CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

ESTUDO COMPARATIVO ENTRE ABORDAGENS NA REPRESENTAÇÃO TEXTUAL E ALGORITMOS USADOS NA CLASSIFICAÇÃO

Discente: Felipe Oliveira Feder

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do grau de mestre.

Orientador: Gustavo Paiva Guedes e Silva

Rio de Janeiro,

Dezembro de 2022

Estudo comparativo entre abordagens na representação textual e algoritmos usados na classificação

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Discente: Felipe Oliveira Feder

Aprovada por:

Presidente, Prof. Gustavo Paiva Guedes e Silva, D.Sc. (orientador)

Prof. Eduardo Bezerra da Silva, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Rio de Janeiro,

Dezembro de 2022

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

F293 Feder, Felipe Oliveira
Estudo comparativo entre abordagens na representação textual e algoritmos usados na classificação / Felipe Oliveira Feder. — 2022.
77f. : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.
Bibliografia : f. 68-77
Orientador: Gustavo Paiva Guedes e Silva

1. Processamento de linguagem natural (Computação). 2. 2. Mineração de dados. 3. Sistemas de computação. 4. Algoritmos computacionais. I. Silva, Gustavo Paiva Guedes e (Orient.). II. Título.

CDD 006.312

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

AGRADECIMENTOS

Agradeço aos que contribuíram, direta ou indiretamente, para a realização desta pesquisa. Um agradecimento especial ao meu pai (in memoriam), Luis Henrique de Alcântara Feder, a minha avó, Marlene de Alcântara Feder, e ao Professor Gustavo Paiva Guedes, meu orientador. Obrigado, Gustavo, por sua contribuição, dedicação e paciência.

RESUMO

Estudo comparativo entre abordagens na representação textual e algoritmos usados na classificação

Discente: Felipe Oliveira Feder

Orientador: Gustavo Paiva Guedes e Silva

Resumo da Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do grau de mestre.

Estamos vivendo uma revolução tecnológica sem precedentes nos últimos anos. A forma como nós nos relacionamos tem sido - e continuará sendo - impactada de diferentes maneiras. Acompanhando as evoluções em hardware e das tecnologias que nos possibilitam produzir e armazenar dados em volumes impensáveis, são observadas, também, evoluções algorítmicas e metodológicas que nos permitem avançar em busca de um mundo inteiramente novo, mesmo lidando com velhas questões tipicamente humanas. A fronteira do entendimento homem-máquina tem sido empurrada adiante constantemente. O processamento de linguagem natural é a ponte que liga a fala humana a possibilidades, antes inimagináveis, de uma máquina interpretá-la e processá-la devidamente. Os meios de representação textual vêm evoluindo consistentemente nas últimas décadas. O Bag-of-Words (BOW), atrelado ao uso de representações numéricas para palavras, vem sendo utilizado com sucesso na representação textual. No entanto, superando as deficiências do BOW, observamos o surgimento de representações numéricas complexas, geradas por redes neurais profundas, que são capazes de conservar as relações semânticas e sintáticas entre as palavras; os Word Embeddings (WE). A fronteira foi empurrada à frente; novas evoluções, novas aplicações, novos usos. O uso de Modelos de Linguagem Neural (MLN), com os WE, atingiu o estado da arte em diferentes tarefas no processamento de texto. Essa pesquisa compara esses dois métodos de representação de palavras, BOW e WE, e seus usos numa tarefa de classificação binária de polaridade. Foram montados dois grupos de classificadores e foram utilizados quatro conjuntos de dados. O primeiro grupo, formado por modelos n-gram, aqui chamados de Modelos de Aprendizagem de Máquina Tradicionais (MAMT), lidou com representações textuais que se serviram do BOW com TF-IDF e do BOW com LSA. O segundo grupo, formado por MLNs, que são modelos provenientes de redes neurais profundas que lidam com tarefas relacionadas ao processamento de texto, usou os WE e os WE Contextuais para representar os textos que seriam processados. Nos experimentos realizados foi observada a superioridade dos modelos de classificação semântica de texto diante dos modelos n-gram. Apesar disso, a escolha sobre qual técnica de representação textual (BOW ou WE) e tipo de modelo de linguagem usar (n-gram ou MLN) depende do contexto, já que os modelos n-gram, mesmo quando comparados as abordagens mais recentes, apresentam desempenho preditivo satisfatório e podem ser úteis em muitos contextos de uso.

Palavras-chave: Bag-of-Words, Word Embedding, Processamento de Linguagem Natural (PLN), Classificação de Texto, Mineração de Texto.

ABSTRACT

Estudo comparativo entre abordagens na representação textual e algoritmos usados na classificação

Discente: Felipe Oliveira Feder

Orientador: Gustavo Paiva Guedes e Silva

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

We are experiencing an unprecedented technological revolution in recent years. The way we relate to each other has been - and will continue to be - impacted in different ways. Following the evolutions in hardware and technologies that allow us to produce and store data in unthinkable volumes, algorithmic and methodological evolutions are also observed that allow us to advance in search of an entirely new world, even dealing with old typically human issues. The frontier of human-machine understanding has been constantly pushed forward. Natural language processing is the bridge that connects human speech to previously unimaginable possibilities for a machine to properly interpret and process it. The means of textual representation have been evolving consistently in recent decades. Bag-of-Words (BOW), linked to the use of numerical representations for words, has been successfully used in textual representation. However, overcoming the deficiencies of BOW, we observed the emergence of complex numerical representations, generated by deep neural networks, which are capable of preserving the semantic and syntactic relationships between words; the Word Embeddings (WE). The frontier was pushed forward; new evolutions, new applications, new uses. The use of Neural Language Models (MLN), with WE, has reached the state of the art in different tasks in text processing. This research compares these two word representation methods, BOW and WE, and their uses in a binary polarity classification task. Two groups of classifiers were set up and four data sets were used. The first group, formed by n-gram models, here called Traditional Machine Learning Models (MAMT), dealt with textual representations that used BOW with TF-IDF and BOW with LSA. The second group, formed by MLNs, which are models from deep neural networks that deal with tasks related to text processing, used the WE and the Contextual WE to represent the texts that would be processed. In the experiments carried out, the superiority of the semantic text classification models over the n-gram models was observed. Despite this, the choice of which textual representation technique (BOW or WE) and type of language model to use (n-gram or MLN) depends on the context, since n-gram models, even when compared to the most recent approaches, have satisfactory predictive performance and can be useful in many contexts of use.

Key-words: Bag-of-Words, Word Embedding, Natural Language Processing (NLP), Text Classification, Text Mining.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

BERT	Representações De Codificador Bidirecional De Transformadores (Bidirectional Encoder Representatios From Transformers, Em Inglês) 3, 4, 11, 15, 16, 24, 25, 26, 32, 35, 37, 42, 43, 44, 45, 47, 49, 50, 51, 52, 53, 54, 55, 61, 62, 63, 67
BI-LSTM	Memória Bidirecional De Curto E De Longo Prazo (Bidirectional Long Short Term Memory, Em Inglês) 4, 11, 16, 24, 42, 43, 44, 45, 46, 47, 50, 53, 54, 55, 58, 62, 63, 65, 67
BLEU	Bilingual Evaluation Understudy Score 36
BOW	Saco De Palavras (Bag-of-Words, Em Inglês) 2, 3, 4, 9, 10, 11, 12, 13, 15, 16, 31, 32, 33, 34, 35, 38, 40, 44, 50, 61, 64
BRNN	Redes Neurais Recorrentes Bidirecionais (Bidirectional Recurrent Neural Networks, Em Inglês) 23
CA	Computação Afetiva (Affective Computing, Em Inglês) 7, 8
CART	Classification And Regression Trees, Em Inglês 21
CWE	Incorporação Contextual De Palavras (Contextual Word Embeddings, Em Inglês) 3, 4, 11, 16, 25, 38, 40, 43, 45, 50, 51, 54, 55, 61, 62, 64
DM	Mineração De Dados (Data Mining, Em Inglês) 1, 6
ELMO	Incorporações Do Modelo De Linguagem (Embeddings From Language Model, Em Inglês) 3, 11, 15, 16, 35, 36
GRU	Unidade Recorrente Fechada (Gated Recurrent Unit, Em Inglês) 24
GS	Grid Search 28, 29, 42, 43, 45, 46, 59, 65
HGS	Halving Grid Search 29, 42, 43, 45, 46, 58, 59, 65
HMM	Modelo Oculto De Markov (Hidden Markov Model, Em Inglês) 24
KNN	K-vizinhos Mais Próximos (K-Nearest Neighbor, Em Inglês) 4, 16, 17, 18, 19, 42, 44, 45, 50, 55, 57, 59, 62
LSA	Analise Semântica Latente (Latent Semantic Analysis, Em Inglês) 3, 4, 10, 11, 12, 13, 38, 40, 43, 44, 50, 56, 57, 59, 61, 62, 64
LSI	Indexação Semântica Latente (Latent Semantic Indexing, Em Inglês) 3, 12, 13
LSTM	Memória De Curto E De Longo Prazo (Long Short Term Memory, Em Inglês) 11, 23, 24, 32, 34, 46
MAMT	Modelos De Aprendizagem De Máquina Tradicionais 4, 40, 42, 43, 44, 46, 49, 50, 51, 53, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67
MLM	Modelagem De Linguagem Mascarada (Masked-Language Modeling, Em Inglês) 25, 26
MLN	Modelos De Linguagem Neural 3, 4, 5, 15, 25, 31, 32, 40, 42, 43, 44, 45, 46, 47, 50, 51, 53, 55, 58, 59, 60, 61, 62, 63, 64, 65, 67
MNB	Multinomial Naive Bayes 4, 16, 42, 44, 45, 50, 55, 56, 57, 59, 62
NLP	Processamento De Linguagem Natural (Natural Language Processing, Em Inglês) 2, 3, 4, 8, 10, 15, 23, 24, 31, 34, 36, 37, 61
NSP	Previsão Da Próxima Frase (Next Sentence Prediction, Em Inglês) 25, 26
OM	Mineração De Opinião (Opinion Mining, Em Inglês) 2, 8
PLSI	Indexação Semântica Latente Probabilística (Probabilistic Latent Semantic Indexing, Em Inglês) 13
RF	Floresta Aleatória (Random Forest, Em Inglês) 4, 16, 21, 22, 42, 44, 45, 50, 57, 58, 59, 62, 64, 67
RI	Recuperação Da Informação (Information Retrieval, Em Inglês) 10
RNN	Redes Neurais Recorrentes (Recurrent Neural Networks, Em Inglês) 22, 23, 24, 33
SA	Análise De Sentimentos (Sentiment Analysis, Em Inglês) 2, 7, 8, 9
SST2	Stanford Sentiment Treebank V2 43, 45, 46, 47, 48, 49, 50, 55, 58, 62

SVD	Decomposição Em Valores Singulares (Singular Value Decomposition, Em Inglês) 13, 14
SVM	Máquina De Vetores De Suporte (Support Vector Machine, Em Inglês) 4, 16, 17, 42, 44, 45, 50, 51, 52, 53, 56, 57, 58, 59, 62, 64, 67
TA	Análise De Texto (Text Analysis, Em Inglês) 1, 6
TF	Frequência Do Termo (Term Frequency, Em Inglês) 3, 10, 12
TF-IDF	Frequência Do Termo - Frequência Inversa Do Documento (Term Frequency - Inverse Document Frequency, Em Inglês)3, 4, 10, 11, 12, 38, 40, 43, 44, 50, 53, 56, 57, 61, 62, 64
TM	Mineração De Texto (Text Mining, Em Inglês) 1, 6, 7, 8
WE	Incorporação De Palavras (Word Embedding, Em Inglês) 3, 4, 10, 11, 15, 16, 38, 40, 43, 45, 50, 54, 58, 61, 62, 64, 65, 66
YAR	Yelp Academic Review 34, 43, 45, 46, 47, 48, 49, 50, 51, 53, 55, 58, 59, 62, 65, 67

Capítulo I Introdução

Nós temos experimentado, nos últimos anos, transformações sem precedentes nas formas como nos relacionamos. A revolução tecnológica tem possibilitado interações das mais variadas, impactando as relações sociais de forma geral. Hoje, por meio dessa revolução, também é possível analisar essas interações com uma riqueza de detalhes impressionante. Com a incorporação de dispositivos computacionais ao nosso dia-a-dia, nos atendendo a finalidades diversas, o volume de dados circulante cresceu vertiginosamente.

