâmetro da rede. Os vetores contém as informações contextuais da palavra com relação às palavras concorrentes.

A partir destes vetores *word embeddings* é possível calcular a proximidade entre palavras, bem como entre documentos de texto. Por exemplo, se duas palavras w_i e w_j estão freqüentemente rodeadas pelo mesmo conjunto de outras palavras, i.e., contexto das palavras, os seus vetores *word embeddings* e_i e e_j vão ser numericamente próximos no espaço vetorial. Ao vetorizar palavras com base no contexto, *word2vec* permite que seja possível realizar poderosas operações matemáticas sobre palavras para detectar suas similaridades.

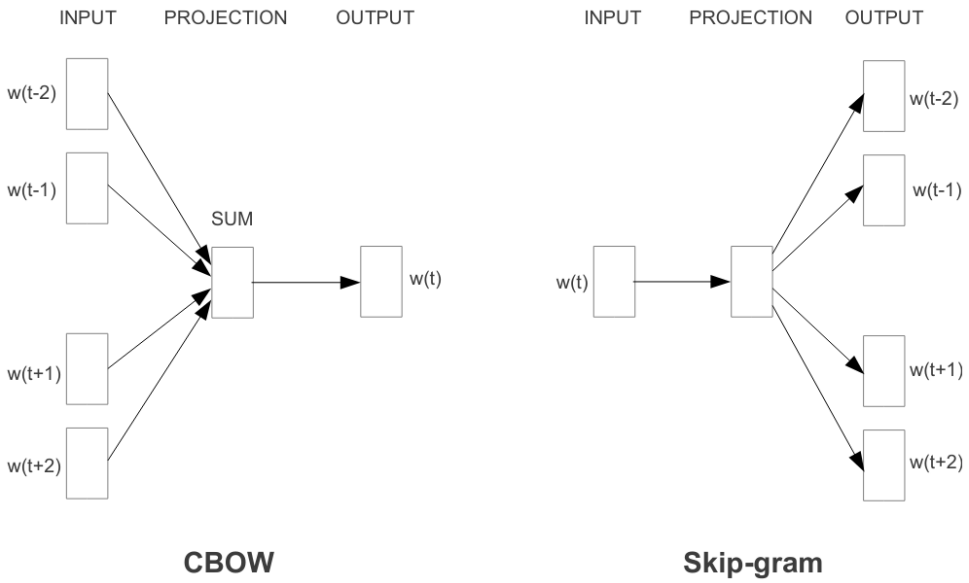

Figura 3 – Arquiteturas CBOW e Skip-gram da rede *word2vec* em (MIKOLOV *et al.*, 2013).

Na Figura 3 pode-se observar que *word2vec* é uma rede neural de camada única que usa uma camada de projeção como camada oculta. Apresenta duas arquiteturas: (i) a arquitetura CBOW prevê a palavra atual com base no contexto, e (ii) a arquitetura Skip-gram prevê as palavras circundantes dadas a palavra atual. É semelhante a um *autoencoder*[4], que codifica cada palavra em um vetor, mas em vez de treinar contra

[4] Rede neural de aprendizado profundo que é treinada para tentar reconstruir sua entrada em sua saída.

palavras da entrada através de reconstrução, as treina contra outras palavras vizinhas na coleção de entrada, o que permitea capturar grande parte das informações semânticas e sintáticas das palavras que esses vetores representam.

Estudos recentes indicam que a representação baseada em *word embeddings* obtém resultados mais satisfatórios do que aqueles obtidos por métodos tradicionais em diferentes tarefas relacionadas à análise de conteúdo textual (BAKAROV, 2018).

2.3.2 *Transformers*

Enquanto arquiteturas de aprendizado profundo, como as redes neurais recorrentes (RNN) eram amplamente utilizadas para analisar textos como dados seqüenciais, elas não conseguiram capturar dependências de longo prazo, i.e., o contexto de palavras, que estão longe de frases anteriores ou talvez até de parágrafos. Conforme a distância entre as palavras aumentava, os modelos não conseguiam identificar a relação entre elas de forma eficiente. Para superar esse problema, foi introduzido o *transformer* (VASWANI *et al.*, 2017).

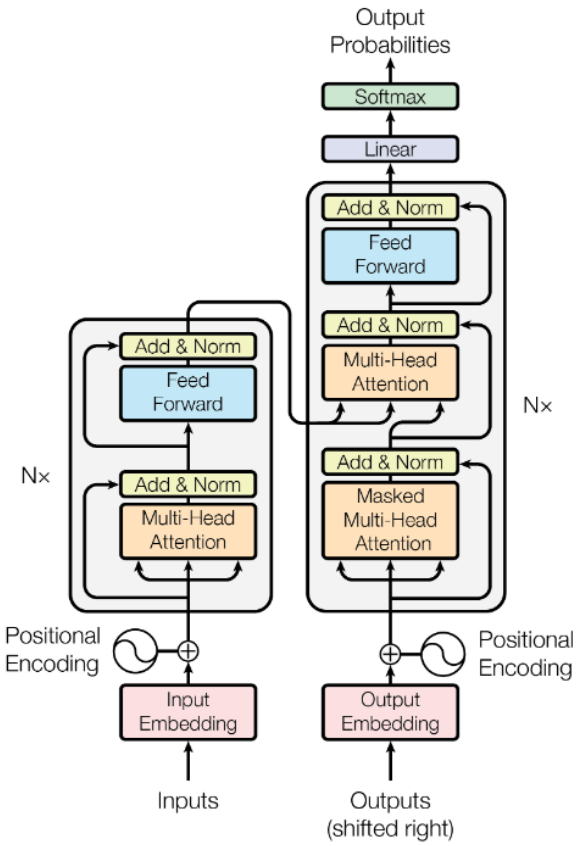

Figura 4 – Arquitetura do modelo *Transformer* em (VASWANI *et al.*, 2017).

O *transformer* é projetado para lidar com dados sequenciais, como a linguagem natural, para tarefas como tradução e resumo de texto. No entanto, ao contrário dos RNN, os *transformers* não exigem que os dados sequenciais sejam processados em ordem. Por

exemplo, se os dados de entrada são uma frase em linguagem natural, o *transformer* não precisa processar o início antes do final da frase. Devido a esse recurso, o modelo permite muito mais paralelização do que as RNN e, portanto, tempos de treinamento reduzidos.