A internet foi e tem sido um fator chave nessas transformações. A popularização de serviços orientados a opiniões de usuários, como redes sociais, blogs, sites de críticas e outros fomentou a utilização e expansão da mesma [89; 100]. Essa expansão da internet tem gerado um volume grande de dados textuais de usuários que expressam suas opiniões, sentimentos e experiências [100].

Em diferentes áreas e em razão de motivos diversos, a estatística tradicional é utilizada para obtenção de informações primárias - recolhidas para análise - por empresas, governos e outros entes sociais. Com o avanço tecnológico das últimas décadas, observamos que ao conhecimento associado a análise estatística, passamos a utilizar, também, o conhecimento oriundo da área de Mineração de Dados (Data Mining, em inglês) (DM). Com o DM podemos descobrir padrões interessantes e conhecimento a partir de grandes volumes de dados [45].

Somando-se a essas duas formas de extração de conhecimento - a análise estatística e a mineração de dados - nós demos um passo à frente com o surgimento da Mineração de Texto (Text Mining, em inglês) (TM). Por meio dela é possível realizar a extração de conhecimento de fontes de dados não-estruturados. O TM é baseado em muitas técnicas utilizadas no campo mais amplo da Análise de Texto (Text Analysis, em inglês) (TA), sendo a aplicação prática de muitas técnicas de processamento analítico do TA [81].

O tipo de dado mais fácil de ser criado, em qualquer cenário de aplicação, é o dado não estruturado [4]. Ainda segundo os autores, isso fomenta uma crescente necessidade do desenvolvimento de métodos e algoritmos que possam lidar com o mesmo. Diferentemente dos dados estruturados que, em geral, são manipulados com sistemas de banco de dados, o dado textual é tipicamente manipulado por meio de um mecanismo de pesquisa devido à falta de estruturas [4]. O problema da mineração de texto tem ganhado atenção crescente em razão do grande volume de dados textuais que são criados na web e em outras aplicações centradas em informações [4].

Alguns exemplos de dados não estruturados, presentes nas organizações, são mensagens de *email*, contratos e relatórios. Essas informações podem possuir vocabulário específico e a dimensão é variável. Além dos dados existentes em organizações, de forma geral, nós podemos criá-los e interagir com os mesmos em fóruns, *twittes* e *posts* na *web*. Assim, uma quantidade grande e variada de textos são escritos utilizando dispositivos computacionais, sendo armazenados em formato digital. Gradualmente, nos últimos anos, observamos o uso da computação em diferentes atividades profissionais, cobrindo uma gama cada vez maior de interações sociais.

A Análise de Sentimentos (Sentiment Analysis, em inglês) (SA), também chamada de Mineração de Opinião (Opinion Mining, em inglês) (OM), é o campo de estudo que analisa as opiniões, sentimentos, avaliações, atitudes e emoções das pessoas em relação às entidades e seus atributos expressos em texto escrito [70; 75]. As entidades podem representar produtos, serviços, organizações, individuos, eventos, questões ou tópicos [70; 75]. O SA estuda, principalmente, opiniões que expressam ou implicam sentimento positivo ou negativo [70]. As técnicas de SA podem ser divididas em três abordagens: as baseadas em aprendizagem de máquina, as baseadas em léxicos e as que se baseiam em uma abordagem híbrida [75].

O interesse em SA/OM pode ser rastreado desde [122], observando-se um aumento considerável de interesse nesse tópico por volta de 2001 [81]. Nós temos, como exemplo de fatores geradores desse aumento de interesse, a evolução dos métodos de Recuperação da Informação e de Aprendizagem de Máquina em Processamento de Linguagem Natural (Natural Language Processing, em inglês) (NLP), a crescente disponibilidade de conjuntos de dados para os algoritmos de Aprendizado de Máquina serem treinados - devido a popularização do *www* e, especificamente, o desenvolvimento de sites de agregação de avaliações -, além da realização de desafios intelectuais e aplicações de inteligência comercial que a área oferece [81; 88].

A possibilidade de lidar com dados não estruturados, com dados textuais, de consumidores, pacientes, usuários e outros interessados oferece a oportunidade do desenvolvimento de soluções em diferentes áreas, variando da previsão de crimes [5], saúde [9], previsões financeiras [42], comércio eletrônico [20], marketing [121] e segurança da informação [16].

Nas últimas décadas foram desenvolvidas soluções algorítmicas que refletem diferentes abordagens conceituais para lidar com os problemas dentro do campo de NLP. Como nós podemos representar palavras/documentos de modo a possibilitar que computadores possam executar tarefas com essas representações? Como possibilitar que uma máquina possa interpretar as palavras/documentos de uma forma mais próxima ao modo como nós, seres humanos, interpretamos e lidamos com elas nas nossas comunicações e uso diário? - essas são algumas das questões pensadas no campo.

Temos observado grandes evoluções nas formas de representação de palavras/documentos nas últimas décadas. No Saco de Palavras (Bag-of-Words, em inglês) (BOW), por exemplo, os docu-

mentos são representados por vetores cujas posições representam as palavras de um determinado vocabulário. O BOW tem grande utilidade na representação textual, podendo ser usado com indicações binárias na representação dos termos, com a Frequência do Termo (Term Frequency, em inglês) (TF), com a Frequência do Termo - Frequência Inversa do documento (Term Frequency - Inverse Document Frequency, em inglês) (TF-IDF), dentre outras medidas sugeridas em outros estudos. Mas além do problema relacionado a alta dimensionalidade dos vetores, as representações das palavras não guardam suas relações semânticas.

Com a Analise Semântica Latente (Latent Semantic Analysis, em inglês) (LSA) (dentre outros métodos) foi observada uma evolução. O LSA [52], às vezes usada como sinônimo do Indexação Semântica Latente (Latent Semantic Indexing, em inglês) (LSI), é um método do BOW para incorporar documentos num espaço vetorial. É uma técnica robusta, não supervisionada, para derivar uma representação implícita da semântica do texto com base na co-ocorrência observada de palavras [85]. Em [37] foi proposto o uso de LSA para sumarização genérica de notícias, como forma de identificar tópicos importantes em documentos sem o uso de recursos lexicais, por exemplo.

Mais recentemente observamos o surgimento da Incorporação de Palavras (Word embedding, em inglês) (WE). Aprender boas representações de palavras WE é um desafio no NLP, sendo as redes neurais atualmente usadas para essa tarefa chamadas de Modelos de Linguagem Neural (MLN) [8; 86]. Com os modelos baseados em algoritmos de redes neurais artificiais profundas, observamos desempenhos no estado da arte no processamento de linguagem natural [67; 99; 92; 119; 27].

Os MLN criam representações de palavras/frases que servem a análise de texto, tratando-se de representações vetoriais de valores reais que codificam os significados das mesmas, possibilitando que as palavras/frases que apresentem afinidades semânticas possuam essa condição codificada em seus vetores numéricos, estando mais próximas no espaço vetorial. Além de absorver características semânticas e sintáticas, essa abordagem reduz o impacto da dimensionalidade observado no modelo BOW. Os MLN podem produzir embeddings livres de contexto, tal como o *Word2Vec* [80; 79], ou dependentes do contexto, apresentados por modelos como o Incorporações do Modelo de Linguagem (Embeddings from Language Model, em inglês) (ELMo) [92] e Representações de Codificador Bidirecional de Transformadores (Bidirectional Encoder Representatios From Transformers, em inglês) (BERT) [27]. A Incorporação Contextual de Palavras (Contextual Word Embeddings, em inglês) (CWE) resolveu a maior limitação do WE livre de contexto: a polissemia, uma palavra com significados diferentes representada por apenas um vetor. Seus *embeddings* contextualizados são resultado de um processo de análise das frases inteiras, possibilitando a aprendizagem de diferentes *embeddings* para palavras polissêmicas.

Os MLN têm muitas vantagens sobre os modelos de linguagem *n-gram* [60]. Em comparação com os modelos de linguagem *n-gram*, os MLN podem lidar com segmentos muito mais longos,

podem generalizar melhor sobre contextos de palavras semelhantes e são mais precisos na previsão de palavras [60]. Por outro lado, os MLN são muito mais complexos, são mais lentos e precisam de mais energia para treinar e são menos interpretáveis do que os modelos *n-gram*, então para muitas tarefas (especialmente menores) um modelo de linguagem *n-gram* ainda é a ferramenta certa [60].

De forma geral, a escolha da forma de representação textual e o algoritmo de classificação mais adequados dependem do contexto; tipo de problema a ser tratado, disponibilidade e tipo de dados, e recursos computacionais disponíveis. No entanto, considerando que as formas de representação textual e algoritmos mais recentes apresentam desempenhos no estado da arte no NLP, em que medida os WE e CWE, usados nos MLNs (Bi-LSTM e BERT), superam os resultados dos modelos *n-gram* (BOW com TF-IDF ou com LSA), usados nos K-vizinhos mais próximos (K-Nearest Neighbor, em inglês) (KNN), Máquina de vetores de suporte (Support Vector Machine, em inglês) (SVM), Multinomial Naive Bayes (MNB) e Floresta aleatória (Random forest, em inglês) (RF), em diferentes conjuntos e com diferentes volumes de dados?

Essa pesquisa tem como objetivo a comparação de diferentes abordagens (no tocante a forma de representação de textos e uso de algoritmos) na classificação da polaridade de textos. Foi adotada uma instância de solução que utiliza dois grupos de modelos numa tarefa de classificação de polaridade binária com quatro conjuntos de dados. De um lado, temos modelos de linguagem *n-gram*, sendo esse grupo chamado, neste trabalho, de Modelos de Aprendizagem de Máquina Tradicionais (MAMT). Esses modelos irão lidar com os exemplos textuais representados por meio do BOW com TF-IDF e, também, com LSA. Fazem parte desse grupo modelos criados por meio dos algoritmos KNN, SVM, MNB e RF. Do outro, o grupo formado por MLN, baseados em algoritmos de redes neurais artificiais profundas, mais recentes, como a Memória Bidirecional de Curto e de Longo Prazo (Bidirectional Long Short Term Memory, em inglês) (Bi-LSTM) e o BERT. Esses últimos irão lidar com os exemplos textuais representados por meio dos WE e os CWE, respectivamente.

Estudos que visem comparar o uso de WE e CWE nos MLN com o BOW com TF-IDF ou LSA, em modelos *n-gram* (grupo MAMT neste estudo), podem contribuir ao desenvolvimento de outros estudos no NLP. Eventualmente o estudo proposto pode contribuir para melhores escolhas/percepções ao lidar com problemas no campo, tanto na área acadêmica, como em aplicações comerciais. Os experimentos realizados utilizaram quatro conjuntos de dados com diferentes tamanhos e fontes.

Após a realização dos experimentos, usando os 7 algoritmos dos dois grupos (MAMT e MLN) e quatro conjuntos de dados, foi possível observar: a) A superioridade dos modelos de classificação semântica de texto (MLN) diante dos modelos *n-gram* (MAMT); b) O benefício de uso de *embeddings* pré-treinados; c) A necessidade de bons recursos computacionais para treinar os modelos, variando de acordo com a abordagem utilizada (*embeddings* sem pré-treinamento, modelos *n-gram*

sem restrição vocabular, especificidades dos algoritmos); d) O desempenho satisfatório dos modelos *n-gram*; e) A dificuldade de comparar os modelos em razão das especificidades dos algoritmos e das técnicas de representação textual, e das variadas possibilidades existentes no uso dos mesmos; f) A determinação sobre qual abordagem utilizar não é simples. A tomada de decisão precisa considerar, em cada caso, os dados (características e volume), os recursos disponíveis, características dos algoritmos, bases de conhecimento, dentre outros. De forma geral, a superioridade dos modelos de classificação semântica de texto, os MLN, é evidente, mas os modelos *n-gram* podem (e devem, a depender do contexto) ser utilizados em muitas tarefas, especialmente menores, já que apresentam desempenho preditivo satisfatório - mesmo comparados as abordagens mais recentes.

Além dessa introdução, o trabalho está dividido em mais cinco capítulos. O segundo capítulo apresenta o arcabouço teórico relacionado a pesquisa. No terceiro capítulo nós temos os trabalhos relacionados ao tema. No quarto capítulo está descrita a abordagem proposta para a pesquisa. No quinto capítulo está descrita a análise experimental. No sexto capítulo estão descritas as considerações finais e, por último, temos as referências deste trabalho.

Capítulo II Fundamentação Teórica

Nesse capítulo é apresentado o arcabouço teórico necessário ao entendimento desta dissertação; seus conceitos básicos fundamentais e recursos tecnológicos utilizados para o desenvolvimento do estudo.

II.1 Mineração de Dados e Mineração de Texto

As metodologias de DM - de forma mais ampla em relação ao TM - são aplicadas em dados estruturados, tendo natureza numérica, repetitiva e quantificável [82]. No DM são utilizados recursos computacionais objetivando a obtenção de informações úteis e padrões ocultos em grandes quantidades de dados estruturados [82].

O desenvolvimento tecnológico do hardware, da web, o advento das redes sociais, a pluralidade de aplicações web, são fatores que possibilitaram o surgimento de grandes repositórios de diferentes tipos de dados. Toda essa tecnologia que observamos atualmente na web estimula a criação de grandes quantidades de dados textuais por diferentes usuários. Em razão do crescimento do volume de dados textuais disponíveis foi criada uma necessidade de aperfeiçoamento algorítmico para que os mesmos pudessem aprender padrões interessantes a partir dos dados de uma forma dinâmica e escalável [4].

O TM é baseada em muitas técnicas utilizadas no campo mais amplo do TA, sendo a aplicação prática de muitas técnicas de processamento analítico do TA [81]. Ela está presente nos diferentes campos de pesquisa existentes no TA - Recuperação da Informação, Extração da Informação, Extração de Conceito, Processamento de Linguagem Natural, Clusterização, Classificação e Mineração Web - baseando-se nas contribuições/conhecimentos de muitas disciplinas externas, tais como a Estatística, Aprendizagem de Máquina, Ciência de Gestão, Inteligência Artificial e Ciência da Computação [81].

O TM faz uso de recursos computacionais para buscar informações úteis em padrões válidos e compreensíveis, atuando em dados não estruturados, apresentados como documentos textuais [31]. Dados textuais tipicamente são manejados por meio de motores de busca em razão da falta de estrutura [4]. De modo contrário, dados estruturados, em geral, são manejados com sistemas de bancos de dados [4]. Um motor de busca permite que o usuário encontre informações úteis numa coleção

de documentos utilizando palavras-chave por meio de uma consulta [4]. Pesquisas que permitam melhorar os motores de busca são tópicos centrais no campo de Recuperação da Informação [4]. Contudo, as pesquisas no referido campo, em geral, focam mais no acesso a informação a análise da informação para a descoberta de padrões, sendo esse último o objetivo primário do TM [4].

O objetivo do acesso a informação é conectar a informação correta ao usuário correto, no tempo certo, com menos ênfase no processamento ou transformação da informação textual [4]. O TM pode ser considerado como indo além do acesso às informações para auxiliar ainda mais os usuários a analisar e digerir informações e facilitar a tomada de decisões [4].

No TM os padrões são extraídos por meio da utilização de algoritmos de aprendizagem de máquina e estatística [103]. O processo pode ser dividido em três etapas; pré-processamento, núcleo da mineração de texto e apresentação [30]. Na etapa de pré-processamento os documentos são processados com o objetivo de prepará-los para a manipulação computacional. Em muitas situações, ocorre, nessa etapa, uma análise linguística com o objetivo de encontrar padrões e relacionamentos [32] - pensada para facilitar o trato computacional que ocorrerá na etapa seguinte. Na etapa-núcleo do processo, que é o núcleo da mineração de texto, são utilizados algoritmos computacionais com o objetivo de extrair padrões que devem ser capazes de trazer informações novas sobre os dados, gerando uma compreensão maior sobre os mesmos e úteis a tomada de decisão [32]. Na etapa de apresentação o resultado desse processamento computacional é convertido em linguagem humana e natural [32].

Podemos observar estudos utilizando técnicas computacionais baseadas em mineração de texto em diferentes aplicações no contexto governamental, científico e empresarial. De acordo com a abordagem e enfoque, podemos ter classificações em diferentes categorias. Como exemplos dessas categorias, nós temos Inteligência Competitiva (Enterprise Business Intelligence) [13; 120], Descoberta eletrônica (E-Discovery) [57; 113; 124], Segurança Nacional ou Inteligência na gestão pública [47], Análise de Sentimentos [83], Processamento de Linguagem Natural [94; 84], dentre outras.

II.2 Computação Afetiva e Análise de Sentimentos

O SA é uma área de pesquisa no campo da Extração de Conceito [81]. A SA é o campo de estudo que analisa as opiniões, sentimentos, avaliações, atitudes e emoções das pessoas em relação às entidades e seus atributos expressos em texto escrito [75; 70]. As entidades podem representar produtos, serviços, organizações, individuos, eventos, questões ou tópicos [75; 70].

A Computação Afetiva (Affective Computing, em inglês) (CA) e o SA são duas áreas de pesquisa científica que buscam, através de métodos computacionais, lidar e interpretar as emoções e sentimentos humanos. Normalmente é utilizada a expressão 'Análise de Sentimentos' quando lidamos com a identificação apenas da polaridade (positivo, negativo e neutro). Em geral, se usa a expressão

'Computação Afetiva' para a identificação de diferentes emoções [19].

Um dos trabalhos pioneiros da CA é o de Manfred Clynes [93]; foi construído um equipamento capaz de medir emoções básicas do ser humano. A partir da publicação do livro 'Affective Computing' de Rosalind Picard em 1995, esse campo de conhecimento se desenvolveu [63]. Segundo [93] a emoção exerce papel fundamental na percepção, memória e atenção, na tomada de decisões e na interação homem-máquina. A aplicação do racional desenvolvido nessa área permite a identificação de opiniões e a classificação da orientação ou polaridade dessas, permitindo uma observação agregada e resumida dessas opiniões nos resultados [11]

O SA é também chamado de OM. Ele é um campo de pesquisa na área de TM e tem como objeto de estudo as opiniões, sentimentos e subjetividades em textos [75]. Há pesquisadores que apontam leves diferenças conceituais entre o SA e o OM [115]. Alguns pesquisadores indicam que o OM é originário da área de pesquisa de Recuperação da Informação - e que o OM extrai e analisa a opinião das pessoas sobre uma entidade (podem ser indivíduos, tópicos ou eventos) - ao passo que o SA foi originada da área de pesquisa de NLP e possui o objetivo de identificar o sentimento que é expressado em um texto e analisá-lo [75; 115]. O objetivo do SA, portanto, é encontrar opiniões, identificar os sentimentos que essas opiniões expressam e, no passo seguinte, classificar sua polaridade [75]. Apesar disso, esses dois problemas são essencialmente semelhantes [115]. Com a aplicação do conhecimento da área de SA, podem ser obtidos dados relacionados ao sentimento de usuários sobre um determinado assunto.

Não existe consenso no campo sobre uma forma padrão para medir sentimento. Uma forma usual e recorrente é pela polaridade de sentimentos [2]. Ela pode ser atribuída de forma discreta, com as categorias 'positiva', 'neutra' ou 'negativa', ou corresponder a uma pontuação em uma escala que representa a avaliação numa dessas três categorias. Assim, de forma discreta, as palavras associadas a elogios, relatos de bons acontecimentos e comparações positivas são 'positivas'. Críticas, relatos de experiências ruins, comparações negativas são 'negativas'. Algumas abordagens discretas lidam com essa classificação de forma binária, sendo 'positiva' ou 'negativa' [95], e outras de forma ternária, sendo 'positiva', 'negativa' ou 'neutra' [2].

O SA atua em três níveis contextuais no processo de classificação: nível de documento, nível de sentença e nível de aspecto [75]. À nível de documento, o mesmo é tratado como uma unidade básica e atômica de informação. O objetivo é classificá-lo considerando se o mesmo expressa um sentimento ou opinião positiva ou negativa (em se tratando de classificação binária) [75]. À nível de sentenças temos uma determinação se cada sentença expressa um sentimento ou opinião positiva ou negativa [75]. Assim como os documentos, nesse caso cada sentença é considerada - individualmente - uma unidade básica e atômica da informação. À nível de aspectos o objetivo é identificar sentimentos a respeito de aspectos de entidades identificadas nos textos [75].

Uma entidade pode possuir um ou mais aspectos a serem considerados na análise e esses aspectos podem ter sentimentos positivos ou negativos. Àqueles que manifestam suas opiniões podem dar diferentes opiniões para diferentes aspectos da mesma entidade [75]. Como exemplo, nesse último caso, teríamos uma sentença como "Ela é interessante, mas não se veste direito!" ou "O computador é bonito, mas não funciona direito" ou, ainda, "O flamengo tem um bom ataque, mas a defesa é ruim!". É importante destacar que, segundo [69], não há diferença fundamental entre a classificação à nível de documento e sentença porque as sentenças são consideradas documentos curtos.

O SA possui três abordagens: a abordagem baseada em léxicos, a abordagem com aprendizado de máquina e a abordagem híbrida [75]. Na abordagem baseada em aprendizagem de máquina os algoritmos utilizam as características sintáticas e linguísticas aprendidas com textos para inferir os sentimentos ao analisar outros textos [75]. O aprendizado pode ser supervisionado, caso em que é preciso utilizar um conjunto de dados textuais rotulados para treinar o classificador, ou pode ser não supervisionado, caso em que os dados textuais não estão rotulados.

Na abordagem baseada em léxicos nós temos a baseada em dicionários - caso em que é utilizado um vocabulário bem definido e manualmente anotado, com orientações bem definidas - e a baseada em corpus, em que o próprio corpus é utilizado para encontrar palavras que expressam opiniões com orientações específicas de contexto. Nesse último caso, a abordagem se baseia em padrões sintáticos e na probabilidade de ocorrência de uma palavra em um conjunto positivo ou negativo de palavras, realizando uma pesquisa numa quantidade muito grande de textos em mecanismos de pesquisa [71; 75]. Na abordagem híbrida temos a utilização de algoritmos baseados em aprendizagem de máquina e o uso de léxicos [75].

II.3 A Hipótese Distribucional no Processamento de Linguagem Natural

A hipótese distribucional foi apresentada por [46] e se baseia no princípio de que as partes de uma linguagem não se relacionam de forma arbitrária. Segundo [102], a ideia geral por trás da hipótese distribucional é clara: existe uma correlação entre semelhança distribucional e a similaridade dos significados das palavras, permitindo que possamos utilizar a primeira para estimar a segunda.

A hipótese distribucional considera que o significado de uma palavra é o seu uso na linguagem, e o uso de uma palavra também está relacionado ao contexto em que a mesma está inserida [123]. Segundo [29], dentre as muitas definições sobre o que seria o "contexto", a mais simples seria a de que um contexto seria apenas um conjunto de palavras que ocorreram nas proximidades da palavra alvo. É muito comum que os documentos sejam representados como BOW, apesar de existirem outras formas de representar um documento de texto. Nesse tipo de representação cada documento é considerado como um conjunto de palavras que são independentes e sem ordem, em que cada documento é um vetor de valores não negativos em cada dimensão [55]. Cada dimensão/palavra do

documento é representado de acordo com a sua frequência, significando que os termos que aparecem com mais frequência são mais importantes e descritivos para o documento [55].

No entanto, nessa representação, a principal desvantagem é a perda de informação semântica, justamente em razão da consideração de que os termos são independentes e sem ordem [12]. Até há poucos anos atrás as áreas de NLP e Recuperação da Informação (Information Retrieval, em inglês) (RI) ainda tratavam as palavras apenas como unidades atômicas. Para mitigar a perda de informação semântica o campo de pesquisa de NLP buscou soluções que levassem em conta a semântica e o contexto de cada elemento da linguagem.