Como visto na Figura 4, esta arquitetura está baseada no modelo *encoder-decoder*. O codificador está à esquerda e o decodificador está à direita. Tanto o codificador quanto o decodificador são compostos de módulos que podem ser empilhados um sobre o outro várias vezes, o que é descrito por Nx na figura. Observa-se que os módulos consistem principalmente de camadas *multi-head attention* e *feed forward*.

O *transformer* usa mecanismos de atenção (LUONG; PHAM; MANNING, 2015) para reunir informações sobre o contexto relevante de uma determinada palavra e depois codificam esse contexto no vetor que representa a palavra, i.e., *word embedding*. Neste modelo, cada palavra está representada por três vetores: chave, valor e consulta. O vetor-consulta busca os vetores-chave de todas as palavras que possam fornecer contexto para ela. Esses vetores-chave estão relacionados a vetores-valor que codificam mais significado sobre a palavra daquele vetor-chave. Qualquer palavra pode ter vários significados e se relacionar com outras palavras de maneiras diferentes, i.e., *multi-headed attention*.

Deve-se notar que a relação das consultas com as chaves e das chaves com os valores é diferenciável. Ou seja, o mecanismo de atenção pode aprender a remodelar a relação entre uma palavra de consulta e as palavras que fornecem contexto à medida que a rede aprende.

2.3.3 Extração de Tópicos em *Word Embeddings*

O avanço mais recente, conceitualmente simples e empiricamente poderoso no campo do processamento da linguagem natural é o novo modelo de representação de linguagem baseado em *transformers* chamado BERT (*Bidirectional Encoder Representations from Transformers*) (DEVLIN *et al.*, 2018). Apresenta duas características distintivas, o ser bidirecional, possibilitando uma melhor compreensão do contexto de uma palavra e do texto como um todo ao analisar as palavras adjacentes em ambas as direções; e sua arquitetura unificada (Figura 5). Este modelo funciona em duas etapas, pré-treinamento e ajuste fino, o que permite criar modelos para diferentes tarefas subseqüentes. Ou seja, o mesmo modelo pré-treinado pode ser ajustado para uma variedade de tarefas finais que podem não ser similares ao modelo da tarefa sobre a qual foi treinado e dar resultados próximos aos obtidos por algoritmos do estado-da-arte.

O algoritmo BERTopic (GROOTENDORST, 2020) foi desenvolvido a fim de aproveitar a capacidade de representação de texto usando modelos BERT para modelagem de tópicos em uma coleção de documentos, assumindo que os tópicos consistem em documentos semanticamente semelhantes. Este algoritmo recebe como entrada um modelo BERT pré-treinado e um corpus ou coleção de documentos. A partir deste modelo pré-treinado,

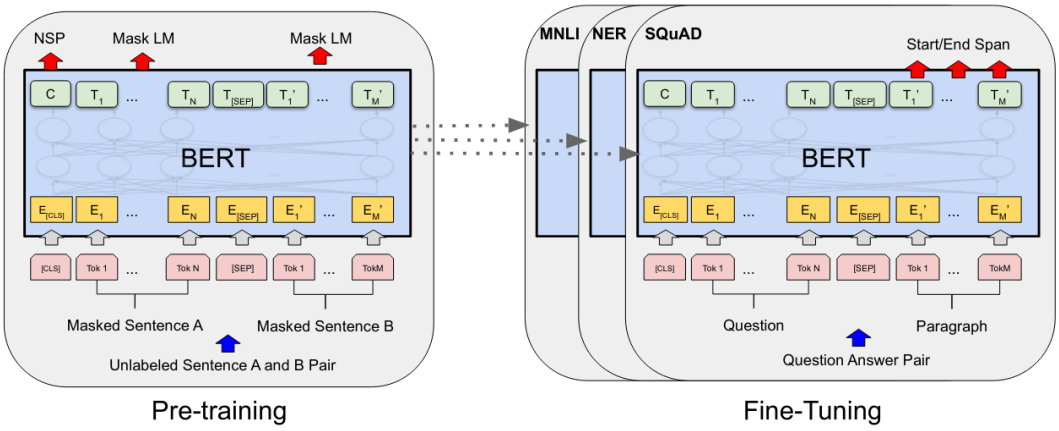

Figura 5 – Procedimentos gerais das etapas de pré-treinamento e ajuste fino do modelo BERT em (DEVLIN *et al.*, 2018).

obtém-se a representação numérica do texto sobre o qual aplica-se o algoritmo de agrupamento baseado em densidade HDBSCAN (CAMPELLO; MOULAVI; SANDER, 2013) para identificar grupos densos que representam documentos semanticamente semelhantes dentro do corpus. O BERTopic propõe uma medida que permite a representação de tópicos com base na TF-IDF[5], que permite atribuir pontuações de importância às palavras dentro de um grupo, e assim descrever os tópicos por meio das palavras mais importantes por grupo.

Este algoritmo tem sido utilizado atualmente para apoiar tarefas como análise de sentimentos (ANWAR *et al.*, 2021), ou para resumir e visualizar comentários sobre documentos legislativos utilizando modelagem de tópicos (SILVA *et al.*, 2021). No entanto, ainda é inviável utilizar esta ferramenta para analisar grandes volumes de texto, pois foi projetada para processamento centralizado ou seqüencial.

2.4 Medidas de Avaliação de Tópicos

2.4.1 Medidas de Coerência

As medidas de coerência de tópicos (RÖDER; BOTH; HINNEBURG, 2015) avaliam um único tópico medindo o grau de semelhança semântica entre as palavras mais representativas no tópico, e.g., aquelas com o TF-IDF mais alto. Estas medidas ajudam a distinguir entre tópicos que são semanticamente interpretáveis e tópicos que são artefatos de inferência estatística. Há duas medidas de coerência na literatura que têm se mostrado compatíveis com os julgamentos humanos sobre a qualidade do tópico: a medida UCI (NEWMAN *et al.*, 2010) e medida UMass (MIMNO *et al.*, 2011).

Ambas as medidas calculam a coerência de um tópico como a soma em pares das

[5] Medida estatística utilizada para avaliar a importância de uma palavra para um documento em uma coleção ou em um corpus.

pontuações de similaridade da distribuição sobre o conjunto de palavras que descrevem o tópico.

$$coherence(V) = \sum_{(v_i, v_j) \in V} score(v_i, v_j, e) \tag{2.1}$$

Onde V é o conjunto de palavras que descreve o tópico e e indica o fator de suavização que assegura que *score* retorne valores reais. Neste trabalho, usamos a medida UMass que define a função de pontuação, i.e., *score*, com base na co-ocorrência de documentos.

$$score(v_i, v_j, e) = \log \frac{D(v_i, v_j) + e}{D(v_j)} \tag{2.2}$$

Onde $D(v_i, v_j)$ conta o número de documentos contendo as palavras v_i e v_j, e $D(v_i)$ conta o número de documentos contendo a palavra v_i.

3 METODOLOGIA

3.1 *Pipeline* de trabalho

A solução proposta está composta de uma seqüência de 5 etapas. A Figura 6 mostra que estas etapas são executadas em dois ambientes de processamento: seqüencial e distribuído ou paralelo. Tanto a coleta de *tweets*, quanto a extração de *text-embeddings* e a persistência desses dados são procedimentos que não envolvem necessariamente processamento paralelo. Isto se deve principalmente a uma exigência técnica desta implementação; a biblioteca *SentenceTranformers* (REIMERS; GUREVYCH, 2020) utilizada para a extração de *text-embeddings* não tem uma implementação disponível no Apache Spark.

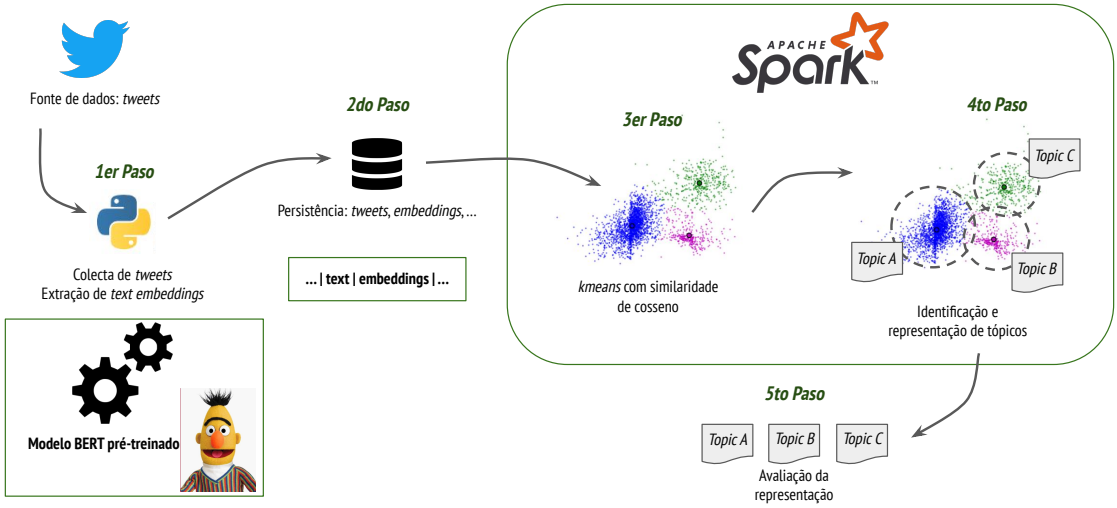

Figura 6 – *Pipeline* de trabalho

Embora uma solução similar esteja disponível na biblioteca Spark NLP[1], optamos por não usá-la, pois os algoritmos disponíveis para extração de *embeddings* de sentenças são muito limitados em comparação com a *SentenceTranformers*.

Vale mencionar que usamos Python, versão 3.8.3, como única linguagem de programação, dado seu amplo uso na construção de soluções de software tanto na indústria quanto no meio acadêmico. A seguir descrevemos cada uma das 5 etapas do nosso *pipeline* de trabalho.

3.1.1 Coleta de *tweets* - Extração de *text embeddings*

Os *tweets* são coletados através da API do Twitter em Python, usando a biblioteca *requests* para elaborar o corpo da solicitação. A fim de fazer qualquer solicitação à API

[1] https://pypi.org/project/spark-nlp/

do Twitter, a API Key e o Access Token são necessários. Neste caso, estamos utilizando uma conta com acesso educacional que permite várias solicitações por dia e com um limite relativamente maior do que uma conta de desenvolvedor. No entanto, não haveria problema em utilizar qualquer tipo de conta.

Os parâmetros considerados dentro do corpo da solicitação são[2]:

- *query*: operadores de busca entendidos pela API.
- *max_results*: número máximo de tweets por página retornados pela API.
- *tweet.fields*: campos com informações requeridas por *tweet* retornados pelo API.
- *next_token*: usado para retornar os elementos da próxima página retornada pela API.

O resultado de cada solicitação está no formato JSON. Usamos a biblioteca *json* para manipular um objeto JSON em Python e extrair as informações solicitadas em *tweets.fields*, i.e., *created_at*, *author_id*, *id*, e *text*. Isto é data de criação, autor, identificador único e texto do *tweet* respectivamente.

A extração dos *text-embeddings* é realizada utilizando um modelo BERT pré-treinado disponível na biblioteca da *SentenceTransformers*. Especificamente estamos usando o modelo denominado *paraphrase-MiniLM-L6-v2*. Depois de carregar o modelo, extraímos os *embeddings* do texto de cada *tweet*. Os *embeddings* gerados são vetores numéricos não-normalizados de 384 dimensões.

No final desta etapa, todos os *tweets* estão armazenados (junto com seus *embeddings*) em uma estrutura chamada *DataFrame* da biblioteca *pandas* e serão então salvos em um banco de dados. As colunas consideradas são: *created_at*, *author_id*, *id*, *text* e *embeddings*.

3.1.2 Persistência

O objetivo de armazenar os *tweets* e *embeddings* em um banco de dados é disponibilizar estas informações para serem carregadas e processadas em um ambiente distribuído por Apache Spark. O modelo de dados utilizado é simples:

- tweets(created_at:string, author_id:string, id:string, text:string, embeddings:JSON)

Estamos utilizando o banco de dados relacional PostgreSQL porque este motor permite armazenar e manipular dados em formato JSON nativamente[3]. A coluna de *embeddings* está no formato JSON para facilitar o carregamento do vetor numérico de 384

[2] https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/guides/standard-operators
[3] https://www.postgresql.org/docs/14/functions-json.html

dimensões que o Apache Spark executará. Desta forma, conseguimos reduzir o tamanho em bytes da tabela tweets, evitando criar 384 colunas adicionais apenas para armazenar um número em precisão de ponto flutuante em cada uma.