Observamos o surgimento do *Bag of* n-grams [66], que considera combinações de até n palavras para obter alguma informação semântica dos documentos e, havendo ainda a deficiência da representação BOW e *Bag of* n-grams em relação a semântica, foram propostos outros modelos para representações mais complexas em um espaço vetorial que agrupa palavras semelhantes, alcançando melhor desempenho preditivo em tarefas do NLP [79]. Eles possuem grande poder de generalização e são conhecidos como WE [18]. É comum, no estado da arte da área de NLP, o uso do WE gerados por redes neurais [18].

II.4 Métodos de Representação de Palavras e Documentos

Dados textuais podem ser representados em diferentes níveis e de formas variadas. Como exemplo, os mesmos podem ser facilmente tratados como BOW ou podem ser tratados como uma sequência de palavras [3]. Nesta seção temos uma breve introdução de três métodos de representação de palavras/documentos que vêm sendo utilizadas nas últimas décadas para lidar com tarefas ligadas ao processamento textual em diferentes áreas de pesquisa, sendo os mesmos explicados em suas respectivas subseções abaixo.

Uma forma bastante utilizada para lidar com o dado textual é representá-lo por meio do BOW. Os documentos são representados como sacos de palavras e as mesmas podem ser representadas de forma binária, indicando a presença ou não do termo no documento, podem receber um valor numérico obtido por meio do cálculo de sua frequência, no caso do TF, ou por meio da frequência do termo no documento e, ao mesmo tempo, considerando, também, a presença do termo em todos os documentos do corpus, no caso do TF-IDF. Além da possibilidade, claro, da utilização de variações e outras medidas. Nesse caso, cada documento é representado por meio de um vetor esparso, cujo tamanho estará diretamente ligado ao tamanho do vocabulário.

Com o LSA observamos um aperfeiçoamento na representação textual. Com ele o documento pode ser representado por meio de um vetor denso contendo os conceitos semânticos identificados no mesmo [24]. Passamos, assim, para uma representação em que cada dimensão corresponde a um conceito ou tópico, em vez de utilizarmos uma dimensão para cada palavra representada por meio

de alguma medida. Passamos de uma alta dimensionalidade para um espaço semântico de menor dimensão. Se, por exemplo, temos um documento sendo representado por meio de um vetor esparso com dimensão 90.000, após o LSA poderemos ter um vetor denso de dimensão 200.

Uma evolução significativa foi observada com o WE. Os WE são representações vetoriais densas obtidas por meio de redes neurais. São representações numéricas de palavras que incorporam características semânticas, fazendo com que palavras que apresentem afinidades semânticas possuam essa condição codificada em seus vetores numéricos. Nesse caso estamos representando o termo, não o conjunto de termos como no BOW e nem o conjunto de tópicos identificados como no LSA.

Uma evolução do WE foi observada com CWE. Trata-se de uma representação vetorial complexa obtida por meio de redes neurais profundas, tendo superado a maior limitação da abordagem clássica do WE: a polissemia, quando temos uma mesma palavra com significados diferentes. Um dos mais populares é o ELMo, que não usa *embeddings* fixos, utiliza uma Bi-LSTM, analisa a sentença inteira e atribui incorporações de palavras às palavras.

Antes da proposição do ELMo (ocorrida em 2018), em 2017 foi publicado um estudo propondo uma nova arquitetura de rede simples, baseada exclusivamente em Mecanismos de Atenção, dispensando recorrências e convoluções inteiramente. Foi demonstrado que modelos de sequência (como o Memória de Curto e de Longo Prazo (Long Short Term Memory, em inglês) (LSTM)) podem ser totalmente substituídos por Mecanismos de Atenção, obtendo até mesmo melhores desempenhos preditivos. Em 2018 foi publicado estudo onde foi proposto o BERT que combina a incorporação de contexto observada no ELMo e vários 'Transformer', além de ser bidirecional (sendo uma novidade para os 'Transformers'). A representação vetorial que o BERT atribui a uma palavra é uma função de toda a frase e, assim, uma mesma palavra pode ter diferentes vetores com base nos contextos em que é observada.

Os CWE e os WE se apoiam na hipótese distribucional, entendendo que palavras que ocorrem em contextos similares tendem a ter significados similares [72; 117]. O LSA, ao extrair tópicos de documentos, também assume que palavras com significados similares ocorrerão em contextos similares. Já o BOW, quando utilizado com medidas como o TF-IDF, não considera a ordem dos termos como relevantes na constituição semântica do documento.

II.4.1 Bag of Words - BOW

O BOW é um modelo básico de representação textual, onde a estrutura linguística dentro do texto é ignorada, sendo a maneira mais comum de modelar documentos [54]. Os documentos são transformados em vetores numéricos esparsos, sendo manipulados por meio de operações algébricas lineares [54]. Essa representação equilibra a eficiência computacional com a necessidade de reter o conteúdo do documento [24]. As posições no vetor representam as palavras de um determinado vo-

cabulário naquele documento, registrando seu peso obtido por meio de alguma medida estabelecida ou mesmo registrando sua ausência.

O BOW tem grande utilidade na representação de documentos, podendo ser usado com indicações binárias na representação dos termos, com o TF, com o TF-IDF, dentre outras medidas indicadas em estudos no campo. O BOW, embora perca a informação de posicionamento nas palavras, geralmente é muito mais simples de lidar do ponto de vista algorítmico quando comparado a abordagem baseada em sequência de palavras [3].

No modelo BOW, uma palavra é representada como uma variável separada com peso numérico de importância variável [54]. O esquema de ponderação mais popular é o TF-IDF:

$$tfidf(w) = tf(w) * log\frac{N}{df(w)} \tag{II.1}$$

Em II.1 tf(w) é a frequência do termo no documento (o número de vezes que aquele termo ocorreu no documento); df(w) é a frequência de documento (o número de documentos que contém o termo); N é o número de documentos no corpus; tfidf(w) é o peso relativo do termo no vetor [54].

Usando o BOW, o corpus pode ser representado como uma matriz documentos * palavras [54]. Quando transformamos com sucesso o corpus de texto em vetores numéricos, podemos analisá-los com técnicas de matemática aplicada e aplicar os métodos existentes de aprendizado de máquina ou mineração de dados, como classificação ou agrupamento [54; 24]. Por meio da técnica de redução de dimensionalidade podemos identificar uma representação em menor dimensão de um conjunto de vetores que preserva propriedades importantes [24]. O BOW, comumente usado na análise de texto, pode ser analisado de forma muito eficiente e retém uma grande quantidade de informações úteis, mas também é problemático porque o mesmo pensamento pode ser expresso usando muitos termos diferentes ou um termo pode ter significados muito diferentes [24]. Além das representações vetoriais não guardarem as relações semânticas entre as palavras, seus vetores costumam ter alta dimensionalidade, dado que a representação vetorial faz referência ao documento, ao conjunto de palavras que o compõe, o que pode gerar dificuldades no trato computacional [24]. Abordagens utilizadas no pré-processamento do dado textual tentam mitigar esse problema da hiperdimensionalidade dos vetores.

II.4.2 Indexação Semântica Latente e Análise Semântica Latente

O LSA, às vezes usada como sinônimo do LSI, é um método do BOW para incorporar documentos e termos num espaço vetorial de baixa dimensão que, intencionalmente, representa os conceitos semânticos dos documentos [24]. Os vetores BOW têm uma dimensionalidade muito alta, onde cada dimensão corresponde a um termo da linguagem [24]. No entanto, para a tarefa de análise de conceitos presentes nos documentos, o ideal é um espaço semântico de menor dimensão, com

cada dimensão correspondendo a um conceito ou um tópico [24]. A redução de dimensão pode ser aplicada para encontrar o espaço semântico e sua relação com a representação BOW [24]. A nova representação no espaço semântico revela a estrutura tópica do corpus mais claramente do que a representação original [24].

É uma técnica robusta, não supervisionada, para derivar uma representação implícita da semântica do texto com base na co-ocorrência observada de palavras [85]. Essa representação identifica e é capaz de desambiguar termos com múltiplos significados, fornecendo uma representação de menor dimensão de documentos que reflita conceitos em vez de termos brutos [24]. Essa abordagem endereça alguns dos problemas observados no BOW [24]. No BOW as relações semânticas entre os termos não são preservadas e, nesse contexto, é indiferente para esse modelo de representação o fato de termos, usualmente, o mesmo pensamento sendo expresso por meio de muitos termos diferentes, em documentos diferentes, ou um único termo com significados muito diferentes em documentos diferentes.

Ao projetar documentos no espaço semântico, o LSA permite a análise de documentos em um nível conceitual, supostamente superando as desvantagens da análise puramente baseada em termos [24]. Por exemplo, na recuperação de informações, os usuários podem usar muitas consultas diferentes para descrever a mesma necessidade de informação e, da mesma forma, muitos dos documentos relevantes podem não conter os termos exatos usados na consulta específica [24]. Nesse caso, projetar documentos no espaço semântico permite que o mecanismo de busca encontre documentos que contenham os mesmos conceitos, mas termos diferentes [24]. A projeção também ajuda a resolver termos associados a vários conceitos [24]. Nesse sentido, o LSA supera as questões de sinonímia e polissemia que afligem a recuperação de informações baseada em termos [24]. A sinonímia pode ser capturada, pois termos com o mesmo significado geralmente ocorrem em contextos semelhantes [24]. A polissemia pode ser abordada, uma vez que termos com significados diferentes podem ser distinguidos por suas ocorrências em contextos diferentes [24].

O LSI foi aplicado a dados de texto na década de 1980 e posteriormente usado para indexação em sistemas de recuperação de informação [26; 24; 52]. Ao longo das últimas décadas a abordagem tem sido utilizada para uma variedade de tarefas. Ele é baseado na Decomposição em Valores Singulares (Singular Value Decomposition, em inglês) (SVD) da matriz termo-documentos, que constrói uma aproximação de baixo nível da matriz original, preservando a similaridade entre os documentos [24; 52]. O LSI destina-se a interpretar as dimensões da aproximação de baixo escalão como conceitos semânticos, embora seja superado a esse respeito por melhorias posteriores, como a Indexação Semântica Latente Probabilística (Probabilistic Latent Semantic Indexing, em inglês) (PLSI) [24; 52].

Em [37] foi proposto o uso do LSA para sumarização genérica de notícias, como forma de

identificar tópicos importantes em documentos sem o uso de recursos lexicais. A construção da representação do tópico é iniciada com o preenchimento de uma matriz A, n por m, onde cada linha corresponde a uma palavra de entrada (n palavras) e cada coluna corresponde a uma sentença na entrada (m sentenças) [85]. A entrada a_{i_j} da matriz, por exemplo, corresponde ao peso da palavra i na sentença j [85]. Se a frase não contém a palavra, o peso é zero, caso contrário o peso é igual ao peso $TF * IDF$ da palavra [85]. As técnicas padrões SVD da álgebra linear são aplicadas à matriz A para representá-la como o produto de três matrizes: $A = U\Sigma\mathrm{V}^T$ [85]. Toda matriz tem uma representação desse tipo [85].

A Matriz U é uma matriz n por m de números reais em que cada coluna pode ser interpretada como um tópico, ou seja, uma combinação específica de palavras de entrada com o peso de cada palavra no tópico dado pelo número real [85]. A matriz Σ é uma matriz diagonal m por m [85]. A única entrada na linha i da matriz corresponde ao peso do "tópico", que é a i-ésima coluna de U [85]. Tópicos com baixo peso podem ser ignorados, excluindo as últimas k linhas de U, as últimas k linhas e colunas de Σ e as últimas k linhas de V^T. Esse procedimento é chamado de redução de dimensionalidade [85]. Corresponde aos limiares empregados nas abordagens de centroide e palavras-tópico, e os tópicos com baixo peso são tratados como ruído [85]. Matriz V^T é uma nova representação das sentenças, uma sentença por linha, cada uma das quais é expressa não em termos de palavras que ocorrem na sentença, mas sim em termos de tópicos dados em U [85]. A matriz $D = \Sigma\mathrm{V}^T$ combina os pesos do tópico e a representação da sentença para indicar até que ponto a sentença transmite o tópico, com d_{ij} indicando o peso do tópico i na sentença j [85].

A proposta original de Gong e Liu era selecionar uma frase para cada um dos tópicos mais importantes [85]. Eles realizam redução de dimensionalidade, retendo apenas tantos tópicos quanto o número de sentenças que desejam incluir no resumo [85]. A frase com maior peso para cada um dos tópicos retidos é selecionada para formar o resumo [85]. Essa estratégia sofre da mesma desvantagem que a abordagem de cadeias lexicais, porque mais de uma frase pode ser necessária para transmitir todas as informações pertinentes a esse tópico [85].

Pesquisadores posteriores propuseram procedimentos alternativos que levaram a um melhor desempenho do resumidor na seleção de conteúdo [85]. Uma melhoria é usar o peso de cada tópico para determinar a proporção relativa do resumo que deve cobrir o tópico, permitindo assim um número variável de frases por tópico [85]. Outra melhoria foi perceber que muitas vezes frases que discutem vários dos tópicos importantes são boas candidatas para resumos [110; 85]. Para identificar tais sentenças, o peso da sentença si é dador por:

$$Weight(s_i) = \sqrt{\sum_{j=1}^{m} d_{i,j}^2} \tag{II.2}$$

Nas muitas variações do algoritmo a representação do tópico permanece a mesma, enquanto que a forma como as sentenças são pontuadas e escolhidas é variável [85]. Isso influencia diretamente no desempenho do resumidor ao selecionar conteúdos importantes [85]. A redução de dimensão começa com uma representação de recursos de documentos (normalmente um modelo BOW) e procura uma representação de dimensão inferior que seja fiel à representação original [24; 52]. Embora esse acoplamento próximo com os recursos originais resulte em uma representação mais coerente que mantém mais informações originais do que o agrupamento, a interpretação das dimensões compactadas ainda é difícil [24; 52]. Especificamente, cada nova dimensão é geralmente uma função de todos os recursos originais, de modo que geralmente um documento só pode ser totalmente compreendido considerando todas as dimensões juntas [24; 52].

II.4.3 Word Embeddings

O uso de vetores numéricos para representação de palavras tem uma longa história [77; 28; 50; 101]. Uma arquitetura de modelo muito popular para estimar o MLN foi proposta em [14], onde uma rede neural *feedforward* com uma camada de projeção linear e uma camada oculta não linear foi usada para aprender conjuntamente a representação vetorial de palavras e uma estatística modelo de linguagem [77]. Esse trabalho foi seguido por muitos outros [77]. Outra arquitetura interessante de MLN foi apresentada em [76; 78], onde os vetores de palavras são primeiro aprendidos usando uma rede neural com uma única camada oculta [77]. Os vetores de palavras são então usados para treinar o [77]. Assim, os vetores de palavras são aprendidos mesmo sem construir o MLN completo [77].

Em [77] foram propostas duas novas arquiteturas de modelo para computar representações vetoriais contínuas de palavras de grandes conjuntos de dados. A qualidade dessas representações foi medida em uma tarefa de similaridade palavras e os resultados foram comparados com as técnicas de melhor desempenho preditivo anteriormente baseadas em diferentes tipos de redes neurais. Foram observadas grandes melhorias na precisão e com um custo computacional muito menor, levando menos de um dia para aprender vetores de palavras de alta qualidade de um conjunto de dados de 1,6 bilhão de palavras. Essas representações vetoriais apresentavam desempenho de última geração para medição de semelhanças sintáticas e semânticas de palavras.

Nos anos seguintes observamos outras contribuições e o aperfeiçoamento dessas representações vetoriais por meios de novos algoritmos e abordagens. Além de [77; 79] podemos pensar no *FastText* [58], no ELMo [92], no 'Transformer' [119] e no BERT [27] como marcos de aperfeiçoamentos no processamento do dado textual, no campo do NLP.

Aprender boas representações de palavras WE é um desafio no NLP, sendo as redes neurais atualmente usadas para essa tarefa chamadas de MLN [8; 86]. Com os modelos baseados em

algoritmos de redes neurais artificiais profundas, observamos desempenhos preditivos no estado da arte no processamento de linguagem natural [67; 99; 92; 119; 27].

Nos últimos anos observamos a proposição do WE, livre de contexto, e o CWE. Naqueles, apesar da nítida evolução quando comparados, por exemplo, a abordagem do BOW, nós podemos observar duas grandes limitações; a polissemia e a homonímia. Palavras polissêmicas e palavras homônimas não são tratadas adequadamente.

Os *embeddings* incorporam características semânticas, fazendo com que palavras que apresentem afinidades semânticas possuam essa condição codificada em seus vetores numéricos e esses estejam próximas no espaço vetorial. Além de absorver características semânticas e sintáticas, essa abordagem reduz o impacto da dimensionalidade observado no BOW. No WE as palavras com múltiplos significados são combinadas em uma única representação, tendo um único vetor no espaço semântico. De forma diferente, o CWE é resultado de um processo de análise das frases inteiras, possibilitando a aprendizagem de diferentes *embeddings* para palavras polissêmicas. Temos representações vetoriais densas, assim como o WE, mas obtidas por meio de redes neurais profundas.

O ELMo [92] e o BERT [27] são populares e se tornaram referências quando falamos em CWE. Eles usam o contexto de uma palavra para desambiguar polissemias. O ELMo [92] usa arquitetura de rede neural Bi-LSTM para analisar a sentença inteira e atribuir um *embedding* para cada palavra, em cada contexto que for observada. O BERT [27] faz uso do 'Transformer' [119], um Mecanismo de Atenção que compreende as relações contextuais entre as palavras no texto.

O BERT [27] combina a incorporação de contexto do ELMo e vários 'Transformers', sendo bidirecional. O vetor que o BERT atribui a uma palavra é uma função de toda a frase, assim nós podemos ter diferentes vetores para a mesma palavra com base em diferentes contextos no qual a mesma foi observada.

II.5 Algoritmos

Nessa seção são apresentados os algoritmos de aprendizagem de máquina utilizados nesse trabalho. São apresentados o KNN, SVM, MNB, RF, Bi-LSTM e o BERT. Em cada subseção são fornecidas informações gerais, informações sobre o funcionamento e referências.

II.5.1 Support Vector Machine

Desenvolvida por Vapnik [118], o SVM, é uma técnica de aprendizado de máquina baseada na teoria do aprendizado estatístico. O objetivo é resolver um problema de classificação encontrando um hiperplano ótimo em um espaço vetorial de alta dimensionalidade [23]. Em outras palavras, o que um SVM faz é encontrar uma linha de separação entre dados de duas classes. Essa linha, chamada comumente de hiperplano entre os dados das classes, busca maximizar a distância entre

os pontos mais próximos em relação a cada uma das classes.

Ele usa um mapeamento não linear para transformar os dados de treinamento originais em uma dimensão superior [45]. Dentro dessa nova dimensão, ele procura o hiperplano de separação ideal linear [45]. Com um mapeamento não linear apropriado para uma dimensão suficientemente alta, os dados de duas classes podem sempre ser separados por um hiperplano [45]. O SVM encontra esse hiperplano usando vetores de suporte (tuplas de treinamento "essenciais") e margens (definidas pelos vetores de suporte) [45].

Os SVMs podem ser SVMs lineares ou SVMs não lineares, situação em que os modelos são utilizados, respectivamente, para classificar dados linearmente separáveis ou não. Os SVMs podem ser usados para predição numérica e para classificação, tendo sido aplicados em uma série de áreas [45].

Embora os tempos de treinamento nos SVMs mais rápidos possam ser lentos, os SVMs são altamente precisos devido a sua capacidade de modelar limites de decisão não lineares complexos [45]. A complexidade do classificador aprendido é caracterizada pelo número de vetores de suporte, não pela dimensionalidade dos dados [45]. Em razão disso, os SVMs tendem a ser menos propensos ao *overfitting* do que alguns outros métodos [45]. Os vetores de suporte encontrados também fornecem uma descrição compacta do modelo [45]. Eles são as tuplas de treinamento essenciais ou críticas - eles estão mais próximos do limite de decisão [45]. Se todas as outras tuplas de treinamento fossem removidas e o treinamento fosse repetido, o mesmo hiperplano de separação seria encontrado [45]. Além disso, o número de vetores de suporte encontrados pode ser usado para calcular um limite (superior) na taxa de erro esperada do classificador SVM, que é independente da dimensionalidade dos dados [45]. Um SVM com um pequeno número de vetores de suporte pode ter boa generalização, mesmo quando a dimensionalidade dos dados é alta [45].

O SVM é efetivo em espaços de alta dimensionalidade e em tarefas em que a dimensionalidade dos dados é maior que o número de exemplos disponíveis [33]. O autor ainda destaca que ele apresenta grande sucesso na tarefa de classificação de textos.

II.5.2 K-Nearest Neighbor

O KNN é o método dos ‘k’ vizinhos mais próximos. Foi descrito pela primeira vez no início dos anos 1950 [45]. Por exigir trabalho intensivo ao lidar com grandes conjuntos de treinamento, não ganhou popularidade até a década de 1960, quando o poder computacional aumentou [45]. Desde então tem sido amplamente utilizado na área de reconhecimento de padrões [45].

Esse classificador é baseado no aprendizado por analogia, ou seja, compara determinada tupla de teste com tuplas de treinamento semelhantes à ela [45]. As tuplas de treinamento são descritas por n atributos, com cada tupla representando um ponto num espaço n-dimensional [45]. Assim, todas as

tuplas de treinamento são armazenadas num espaço de padrão n-dimensional [45]. Dada uma tupla desconhecida, o classificador KNN pesquisa o espaço padrão para as k tuplas de treinamento que estão mais próximas da tupla desconhecida [45]. Essas k tuplas de treinamento são os k "vizinhos mais próximos"da tupla desconhecida [45]. É utilizada uma medida de distância para definir a "proximidade", como a distância euclidiana, por exemplo [45]. De forma geral, se o problema envolve dados numéricos, é realizada normalização dos valores de cada atributo antes do processamento, pois isso ajuda a evitar que atributos com intervalos inicialmente grandes (renda, por exemplo) superem os atributos com intervalos inicialmente menores (atributos binários, por exemplo) [45].

Na classificação do KNN, a tupla desconhecida é atribuída à classe mais comum entre seus k-vizinhos mais próximos [45]. Quando $k = 1$, a tupla desconhecida é atribuída à classe da tupla de treinamento que está mais próxima dela no espaço padrão [45]. Esse tipo de classificador também pode ser usado para predição númerica, ou seja, para retornar uma predição com valor real para uma determinada tupla desconhecida [45]. Nesse caso, o classificador retorna o valor médio dos rótulos de valor real associados aos k vizinhos mais próximos da tupla desconhecida [45].

Para lidar com atributos nominais ou categóricos, se os atributos das tuplas que estão sendo comparadas forem idênticos (por exemplo, as tuplas T1 e T2 são iguais a ‘flamengo’) então a diferença entre as duas é considerada como 0 [45]. Se as duas são diferentes (por exemplo, a tupla T1 é ‘flamengo’, mas a tupla T2 é ‘vasco’), então a diferença é considerada 1 [45]. Outros métodos podem incorporar esquemas mais sofisticados para classificação diferencial [45].

Se o valor de determinado atributo A está ausente na tupla T1 e/ou na tupla T2, o modelo pode assumir a diferença máxima possível, supondo que cada um dos atributos tenha sido mapeado para o intervalo [0,1] [45]. Para atributos nominais, pode ser considerado o valor da diferença ‘1’ se um ou ambos os valores correspondentes de ‘A’ estiverem ausentes [45]. Se ‘A’ for numérico e ausente em ambas as tuplas T1 e T2, a diferença também será considerada 1 [45]. Se apenas um valor estiver ausente e o outro (vamos chamar de v’) estiver presente e normalizado, então poderemos considerar a diferença como |1 - v’| ou |0 - v’| (ou seja, 1 - v’ ou v’), o que for maior [45].