A conexão com o banco de dados e a inserção de cada *tweet* com suas respectivas informações é feita usando as bibliotecas *psycopg2* e *SQLAlchemy*. Vale a pena mencionar que não há ordem de inserção no banco de dados. Qualquer análise de temporalidade nos *tweets* pode ser realizada utilizando a coluna *created_at*.

No final desta etapa, todos os *tweets* coletados estarão disponíveis para processamento posterior, acessando uma única tabela em um banco de dados relacional. Como o objetivo deste trabalho não é uma análise histórica como as realizadas em *data warehouses*, o espaço ocupado pelas informações armazenadas será recuperado após o final de todo o processamento proposto em nosso *pipeline*.

3.1.3 Agrupamento no espaço vetorial

A Figura 7 dá uma visão geral do ecossistema que o Apache Spark tem, até o momento, para desenvolver soluções analíticas de dados em vários formatos de forma distribuída. A solução proposta utiliza duas das quatro bibliotecas que compõem o ecossistema dito. O Apache Spark é escrito inteiramente na linguagem de programação Scala. O *PySpark* foi lançado para apoiar a colaboração da Apache Spark e Python, e atualmente é sua API Python.

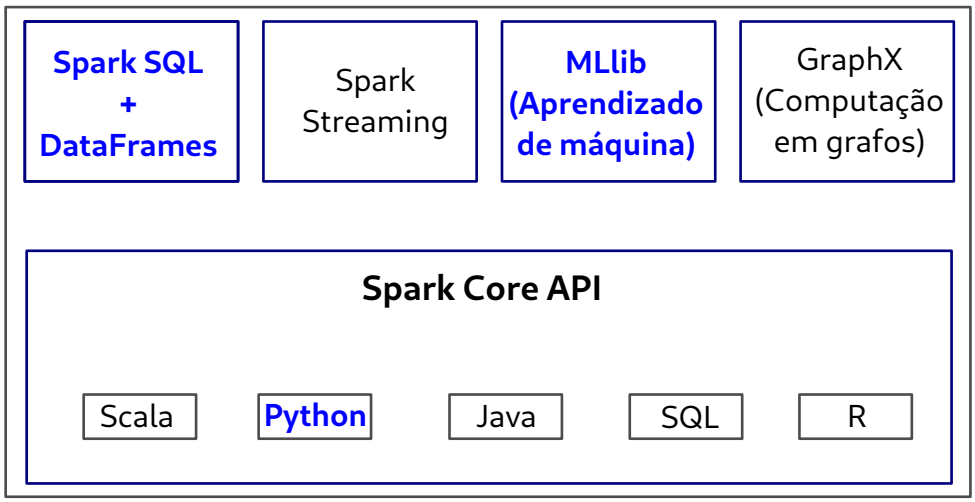

Figura 7 – Ecosistema do Apache Spark.

A fim de carregar as informações armazenadas na tabela de tweets e processá-las no ambiente distribuído do Apache Spark, executamos os seguintes passos:

- Estabelecer uma SparkSession[4] com o nó mestre de um cluster de Apache Spark.

[4] https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/spark_session.html

- Carregar os dados por meio da sessão criada usando a função *read* seguida da função *jdbc*.

Os dados são carregados em um DataFrame[5] de Spark. Na Figura 8 podemos ver uma amostra do DataFrame gerado.

```
+-------------------+-------------------+--------------------------------------------------+--------------------------------------------------+
|                 id|         created_at|                                              text|                                         embedding|
+-------------------+-------------------+--------------------------------------------------+--------------------------------------------------+
|1540322752732794884|2022-06-24 10:15:36|@Angelodireita @UOL Falta de informação da niss...|[-0.0640762001, 0.2191245258, -0.1708070636, 0....|
|1540322724521861120|2022-06-24 10:15:29|     vacina do krl mds parece q meu braço vai cair|[-0.099484548, 0.0791809931, -0.0342390351, -0....|
|1540322709472792577|2022-06-24 10:15:26|Não importa o que seja feito (Vacina, tratament...|[-0.1054416224, 0.2029563934, -0.0803257152, -0...|
|1540322642858754048|2022-06-24 10:15:10|4 Dose ✓ (Janssen Farmacêutica )
Viva a Vacina...|[-0.1962041408, 0.1915365905, -0.2146522254, -0...|
|1540322635191664641|2022-06-24 10:15:08|Vacina contra influenza poderá ser tomada por q...|[-0.1096451283, 0.0743908957, -0.3671946824, -0...|
|1540322630259073024|2022-06-24 10:15:07|              @maipurper a vacina é brabaaaaa ✨ |[-0.1387689263, 0.2135624737, -0.3137665689, -0...|
|1540322534763241479|2022-06-24 10:14:44|obrigada ciência por ter feito a vacina e tudo ...|[-0.1571368575, 0.1418645531, -0.0909909382, -0...|
|1540322534406733825|2022-06-24 10:14:44|@CNNBrasil Antivaxx que tomam vacina. So no Bra...|[-0.0481293947, 0.0508625321, -0.3289863169, -0...|
|1540322532053684224|2022-06-24 10:14:44|@CNNBrasil Estudo feito por quem vendeu a vacin...|[-0.2857069969, 0.2387859225, -0.3737746477, -0...|
|1540322510532665350|2022-06-24 10:14:38|Fui lá tomar minha vacina, como as outras 3 dos...|[0.2125942856, 0.1861202866, -0.2054792792, -0....|
+-------------------+-------------------+--------------------------------------------------+--------------------------------------------------+
```

Figura 8 – *Tweets* carregados no DataFrame (amostragem de 10 *tweets*).

É neste DataFrame que realizamos a tarefa de agrupar todos os *tweets* pré-processados utilizando o espaço vetorial de 384 dimensões criado pelos *text-embeddings*. Para isso, usamos o algoritmo *KMeans*[6] já disponível no Apache Spark juntamente com a distância cosseno como métrica de similaridade.