Um bom valor para **k** pode ser determinado experimentalmente [45]. Começando com $k = 1$, usamos um conjunto de testes para estimar a taxa de erro do classificador [45]. Esse processo pode ser repetido a cada vez, incrementando k para permitir mais um vizinho [45]. O valor k que fornece a taxa de erro mínima pode ser selecionado [45]. Em geral, quanto maior o número de tuplas de treinamento, maior será o valor de k (de modo que as decisões de classificação e predição numérica podem ser baseadas em uma porção maior de tuplas armazenadas) [45]. Conforme o número de tuplas de treinamento se aproxima do infinito e $k = 1$, a taxa de erro não pode ser pior do que duas vezes a taxa de erro de Bayes (sendo o último mínimo teórico) [45]. Se k também se aproxima do infinito, a taxa de erro se aproxima da taxa de erro de Bayes [45].

O KNN é uma maneira relativamente simples de agrupar documentos com base em suas distâncias até os K documentos "vizinhos" mais próximos, sendo essas distâncias medidas em razão das variáveis de texto aplicadas à eles [81]. É um método simples, mas eficaz para uma classificação [41].

O KNN é apropriado para um corpus que contém vários grupos de documentos relativamente semelhantes [81]. Uma desvantagem desse algoritmo é que ele é relativamente lento para concluir a operação de agrupamento e pondera todas as variáveis igualmente, mesmo quando algumas possam ser mais importantes do que outras [81]. Sua baixa eficiência, sendo um método de aprendizagem preguiçosa, o torna proibitivo em muitas aplicações como na mineração da web dinâmica para grandes repositórios, por exemplo [41].

Classificadores de vizinho mais próximo usam comparações baseadas em distância que intrinsecamente atribuem peso igual a cada atributo [45]. Portanto, eles podem sofrer de baixa precisão quando recebem atributos ruidosos ou irrelevantes [45]. O método, no entanto, foi modificado para incorporar a ponderação de atributo e a remoção de tuplas de dados com ruído [45]. A escolha de uma medida de distância pode ser crítica [45]. A distância de Manhattan (quarteirão), ou outras medidas de distância, também podem ser usadas [45].

Classificadores de vizinho mais próximo podem ser extremamente lentos ao classificar tuplas de teste [45]. Se D for um banco de dados de treinamento de |D| tuplas e $k = 1$, então O (|D|) comparações são necessárias para classificar uma dada tupla de teste [45]. Ao pré-classificar e organizar as tuplas armazenadas em árvores de pesquisa, o número de comparações pode ser reduzido a O (log |D|) [45]. A implementação paralela pode reduzir o tempo de execução a uma constante, ou seja, O (1), que é independente de |D| [45].

Outras técnicas para acelerar o tempo de classificação incluem o uso de cálculos de distância parcial e edição das tuplas armazenadas [45]. No método de distância parcial, calculamos a distância com base em um subconjunto de n atributos [45]. Se essa distância exceder um limite, a computação posterior para a tupla armazenada fornecida é interrompida e o processo segue para a próxima tupla armazenada [45]. O método de edição remove tuplas de treinamento que se provam inúteis [45]. Esse método também é conhecido como poda ou condensação porque reduz o número total de tuplas armazenadas [45].

II.5.3 Naive Bayes

O *Naive Bayes* é classificador probabilístico simples baseado no teorema de Bayes, que afirma que a probabilidade de ocorrência de um evento é igual à probabilidade intrínseca (calculada a partir dos dados disponíveis no presente) vezes a probabilidade de que aconteça novamente no futuro (com base no conhecimento de sua ocorrência no passado) [81]. Essa ideia é usada em classificadores *Naive*

Bayes apenas na medida em que a proporção de cada classe em um conjunto de dados reflete a probabilidade "anterior" de que qualquer novo objeto pertencerá a uma classe ou outra [81].

Em termos Bayesianos, uma tupla X de dados é considerada uma "evidência", sendo descrita por medições feitas num conjunto de n atributos [45]. Considerando H alguma hipótese, como a de que a tupla de dados X pertence a uma classe específica C, para problemas de classificação, queremos determinar P(H | X), isto é, a probabilidade de que a hipótese H se sustenta dada a "evidência" ou tupla de dados observados X [45]. Estamos procurando a probabilidade da tupla X pertencer à classe C, dado que conhecemos a descrição dos atributos de X [45].

P (H | X) é a probabilidade posterior, ou probabilidade a posteriori, de H condicionado em X [45]. Por exemplo, suponha que a nossa tupla represente dados de alunos de graduação em Ciência da Computação no CEFET-RJ, sendo atributos de idade e estado civil, respectivamente, e que X seja um aluno de de 33 anos e solteiro. Suponha que H seja a hipótese de que nosso aluno entrará no curso de pós-graduação em Ciência da Computação na mesma instituição. Então, P (H | X) representa a probabilidade de que o aluno X irá entrar no curso de pós graduação, dado que sabemos a idade e estado civil do mesmo.

Em contraste, P (H) é a probabilidade anterior, ou probabilidade a priori, de H [45]. No nosso exemplo essa é a probabilidade de que qualquer aluno entrará no curso de pós-graduação, independentemente da idade, estado civil ou qualquer outra informação para esse assunto. A probabilidade posterior, P (H | X), é baseada em mais informações (nesse exemplo, informações do aluno) do que a probabilidade anterior, P (H), que é independente de X.

Da mesma forma, P (X | H) é a probabilidade posterior de X condicionada em H [45]. Ou seja, é a probabilidade de que um aluno, X, tenha 33 anos e seja solteiro, dado que sabemos que o aluno entrará na pós-graduação.

P (X) é a probabilidade anterior de X [45]. No nosso exemplo, é a probabilidade de um aluno do nosso conjunto de estudantes de graduação ter 33 anos e ser solteiro. P (H), P (X | H) e P (X) podem ser estimadas a partir dos dados fornecidos [45]. O teorema de Bayes é útil porque fornece uma maneira de calcular a probabilidade posterior, P (H | X), de P (H), P (X | H) e P (X) [45].

Esses classificadores fazem uma suposição simplista, ingênua, de que todos os atributos de um exemplo são independentes uns dos outros, dado o contexto da classe. Ou seja, presume a independência de variáveis aleatórias [81; 74; 65]. Essa suposição é chamada de independência condicional de classe e é feita para simplificar os cálculos envolvidos [45]. Apesar de seu *design* simplista e sua suposição ingênua (não verdadeira na maioria das tarefas do mundo real), esses classificadores podem ser notavelmente eficientes e precisos, particulamente quando o número de variáveis é alto [81; 74].

O cálculo é realizado conforme a equação II.3, em que C indica as classes e A os atributos. O

Multinomial *Naive Bayes* utiliza uma variação dessa teoria e usa um modelo multinomial para obter a frequência de uma variável no conjunto [74].

$$P(A|C) = P(A)\frac{P(C|A)}{P(C)} \tag{II.3}$$

II.5.4 Random Forests

O RF é um tipo de método conjunto, *ensemble method*, que combina vários modelos de árvores de decisão. Um *ensemble* combina uma série de k modelos treinados (ou classificadores), M1, M2, ..., Mk, com o objetivo de criar um modelo de classificação composto aprimorado, M [45]. Um dado conjunto de dados, D, é usado para criar k conjuntos de treinamento, D1, D2, ..., Dk, onde Di ($1 \leq$ i $\leq$ k $-$ 1) é usado para gerar o classificador Mi [45]. Dada uma nova tupla de dados para classificar, os classificadores votam cada um retornando uma previsão de classe [45]. O *ensemble* retorna uma previsão de classe com base nos votos dos classificadores [45]. Um *ensemble* tende a ser mais preciso do que seus classificadores trabalhando separadamente [45].

No RF são utilizadas somente Árvores de Decisão. As árvores individuais são geradas usando uma seleção aleatória de atributos em cada nó para determinar a divisão [45]. Cada árvore depende dos valores de um vetor aleatório amostrado independente e com a mesma distribuição para todas as árvores da floresta [45]. Durante a classificação, cada árvore vota e a classe mais popular é retornada [45].

Em geral, para gerar k árvores de decisão para o Ensemble, temos o seguinte: para cada iteração, i (i = 1, 2, ..., k), um conjunto de treinamento, Di , de d tuplas é amostrado com substituição de D [45]. Ou seja, cada Di é uma amostra *bootstrap* de D, de modo que algumas tuplas podem ocorrer mais de uma vez em Di, enquanto outras podem ser excluídas [45]. Seja F o número de atributos a serem usados para determinar a divisão em cada nó, onde F é muito menor que o número de atributos disponíveis [45]. Para construir um classificador de árvore de decisão, Mi, selecione aleatoriamente, em cada nó, F atributos como candidatos para a divisão no nó [45].

A metodologia CART é usada para cultivar as árvores [45]. O Classification and Regression Trees, em inglês (CART) suporta variáveis de destino numéricas (regressão) e não computa conjuntos de regras [91]. As árvores são cultivadas até o tamanho máximo e não são podadas [45]. RF formados dessa forma, com seleção aleatória de entrada, são chamados de *Forest RI* [45]. Outra forma de RF, chamado *Forest RC*, usa combinações lineares aleatórias dos atributos de entrada [45]. Em vez de selecionar aleatoriamente um subconjunto dos atributos, ele cria novos atributos (ou recursos) que são uma combinação linear dos atributos existentes [45].

Ou seja, um atributo é gerado especificando L, o número de atributos originais a serem combinados [45]. Em um dado nó, L atributos são selecionados aleatoriamente e somados com coeficientes

que são números aleatórios uniformes em [1, 1] [45]. F combinações lineares são geradas e é feita uma pesquisa sobre eles para a melhor divisão [45].

Essa forma de RF é útil quando há apenas alguns atributos disponíveis, de modo a reduzir a correlação entre classificadores individuais [45]. RF são comparáveis em precisão ao *AdaBoost*, mas são mais robustas a erros e discrepâncias [45]. O erro de generalização para uma *floresta* converge enquanto o número de árvores na floresta for grande [45].

Assim, *overfitting* não é um problema [45]. A precisão de um RF depende da força dos classificadores individuais e da medida da dependência entre eles [45]. O ideal é manter a força dos classificadores individuais sem aumentar sua correlação [45]. RF são insensíveis ao número de atributos selecionados para consideração em cada divisão [45]. O mesmo autor fez uma observação empírica interessante, indicando que o uso de um único atributo de entrada aleatório pode resultar em boa precisão - que geralmente é maior do que ao usar vários atributos.

Como os RF consideram muito menos atributos para cada divisão, eles são eficientes em bancos de dados muito grandes [45]. Eles podem ser mais rápidos do que 'ensacamento' (*bagging*) ou 'impulsionamento' (*boosting*) [45]. Os RF fornecem estimativas internas de importância variável [45].

II.5.5 Bi-LSTM

As Redes Neurais Recorrentes (Recurrent Neural Networks, em inglês) (RNN) utilizam a saída resultante do processamento de uma entrada prévia como *input* para a próxima entrada no processamento de cada neurônio [38]. Ou seja, a entrada é formada não apenas pelo exemplo de entrada atual que está sendo visto, mas também pelo que foi percebido anteriormente no tempo. Temos, assim, que as redes recorrentes trabalham com duas fontes de entrada, o passado recente e o presente, sendo a combinação dos dois a forma utilizada para determinar como será a resposta a novos dados [38].

Os RNN se diferenciam das redes *feedforward* pelo *loop* de *feedback* conectado às suas decisões anteriores, adicionando suas próprias saídas momento após momento como entrada [38]. Esse mecanismo atua como se fosse uma *memória*. Essa adição de *memória* às redes neurais tem o objetivo de fazer com que o RNN consiga absorver informações na própria sequência, permitindo-os fazer o uso dessas informações em tarefas que as redes *feedforward* não conseguem lidar [38].