Na Figura 9 podemos ver o resultado final desta etapa. Cada *tweet* é classificado em um grupo respectivo, especificado na coluna *cluster* do DataFrame recém-gerado.

```
+-------------------+-------------------+-------------------+-------------------+-------+
|                 id|         created_at|               text|           features|cluster|
+-------------------+-------------------+-------------------+-------------------+-------+
|1535376307076452354|2022-06-10 18:40:12|Capital paulista ...|[-0.1015244722,0....|      8|
|1535376290433204224|2022-06-10 18:40:08|tive que ir embor...|[-0.0496278554,0....|      5|
|1535376261928845312|2022-06-10 18:40:01|Barros é líder do...|[0.1037159711,0.1...|      8|
|1535376247450218496|2022-06-10 18:39:57|Barros é líder do...|[0.1185903922,0.1...|      8|
|1535376218304008193|2022-06-10 18:39:51|#CaboVerde regist...|[0.0287345126,0.5...|     13|
|1535376063320141824|2022-06-10 18:39:14|@ErikaOlivairiss ...|[-0.2523294985,0....|     17|
|1535375898844667904|2022-06-10 18:38:34|a porqueira dos e...|[-0.239273563,0.5...|     13|
|1535375892268163073|2022-06-10 18:38:33|@Rconstantino É p...|[0.0432636626,0.3...|      3|
|1535375763897208833|2022-06-10 18:38:02|@Thumbseinerd Tá ...|[-0.0647993758,0....|      5|
|1535375657135312896|2022-06-10 18:37:37|@STF_oficial Vejo...|[-0.146214366,0.4...|      3|
+-------------------+-------------------+-------------------+-------------------+-------+
```

Figura 9 – *Tweets* agrupados.

[5] https://spark.apache.org/docs/latest/sql-programming-guide.html

[6] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.clustering.KMeans.html

3.1.4 Identificação e representação de tópicos

Após termos agrupados os *tweets* por suas representações vetoriais, vamos identificar os termos que caracterizam cada agrupamento. Para isso, pré-processamos o texto que compõe cada tweet e calculamos a estatística *tf-idf*. Cada cluster identificado é um tópico composto de todos os tweets pertencentes ao mesmo cluster.

O pré-processamento é padrão em tarefas de NLP. O texto de cada *tweet* é tokenizado e depois as *stop-words* são removidas. Para isso, usamos os modelos Tokenizer [7] e StopWordsRemover [8]. O cálculo do *tfidf* é feito em duas partes, primeiro usamos o modelo CountVectorizer[9] para calcular o freqüência de palavras, i.e., *tf*, e depois o modelo IDF[10] para completar o cálculo do *tfidf*.