A informação sequencial fica armazenada no estado oculto do RNN, que passa por muitas etapas de tempo à medida que avança em sequência para afetar o processamento de cada novo exemplo [38]. Essas correlações entre o que está sendo observado são separadas por muitos momentos, sendo chamadas de "dependências de longo prazo", porque um evento no tempo depende e é uma função de um ou mais eventos que o precederam [38]. O RNN é uma forma de compartilhar pesos ao longo

do tempo. A informação circula nos estados ocultos do RNN afetando o *comportamento* da rede ao longo do processamento.

O problema relacionado a dissipação do gradiente e a dificuldade em treinar as redes neurais artificiais é mais acentuado com os RNN [38]. Essas redes possuem dificuldades para lidar e aprender dependências de longo alcance, tais como as ligações entre palavras que estão separadas por muitas palavras intermediárias, por exemplo. Isso acaba representando um problema pois o significado da frase pode ser determinado por palavras que não estão muito próximas. Para lidar com essa questão é utilizada regularização. Outro forma é utilizar funções de ativação $ReLU$ em vez de utilizar *tanh* ou *sigmóide*, já que a derivada $ReLU$ é uma constante de 0 ou 1 e, assim, não é tão provável que ocorra a dissipação do gradiente.

A idéia principal das Redes Neurais Recorrentes Bidirecionais (Bidirectional Recurrent Neural Networks, em inglês) (BRNN) [104; 10] é apresentar os dados sequenciais de treinamento para frente e para trás. Isso significa que para cada ponto em uma determinada sequência, a BRNN tem informações sequenciais completas sobre todos os pontos anteriores e posteriores [39]. Para dados textuais é comum que o BRNN apresente um desempenho preditivo melhor se não processar apenas a sequência do início ao fim, mas também do fim ao início, em torno de um caracter ou palavra [39]. Para prever a próxima palavra em uma frase, ou caracter numa palavra, geralmente é útil ter o contexto em torno da palavra, ou do caracter, não apenas as palavras ou caracteres que vêm antes da palavra ou caracter alvo [39]. De fato, o melhor desempenho preditivo foi obtido com o processamento bidirecional em variadas aplicações [107].

Os RNN mostraram grande sucesso em muitas tarefas do NLP, sendo uma das arquiteturas mais prevalentes devido a sua capacidade de lidar com textos de comprimento variável [99]. Os tipos de RNN mais utilizados são as LSTM, que são melhores na captura de dependências de longo prazo do que os RNN em sua arquitetura padrão. As LSTM são essencialmente a mesma estrutura que os RNN, mas possuem uma maneira diferente de computar o estado oculto [38].

As LSTM [51], são RNN cujos neurônios também possuem um estado interno que é utilizado como espaço de memória para o processamento. A LSTM pode capturar as dependências longas em uma sequência introduzindo uma unidade de memória e um mecanismo de portões que visa decidir como utilizar e atualizar as informações mantidas na célula de memória [99]. A informação pode ser armazenada e acessada em momentos temporais diferentes. Simplificadamente, para cada entrada processada por um neurônio, entram no processamento, também, a saída anterior e o que está armazenado no estado interno. O resultado dos cálculos de cada neurônio é utilizado não só para prover um valor de saída, mas também para atualizar o estado interno [38]. São utilizada em aplicações em que a entrada é uma sequência ordenada e a informação recente na sequência pode ser importante [38].

As LSTM foram propostas pela primeira vez em 1997 por [51] e são modelos bastante utilizados em Processamento de Linguagem Natural, sendo populares para lidar com classificação de sentimentos [99]. O impacto da rede LSTM é notável em transcrição de fala para texto, modelagem de linguagem, tradução automática, dentre outros usos [68].

A Unidade Recorrente Fechada (Gated Recurrent Unit, em inglês) (GRU), proposta pela primeira vez em 2014 [21], é uma versão simplificada da LSTM [38]. As redes LSTM e GRU são arquiteturas RNN explicitamente projetadas para lidar com a dissipação do gradiente e aprender eficientemente dependências de longo alcance [107; 21]. O principal insight no projeto das LSTM foi a inserção de controles não lineares dependentes de dados na célula RNN, que podem ser treinados para garantir que o gradiente da função objetivo em relação ao sinal de estado não desapareça (a quantidade diretamente proporcional ao parâmetro seja atualizado durante o treinamento pelo método do Gradiente Descendente) [107; 51].

A estrutura de um RNN é muito semelhante ao Modelo Oculto de Markov (Hidden Markov Model, em inglês) (HMM) [38]. A principal diferença é como os parâmetros são calculados e construídos [38]. Uma das vantagens da LSTM é a insensibilidade ao comprimento do gap. RNN e HMM dependem do estado oculto antes da emissão/sequência [38]. Se o objetivo for prever a sequência após 1.000 intervalos em vez de 100, o modelo esqueceu o ponto de partida até então observado [38]. Mas um modelo LSTM é capaz de "lembrar" por conta de sua estrutura de células, o diferencial da arquitetura LSTM [38].

O LSTM possui uma estrutura em cadeia que contém quatro redes neurais possuindo, também, diferentes blocos de memória chamados células [38]. As informações são retidas pelas células e as manipulações de memória são feitas pelos portões (*gates*) [107], que controlam o fluxo da informação dentro da célula. Eles possuem parâmetros como pesos e bias. Existem três portões [107]; um que controla o quanto de informação armazenada é usada como entrada para os cálculos, outro que controla o quanto de informação atual é armazenada no estado e um terceiro que controla o quanto a saída é determinada pelo cálculo atual comparado a informação armazenada. Os parâmetros dos portões também são aprendidos durante o treinamento [38; 107].

Uma Bi-LSTM é formado por dois LSTM: uma que recebe a entrada em uma direção direta e a outra na direção inversa. As Bi-LSTM aumentam, de forma efetiva, a quantidade de informações para a rede, possibilitando uma melhora da percepção de contexto pelo algoritmo [38; 107].

II.5.6 BERT

BERT é um algoritmo de aprendizagem profunda para NLP [27]. O BERT foi projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente o contexto esquerdo e direito em todas as camadas [27]. Como resultado, o modelo BERT pré-

treinado pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração para uma ampla gama de tarefas, como resposta a perguntas e inferência de linguagem, sem modificações substanciais na arquitetura específica da tarefa [27].

O BERT lida com o dado textual e computa os CWE com base no conjunto completo das palavras que orbitam uma determinada palavra (seus arredores, à esquerda e à direita da palavra). Ele faz uso do 'Transformer' para modelagem da linguagem, lendo toda a sequência de palavras de uma vez. [27]. É considerado bidirecional, mas é mais preciso enxergá-lo como não direcional. Isso o diferencia dos esforços anteriores, dos MLN tracionais, já que esses analisam uma sequência de texto da esquerda para a direita ou, de forma combinada, da esquerda para a direita e da direita para a esquerda. Assim, o BERT pode extrair um senso mais profundo de contexto e do fluxo de linguagem dos que os MLN de direção única [27].

Por meio do 'Transformer' o BERT consegue discernir o contexto das palavras com base no conjunto de palavras dos arredores (esquerdo e direito da palavra) - em vez de lidar de forma ordinária com palavras que seguem ou precedem o termo-alvo - aprendendo, assim, as relações contextuais entre palavras em um texto [27]. O 'Transformer', em sua concepção original, inclui dois mecanismos separados, o codificador que lê a entrada de texto e o decodificador que produz uma previsão para a tarefa. O BERT faz uso, no 'Transformer', apenas do mecanismo do codificador, já que o objetivo é gerar um modelo de linguagem [27].

Quando treinamos MLN utilizamos alguma estratégia de previsão dos termos processados, sendo a previsão da próxima palavra em uma sequência algo usual. Essa abordagem direcional limita o aprendizado do contexto. O BERT usa duas estratégias de treinamento que entregam melhores resultados: Modelagem de Linguagem Mascarada (Masked-Language Modeling, em inglês) (MLM) e a Previsão da Próxima Frase (Next Sentence Prediction, em inglês) (NSP) [27].

No MLM, antes de usar as sequências de palavras no BERT, 15% dos *tokens* em cada sequência são escolhidos aleatoriamente e substituídos por outro *token* [27]. 80% desses *tokens* são substituídos pelo *token [MASK]*, 10% por uma palavra aleatória e em 10% são mantidas as palavras originais [27]. O algoritmo passa, então, a tentar prever as palavras mascaradas tendo por base o contexto fornecido pelas outras palavras na sequência que não foram mascaradas [27].

A função de perda do BERT considera somente a previsão dos *tokens* que foram mascarados, ignorando a previsão daqueles que não foram [27]. A abordagem não direcional MLM do BERT converge mais lentamente do que os modelos direcionais (dado que apenas 15% das palavras são previstas em cada lote), mas isso é compensado por sua maior percepção do contexto e esse tipo de treinamento supera o treinamento direcional após um pequeno número de etapas de pré-treinamento [27].

Na NSP, durante o treinamento, o modelo recebe, como entrada, pares de frases e o objetivo é

aprender a prever se a segunda frase do par é a frase subsequente no dado de origem [27]. Nesse processo de treinamento, 50% das entradas são pares de frases em que a segunda frase é a frase subsequente no documento de origem e os outros 50% são pares de frases em que a segunda frase é uma frase aleatória escolhida no corpus como segunda frase [27]. Para fazer a distinção entre as duas sentenças no treinamento a entrada é processada antes de entrar no modelo, recebendo um *token [CLS]* no início da primeira frase e um *token [SEP]* no final de cada frase [27]. Um *embedding* de frase com referência a frase A ou a frase B e um *embedding* posicional, indicando sua posição na sequência, são adicionados a cada *token* [27].

Após essa preparação, com o intuito de prever se a segunda frase está conectada à primeira frase, toda a sequência de entrada passa pelo modelo do 'Transformer', a saída do *token [CLS]* é transformado em um vetor usando uma camada de classificação simples (matrizes aprendidas de pesos e vieses) e a probabilidade da frase seguinte ser a frase seguinte no dado de origem é calculada com a função de ativação *softmax* [27]. Ao treinar o BERT, o MLM e a NSP são treinados em conjunto tendo como objetivo a minimização da função de perda combinada das duas abordagens [27].

O BERT possui uma limitação quantitativa de 512 *tokens* para cada uma das entradas que é processada pelo modelo [111; 27]. O limite deriva dos embeddings posicionais processados na arquitetura do 'Transformer', sendo necessária a imposição de um comprimento máximo. Tanto a implementação original do BERT quanto muitas derivadas (que utilizam 'Transformer') consumirão 512 *tokens* no máximo, truncando e descartando *tokens* excedentes [111; 27]. O BERT recebe uma entrada de no máximo 512 *tokens* e gera a representação da sequência. A sequência tem um ou dois segmentos (sentenças), onde o primeiro *token* da sequência é sempre *[CLS]* e o *[SEP]* é utilizado para separar os segmentos [111]. Com isso, ao lidar com textos longos, é necessário utilizar uma estratégia para escolher 510 *tokens* da entrada, caso contrário, o próprio BERT fará isso, escolhendo os 510 primeiros. Há métodos hierárquicos e métodos de truncamento [111]. Na estratégia de truncamento, como exemplo, nós podemos manter os primeiros 510 *tokens*, manter os últimos 510 *tokens* ou selecionar os primeiros 128 e os últimos 382 *tokens*, descartando os demais [111]. Essa escolha precisa observar as características dos dados textuais processados e é variável.

II.6 F1-Score

O F1-score ou F1 é a média harmônica da precisão e da revocação, que são outras duas medidas [44]. As medidas de precisão e revocação são amplamente utilizadas na classificação, sendo suas pontuações normalmente usadas juntas, onde os valores de precisão são comparados para um valor fixo de revocação, ou vice-versa [44].