Por razões de escalabilidade, o Apache Spark permite manipular vetores esparsos de forma distribuída. Na figura 10 podemos ver o DataFrame gerado com os valores *tfidf* para cada *tweet*, de acordo com as palavras que o compõem. A coluna tfidf armazena os vetores esparsos com esses valores calculados.

```
+-------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|cluster|                                                                                                                                                              tfidf|
+-------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|     20|(870,[4,6,22,48,78,95,96,147,312,331,597,618,679],[3.9201895680945396,2.597672113452404,3.264151046930188,7.6213895065965165,3.8106947532...|
|      6|(870,[0,6,13,26,126,388,408],[0.2950870204583586,2.597672113452404,2.7428541232969024,3.2128577525426376,4.129148484416793,4.822295664976...|
|     17|(870,[66,157,174,460,730,842],[3.8106947532982582,4.2626798770413155,4.2626798770413155,5.109977737428519,5.109977737428519,5.10997773742...|
|      1|                                                                                                (870,[8,122,839],[2.379948629607534,4.011365448760409,5.109977737428519])|
|     21|(870,[0,1,9,20,87,184,290],[0.2950870204583586,1.7087803557663637,2.447389910403066,2.989714201228428,3.906004933102583,4.262679770413155...|
|      1|                                                                                                                                (870,[0],[0.2950870204583586])|
|      5|(870,[0,8,17,100,134,152,157,229,280,432],[0.2950870204583586,2.379948629607534,2.8412941961101548,3.906004933102583,4.262679770413155,4...|
|      1|                                                                                                (870,[14,50,210],[2.7120824646301487,3.7236833763086286,4.4168305568685735])|
|      1|                                                                                (870,[0,157,210,579],[0.2950870204583586,4.2626798770413155,4.4168305568685735,5.109977737428519])|
|      5|(870,[1,3,4,8,24,65,114,138,162,201,396,509,513],[1.7087803557663637,1.8647846042429448,1.9600947840472698,2.379948629607534,3.1640675883...|
+-------+--------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

Figura 10 – *tf-idf* dos textos por *tweet*.

Para obter a representação dos tópicos, agregamos os valores *tfidf* de todas as palavras ou termos usados nos *tweets* pertencentes ao mesmo cluster e ordenamos estes termos de acordo com os valores agregados (pode ser agregação por soma ou média). Podemos representar cada tópico escolhendo as primeiras 10 palavras do ranking, como mostrado na figura 11. Estas palavras ou termos recebem o nome de descritores.

3.1.5 Avaliação da representação de tópicos

Conforme mencionado na seção 2.4, para avaliar os tópicos identificados, usamos a medida de coerência. A medida de coerência de tópico é amplamente utilizada em NLP e avalia quão bem um tópico é apoiado por um conjunto de textos (chamado de corpus de referência). Ele usa estatísticas e probabilidades extraídas do corpus de referência,

[7] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.Tokenizer.html

[8] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StopWordsRemover.html

[9] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.CountVectorizer.html

[10] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.IDF.html

topic	topicWords
0	[fila, salvar, feito, números, ah, @cnnbrasil, importa, vacina!, 2022, opinião]
1	[pra, hora, dor, deus, ngm, vacina??, tomar, corpo, obrigado, tá]
2	[reação, pqp, tomei, deu, ainda, ontem, eh, parece, to, horas]
3	[faz, todo, pro, mundo, aí, risco, ficou, anti, quer, vírus]
4	[ir, tomar, +, mim, trabalho, mãe, 🙄, estado, cabeça, adianta]
5	[gripe, dia, saúde, quarta, efeito, tudo, ministério, nada, talvez, bom]
6	[bando, vírus, gosta, fortal, política, @prefeiturapmf, nojo, vacina,vcs, brasileira,desse, disso]
7	[causa, ter, vcs, milhão, meio, deixar, médico, brasileiros, 1, fome]
8	[, governo, corrupção, brasil, cartão, milhões, sigilo, lista, caso, maior]
9	[onde, informa, nesta, sexta-feira,, faço, contra, sarampo, completo, imbecil, #saude]
10	[relação, salva, 7, jair, ver, vi, federal, @felipeneto, vidas,, assassinato]
11	[máscara, criança, pode, crianças, morrer, bandido, vacina,, durante, filho, pessoas]
12	[n, vou, calendário, hoje, toda, kkkkk, mudou, sempre, posto, tomo]
13	[doses, 4, tomou, 3, merda, aqui, assim, duas, vez, junto]
14	[irmão, @thiagosgbr, @tiochato3, @stephanevw, quadro, não,, fala, vacinar, ódio, morreu]
15	[dose, reforço, covid, terceira, vacinação, covid,, tão, amo, obrigatoriedade, pior]
16	[@jairbolsonaro, mil, vc, presidente, vida, sobre, pessoas, então, vacina,, comprar]
17	[dando, apoia, precisa, @pedrodoria, vida., né?, ?, teste., tadinho, pandemia,]
18	[q, braço, direito, bolsonaro, além, cara, alguém, n, cloroquina, patético,]
19	[começo, mesma, sequer, puta, @delucca, @guerreiracomun1, indiana, dms, iriam, vacina!]

Figura 11 – Tópicos representados.

especialmente focalizando o contexto das palavras, para dar uma pontuação de coerência ao tópico. A avaliação da coerência dos tópicos é realizada no ambiente seqüencial e utilizando a biblioteca *gensim* [11].

[11] https://radimrehurek.com/gensim/models/coherencemodel.html

4 RESULTADOS

4.1 Experimentos Realizados

4.1.1 COVID19 *Tweets*

Este conjunto de dados está disponível no Kaggle[1]. Ele foi compilado através de uma coleta diária de *tweets* com a *hashtag* #covid19 em 2020, de 24 de julho a 30 de agosto. Tem $\sim$ 180k *tweets* em língua inglesa. Há vários campos de informação por *tweet* armazenados neste conjunto de dados (13 campos). Como parte da avaliação experimental da solução proposta, consideramos apenas os campos *created_at* e *text*. O campo *id* foi gerado seqüencialmente para cada *tweet* no conjunto de dados. O campo *author_id* foi ignorado por ser informação que não é utilizada atualmente nos experimentos.

Vale a pena mencionar que a coleta de dados está considerada dentro do *pipeline* de trabalho, como explicado na Seção 3.1. Entretanto, usamos este conjunto de dados para acelerar a disponibilidade de uma fonte de dados para a avaliação experimental, assim como para facilitar a reprodutibilidade dos resultados obtidos.

4.1.2 Resultados

Para comparar os métodos de extração de tópicos, precisamos de uma medida agregada que represente a qualidade de um método inteiro, em vez de tópicos individuais. A coerência média fornece uma forma rápida de sumarizar a qualidade dos métodos avaliados.

A Figura 12 mostra os valores agregados de coerência para cada método à medida que o número de tópicos varia. Esses valores indicam que ambos os métodos tendem a se estabilizar à medida que mais tópicos são tentados a serem extraídos, já que o desempenho do método não diminui para um número maior de tópicos. Também pode ser visto que a curva de valores obtida pela solução proposta baseada em *embeddings* está acima da obtida pelo método LDA, o que implica que a coerência dos tópicos obtidos pela proposta é ligeiramente superior. Vale destacar que são estratégias não supervisionadas e a medida de coerência apenas sugere que os tópicos podem estar melhor formados quando palavras que formam os tópicos possuem maior probabilidade de ocorrerem juntas nos documentos. Desta forma, além desta medida, é importante analisar visualmente e qualitativamente os tópicos formados.

Na tabela 1 podemos ilustrar os resultados obtidos pelos dois métodos avaliados. Definimos 50 tópicos porque com este número de tópicos obtivemos os valores mais estáveis de coerência em ambos os casos. Vale a pena mencionar que estamos apresentando apenas

[1] https://www.kaggle.com/datasets/gpreda/covid19-tweets

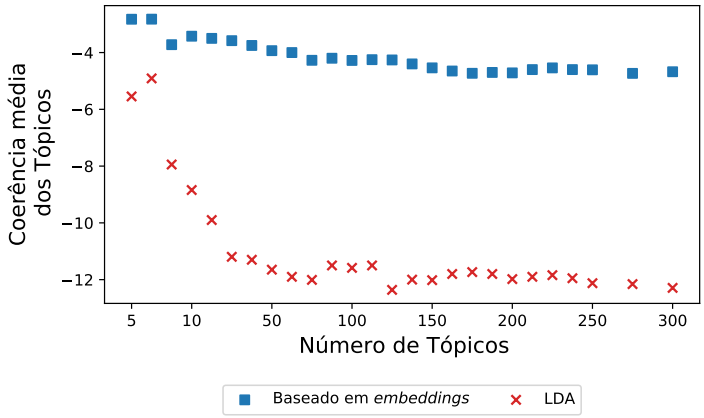

Figura 12 – Coerência média dos tópidos por cada método.

os cinco primeiros tópicos obtidos por cada método, ordenados pelos valores de coerência por tópico.

Tópico_#ID	Solução Proposta	LDA(Baseline)
Tópico_1	tested, recovery, smt, tests, speedy, congress, president, ji, good, health	covid, thank, later, avoid, chat, todayget, exempt, vaccineavoid, passesjoin, vaxxcovid
Tópico_2	covid, rw_christian, amp, breezergalway, dr_colemami, stevenwilson, heynursekat, marcucio_lab, ssz, rolo_tamasi	bc, please, missing, drug, often, company, million, cause, trial, mecfs
Tópico_3	mask, people, masks, wear, wearing, still, get, twitch, one, amp	positive, tests, gandhi, sonia, tested, congress, isolation, remain, months, president
Tópico_4	money, business, amp, tax, pay, like, many, still, get, people	people, mask, got, get, still, vaccinated, even, masks, many, know
Tópico_5	nfl, vikings, preseason, positive, game, tests, kirk, play, cousins, opener	week, go, away, next, back, lol, first, keep, booster, going

Tabela 1 – 10 palavras mais representativas por tópico identificadas de acordo com a **solução proposta** (esquerda) vs **LDA** (direita).

4.1.3 Análise temporal dos tópicos obtidos

Nesta seção, apresentamos a evolução temporal dos tópicos identificados pela proposta. Como mencionado na seção 3.1.3, os tópicos estão identificados com os grupos encontrados pelo algoritmo *KMeans* no espaço vetorial gerado pelos *text-embeddings* de cada *tweet*. Cada grupo pode ser representado pelos centroides calculados por *KMeans*. Desta forma, podemos dizer que cada tópico está relacionado ao centróide de seu respectivo grupo de *tweets*.

Com base nisso, é possível analisar a evolução temporal dos tópicos, definindo janelas de tempo para processamento em lote. Revisitando a Figura 6, este processamento

em lote pode ser realizado considerando as 4 primeiras etapas de nosso *pipeline* de trabalho. A janela de tempo é determinada no momento da coleta dos *tweets* com os parâmetros *start_time* e *end_time*, permitindo diferentes níveis de granularidade, e.g., minuto, hora, dia, semana, mês, ano.

Um tópico pode passar por três transições no tempo, com base no proposto em (OLIVEIRA; GAMA, 2010): (i) nascimento, (ii) permanência, ou (iii) morte. As transições dos tópicos, de um lote$_t$ para o próximo lote$_{t+1}$, são entendidas em função da distância Euclidiana entre os centróides de ambos os lotes e um parâmetro β, que determina o limite mínimo para considerar uma distância como uma transição.

Para fazer a análise temporal, dividimos o conjunto de dados COVID19 em 5 semanas ou lotes de *tweets*. Cada lote foi processado para identificar 5 tópicos. Na Figura 13 ilustramos os três tipos de transições experimentados pelos tópicos identificados pela abordagem proposta durante as 5 semanas. Podemos ver que da Semana 2 à Semana 5 é possível identificar tópicos nascendo e outros morrendo. Para simplificar a análise, consideramos que qualquer tópico que recebe uma conexão de um ou mais tópicos da semana anterior passou por uma transição de permanência.

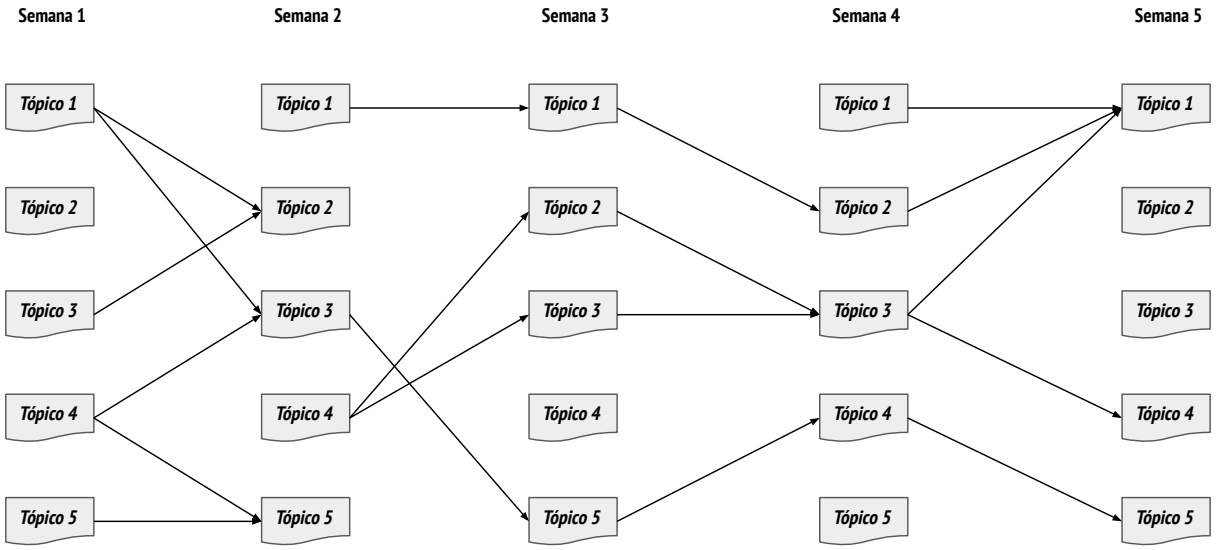

Figura 13 – Transições dos tópicos durante 5 semanas para $\beta = 0,8$.

5 CONCLUSÃO

Com base nos resultados apresentados no capítulo anterior, podemos responder às questões de pesquisa formuladas na seção 1.3:

Q1 *"Os tópicos identificados com base nesta proposta, em comparação com aqueles obtidos usando LDA, representam melhor o conteúdo daqueles tweets que os compõem?"*

R1 Sim, os tópicos identificados pela solução proposta apresentam melhores valores agregados de coerência; mesmo considerando um número suficientemente grande de tópicos. Com isso mostramos que o espaço vetorial gerado por *embeddings* usando modelos BERT pré-treinados representa melhor a semântica do corpus de documentos do que modelos clássicos como LDA.

Q2 *"Esta proposta é comparável em escalabilidade à versão disponível do LDA em Apache Spark?"*

R2 Sim, pois todas as etapas computacionalmente mais exigentes do nosso *pipeline* de trabalho (Figura 6) foram implementadas para execução em processamento distribuído. Todos os experimentos realizados para avaliar os dois métodos foram executados nas mesmas condições de hardware e usando o mesmo software, i.e., Apache Spark 3.

Finalmente, algum trabalho futuro a ser considerado poderia ser explorar um procedimento mais elaborado para pré-processar o conteúdo textual dos *tweets*, o que permitirá uma melhor representação dos tópicos identificados; considerando que as informações semânticas presentes nos *tweets* são acompanhadas de ruído, e.g., *hashtags*, URLs, emoticons. Também é possível explorar janelas de diferentes tamanhos ou granularidades para realizar uma análise temporal mais ou menos detalhada, dependendo da natureza do evento a ser estudado.

REFERÊNCIAS

AGGARWAL, C. C. **Machine learning for text**. [*S.l.: s.n.*]: Springer, 2018.

ANWAR, A. *et al.* Analyzing qanon on twitter in context of us elections 2020: Analysis of user messages and profiles using vader and bert topic modeling. *In*: **DG. O2021: The 22nd Annual International Conference on Digital Government Research**. [*S.l.: s.n.*], 2021. p. 82–88.

AZIZI, F. *et al.* Extracting major topics of covid-19 related tweets. **arXiv preprint arXiv:2110.01876**, 2021.

BAKAROV, A. A survey of word embeddings evaluation methods. **arXiv preprint arXiv:1801.09536**, 2018.

BLEI, D. M. Probabilistic topic models. **Communications of the ACM**, ACM New York, NY, USA, v. 55, n. 4, p. 77–84, 2012.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. **the Journal of machine Learning research**, JMLR. org, v. 3, p. 993–1022, 2003.

CAMPELLO, R. J.; MOULAVI, D.; SANDER, J. Density-based clustering based on hierarchical density estimates. *In*: SPRINGER. **Pacific-Asia conference on knowledge discovery and data mining**. [*S.l.: s.n.*], 2013. p. 160–172.

CASALINO, G. *et al.* A framework for intelligent twitter data analysis with non-negative matrix factorization. **International Journal of Web Information Systems**, Emerald Publishing Limited, 2018.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM New York, NY, USA, v. 51, n. 1, p. 107–113, 2008.

DEERWESTER, S. *et al.* Indexing by latent semantic analysis. **Journal of the American society for information science**, Wiley Online Library, v. 41, n. 6, p. 391–407, 1990.

DEVLIN, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

FÉVOTTE, C.; IDIER, J. Algorithms for nonnegative matrix factorization with the β-divergence. **Neural computation**, MIT Press, v. 23, n. 9, p. 2421–2456, 2011.

GHANI, N. A. *et al.* Social media big data analytics: A survey. **Computers in Human Behavior**, Elsevier, v. 101, p. 417–428, 2019.

GROOTENDORST, M. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics. **Version v0**, v. 4, 2020.

KOLAJO, T.; DARAMOLA, O.; ADEBIYI, A. Big data stream analysis: a systematic literature review. **Journal of Big Data**, Springer International Publishing, v. 6, p. 47, 2019.

LAHOTI, P.; GARIMELLA, K.; GIONIS, A. Joint non-negative matrix factorization for learning ideological leaning on twitter. *In*: **Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining**. [*S.l.: s.n.*], 2018. p. 351–359.

LUONG, M.-T.; PHAM, H.; MANNING, C. D. Effective approaches to attention-based neural machine translation. **arXiv preprint arXiv:1508.04025**, 2015.

MIKOLOV, T. *et al.* Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MIMNO, D. *et al.* Optimizing semantic coherence in topic models. *In*: **Proceedings of the 2011 conference on empirical methods in natural language processing**. [*S.l.: s.n.*], 2011. p. 262–272.

NEGARA, E. S.; TRIADI, D.; ANDRYANI, R. Topic modelling twitter data with latent dirichlet allocation method. *In*: IEEE. **2019 International Conference on Electrical Engineering and Computer Science (ICECOS)**. [*S.l.: s.n.*], 2019. p. 386–390.

NEWMAN, D. *et al.* Evaluating topic models for digital libraries. *In*: **Proceedings of the 10th annual joint conference on Digital libraries**. [*S.l.: s.n.*], 2010. p. 215–224.

NUGRAHANINGSIH, N.; LESTARI, A.; KAROLITA, D. Identifying the offensive words in the twitter using latent semantic analysis. *In*: IEEE. **2020 6th International Conference on Computing Engineering and Design (ICCED)**. [*S.l.: s.n.*], 2020. p. 1–4.

NUGROHO, R. *et al.* A survey of recent methods on deriving topics from twitter: algorithm to evaluation. **Knowledge and Information Systems**, Springer, v. 62, n. 7, p. 2485–2519, 2020.

OLIVEIRA, M.; GAMA, J. Understanding clusters evolution. *In*: **Workshop on Ubiquitous Data Mining**. [*S.l.: s.n.*], 2010. v. 500, n. 19, p. 16–20.

OSTROWSKI, D. A. Using latent dirichlet allocation for topic modelling in twitter. *In*: IEEE. **Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)**. [*S.l.: s.n.*], 2015. p. 493–497.

REIMERS, N.; GUREVYCH, I. Making monolingual sentence embeddings multilingual using knowledge distillation. *In*: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing**. Association for Computational Linguistics, 2020. Available at: https://arxiv.org/abs/2004.09813.

RÖDER, M.; BOTH, A.; HINNEBURG, A. Exploring the space of topic coherence measures. *In*: **Proceedings of the eighth ACM international conference on Web search and data mining**. [*S.l.: s.n.*], 2015. p. 399–408.

SAKAKI, T.; OKAZAKI, M.; MATSUO, Y. Earthquake shakes twitter users: real-time event detection by social sensors. *In*: **Proceedings of the 19th international conference on World wide web**. [*S.l.: s.n.*], 2010. p. 851–860.

SANKARANARAYANAN, J. *et al.* Twitterstand: news in tweets. *In*: **Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems**. [*S.l.: s.n.*], 2009. p. 42–51.

SANTILLI, S.; NOTA, L.; PILATO, G. The use of latent semantic analysis in the positive psychology: A comparison with twitter posts. *In*: IEEE. **2017 IEEE 11th International Conference on Semantic Computing (ICSC)**. [*S.l.: s.n.*], 2017. p. 494–498.

SILVA, N. F. da *et al.* Evaluating topic models in portuguese political comments about bills from brazil's chamber of deputies. *In*: SPRINGER. **Brazilian Conference on Intelligent Systems**. [*S.l.: s.n.*], 2021. p. 104–120.

SINGH, P. *et al.* Can twitter analytics predict election outcome? an insight from 2017 punjab assembly elections. **Government Information Quarterly**, Elsevier, v. 37, n. 2, p. 101444, 2020.

TEH, Y. W. **Dirichlet Process.** 2010.

VASWANI, A. *et al.* Attention is all you need. *In*: **Advances in neural information processing systems**. [*S.l.: s.n.*], 2017. p. 5998–6008.

VAYANSKY, I.; KUMAR, S. A. A review of topic modeling methods. **Information Systems**, Elsevier, v. 94, p. 101582, 2020.

WALLACH, H. M. Topic modeling: beyond bag-of-words. *In*: **Proceedings of the 23rd international conference on Machine learning**. [*S.l.: s.n.*], 2006. p. 977–984.

YANG, S.; ZHANG, H. Text mining of twitter data using a latent dirichlet allocation topic model and sentiment analysis. **International Journal of Computer and Information Engineering**, v. 12, n. 7, p. 525–529, 2018.

ZAHARIA, M. *et al.* Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *In*: **9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)**. [*S.l.: s.n.*], 2012. p. 15–28.

ZAHARIA, M. *et al.* Apache spark: a unified engine for big data processing. **Communications of the ACM**, ACM New York, NY, USA, v. 59, n. 11, p. 56–65, 2016.

ZIMBRA, D. *et al.* The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. **ACM Transactions on Management Information Systems (TMIS)**, ACM New York, NY, USA, v. 9, n. 2, p. 1–29, 2018.