A precisão é o percentual de documentos corretamente classificados como positivos, sendo real-

mente positivos. Ou seja, qual o percentual de acertos de uma classificação em que os documentos foram classificados como sendo da classe A e realmente são dessa classe. A equação II.4 mostra o cálculo da precisão; TP (*true positive*) faz referência aos documentos classificados como positivos, sendo positivos de fato, e FP (*false positive*) faz referência aos documentos que foram classificados como positivos e que, na verdade, são negativos.

$$precision = \frac{TP}{TP + FP} \tag{II.4}$$

A revocação indica, no contexto da aprendizagem de máquina, a porcentagem de documentos positivos que são classificados como positivos. A equação II.5 mostra o cálculo da revocação. TP (*true positive*) faz referência aos documentos que foram classificados como positivos e, de fato, são positivos, e FN (*false negative*) faz referência aos documentos que foram classificados como negativos e que, na verdade, são positivos.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{II.5}$$

Em suma, a precisão pode ser pensada como uma medida de exatidão (qual a porcentagem de tuplas rotuladas como positivas são realmente positivas?), enquanto o *recall* é uma medida de completude (qual a porcentagem de tuplas positivas são rotuladas como positivas?) [44]. O *recall* é o mesmo que sensibilidade ou a taxa de verdadeiro positivo [44].

A medida F1-score ou F1 combina a precisão e a revocação numa única medida, como indicado na equação II.6:

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{II.6}$$

II.7 Validação Cruzada k-fold

A validação cruzada k-fold é uma forma de validação de modelos de aprendizagem de máquina [64; 44]. Para estimar a capacidade de generalização do modelo, os dados iniciais são divididos aleatoriamente em k subconjuntos mutuamente exclusivos, D1, D2, ..., DK, cada um de tamanho aproximadamente igual, sendo treinamento e teste realizados k vezes [44; 64]. Na iteração i, a partição DI é reservada como o conjunto de teste e as partições restantes são usadas coletivamente para treinar o modelo [44]. Assim, na primeira iteração, os subconjuntos D2,..., DK servem, em grupo, como o conjunto de treinamento para obter um primeiro modelo, que é testado em D1; a segunda iteração é treinada nos subconjuntos D1, D3, ..., Dk e testados em D2; e assim por diante [44]. A figura II.1 exemplifica a primeira iteração utilizando um $k = 5$, como exemplo.

A figura II.2 mostra como ficam as diferentes iterações e divisões no conjunto de dados com o

Figura II.1: Exemplo de divisão do conjunto de dados para validação cruzada com k=5. Fonte: Elaboração Própria.

mesmo $k = 5$ da figura II.1.

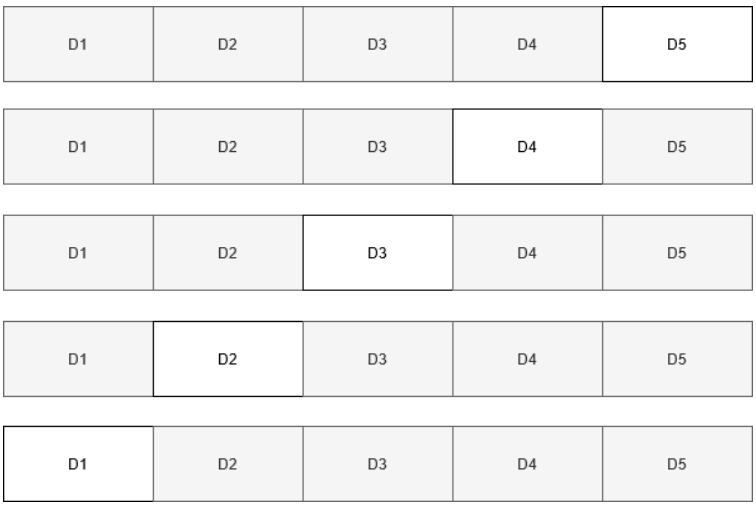

Figura II.2: Subconjuntos em azul para treinamento e em branco para teste. Fonte: Elaboração Própria.

Ao contrário dos métodos de validação e subamostragem aleatória, aqui cada exemplo é usado o mesmo número de vezes para treinamento e uma vez para teste [44]. Para classificação, a estimativa da acurácia é obtida pela média das acurácias das k iterações [44; 64]. A capacidade do modelo é obtida pela média das medidas das k iterações, da mesma forma.

Na validação cruzada estratificada, os subconjuntos são estratificados de modo que a distribuição de classe das tuplas em cada subconjunto é aproximadamente o mesmo que nos dados iniciais [44]. Em geral, a validação cruzada estratificada de 10 é recomendado para estimar a precisão (mesmo que o poder de computação permita o uso de mais subconjuntos) devido ao seu viés e variância relativamente baixos [44].

II.8 Grid Search

O Grid Search (GS) enumera exaustivamente todas as combinações de hiperparâmetros de um subconjunto especificado do espaço de hiperparâmetros de um algoritmo de aprendizado e avalia cada combinação [15; 56]. O algoritmo deve ser guiado por alguma medida de desempenho preditivo, normalmente medida por *cross-validation* no conjunto de treinamento [15; 56]. É uma abordagem computacional que pode levar um tempo considerável, uma vez que depende dos recursos computa-

cionais disponíveis, da natureza do algoritmo de aprendizagem e do tamanho do problema [15; 35].

O GS é muito utilizado, apesar de décadas de pesquisa em otimização global, uma vez que é simples de implementar, a paralelização é factível e esse tipo de busca é confiável em espaços de baixa dimensão [15; 35]. O problema com essa abordagem é que o número de avaliações aumenta exponencialmente à medida que aumentamos o subconjunto especificado do espaço de hiperparâmetros do algoritmo de aprendizado utilizado [98]. Ao lidar com grandes conjuntos de dados, o controle do espaço de hiperparâmetros e a utilização de um subconjunto dos dados são necessários para garantir que o método pare em um tempo razoável, embora isso diminua a validade da solução [98].

II.9 Halving Grid Search

No Halving Grid Search (HGS) o treinamento é realizado nos subconjuntos de dados, em vez de utilizar todo o conjunto. Após N número de iterações, ocorre a seleção dos melhores candidatos com um tempo de avaliação consideravelmente mais rápido [59; 91].

Em dezembro de 2020 o *scikit-learn* lançou a implementação de duas novas classes para ajuste de hiperparâmetros - *HalvingGridSearch* e *HalvingRandomSearchCV* [91]. O *Grid Search* e o *Random Search* treinam os candidatos em todos os dados de treinamento, enquanto o HGS e o *Halving Random Search* adotam um procedimento diferente chamado *halving* sucessivo [59; 91].

No HGS temos uma disputa entre todas as combinações de hiperparâmetros candidatas. Na primeira iteração o HGS treina todos os candidatos utilizando um pequeno subconjunto dos dados de treinamento [91]. Na próxima iteração apenas os candidatos com melhor desempenho preditivo são escolhidos e recebem mais dados para continuarem na avaliação [91]. Com isso, a cada nova iteração, o número de candidatos restantes passa a receber cada vez mais exemplos do conjunto até que o melhor candidato, isto é, o melhor conjunto de hiperparâmetros, seja mantido [91]. Esse processo pode ser controlado, na implementação do *scikit-learn*, por dois argumentos principais, *factor* e *min resources* [91]. O *min resources* utiliza um número inteiro que especifica a quantidade de exemplos do conjunto de treinamento que será usada na primeira iteração [91]. Todos os candidatos são treinados com esses dados e, na próxima iteração, *min resources* cresce e o número de candidatos diminui pelo *factor* especificado, ficando apenas os melhores candidatos da iteração anterior [91]. A iterações seguintes obedecem a mesma regra até que o melhor candidato seja encontrado [91].

Se, por exemplo, temos um conjunto com 800 exemplos e 40 candidatos (combinações possíveis entre os hiperparâmetros) na grade de parâmetros, ao utilizar um *min resources* de 30 e escolhermos um *factor* de 2, teremos o quadro II.1 abaixo:

Usando o *min resources* com 30 com 40 candidatos, só poderemos executar cinco iterações porque ficaremos sem candidatos antes de esgotar todos os exemplos do conjunto. Nesse caso, o

iteração	número exemplos	número candidatos	fator
1	30	40	2
2	60	20	2
3	120	10	2
4	240	5	2
5	480	2	2

Tabela II.1: Halving Grid Search

restante dos exemplos não utilizados (800 - 480 = 320) será descartado e o melhor candidato será encontrado treinando apenas em 480 exemplos do conjunto de dados.

A combinação do *min resources* e *factor* precisa ser bem pensada, já que é possível, também, ficar com muitos candidatos lidando com a maior parte do conjunto de dados, hipótese em que poderá ser observada um impacto no tempo de processamento e o usuário não irá tirar proveito dessa abordagem algorítmica. Caso o usuário não queira escolher o *min resources* é possível utilizar o valor *default* 'exhaust'. Nesse caso o número mínimo de recursos será determinado automaticamente para criar a melhor combinação possível com *factor* e o número de candidatos [91]. Com o 'exhaust' a primeira iteração é definida de modo que a última iteração use o máximo de exemplos possível do conjunto [91]. Em geral, isso leva a uma escolha mais precisa na seleção do melhor candidato, mas consome mais tempo no processamento [91].

Capítulo III Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos relacionados ao estudo desta dissertação. O objetivo foi o de identificar trabalhos que comparassem, em alguma medida, diferentes abordagens na classificação de texto; diferentes formas de representação textual e seus usos em algoritmos de aprendizagem de máquina. Além desses trabalhos comparativos, foram pesquisados trabalhos que indicassem aplicações utilizando algum método de representação textual e, também, aqueles estudos que, não realizando uma comparação direta entre métodos de representação textual diferentes ou não indicando a aplicação de um dos métodos num caso concreto, fossem propostas de evolução dessas representações por meio de novos algoritmos para tarefas no campo do NLP. Nesse sentido, os trabalhos relacionados ao estudo estão divididos em três grupos.

Foram utilizadas *strings* preliminares no decorrer do estudo para observar as pesquisas no campo, tais como: ("analysis of textual representation" AND "bag-of-words" AND "neural language models"), ("analysis of textual representation") e ("textual representation" AND "bag-of-words" AND "neural language models"). Essas buscas ocorreram em datas diferentes e não trouxeram um resultado relevante. Em 21/12/2021 foi utilizada a *string* "bag-of-words" AND "neural language models" no *scopus* [Scopus website] e foram retornados 17 documentos. Os trabalhos descritos resumidamente abaixo são aqueles que estão comparando, em alguma medida, como dito acima, diferentes abordagens na classificação de texto - diferentes formas de representação de palavras/frases e usos em algoritmos de aprendizagem de máquina - ou apresentando uma aplicação de ao menos um dos métodos de representação textual num caso concreto, ou apresentando propostas para aperfeiçoamento dos métodos de representação textual por meio de novos algoritmos para execução de tarefas no campo do NLP. Nesse último caso foram incluídos estudos que marcaram avanços na área, independentemente de fazerem parte do resultado da *string* utilizada e mencionada acima.

Em [8], [25], [6] e [48] foi observada, em alguma medida, uma comparação entre diferentes formas de representação de palavras/frases e usos em algoritmos de aprendizagem de máquina. [8] investigou a mineração de opinião para análises de aplicativos. O estudo comparou técnicas de representação textual para classificação, análise de sentimento e previsão de utilidade a partir de avaliações de aplicativos. Foram discutidas e avaliadas diferentes técnicas para a representação textual de revisões, desde o tradicional BOW até os mais recentes modelos de linguagem neural MLN. Os resultados indicaram que o modelo tradicional de BOW, combinado com uma análise

cuidadosa das técnicas de pré-processamento de texto, ainda é competitivo, obtendo resultados próximos aos modelos de linguagem neural nas tarefas de classificação, análise de sentimento e previsão de utilidade. No entanto, observaram os autores, os MLN se mostraram mais vantajosos, uma vez que alcançaram um desempenho preditivo muito competitivo em todas as tarefas preditivas abordadas no trabalho, fornecendo redução de dimensionalidade significativa e lidando de forma mais adequada com a proximidade semântica entre os textos das avaliações.

[25] realizou um estudo sobre o BERT, modelo de linguagem neural contextual, com o intuito de obter uma compreensão textual mais profunda no campo de Recuperação da Informação. Os resultados experimentais indicaram que as representações contextuais do BERT são mais eficazes do que os *embeddings* tradicionais de modelos que o precederam. Comparado ao modelo de Recuperação da Informação baseado no BOW, o BERT pode aproveitar melhor as estruturas da linguagem, trazendo grandes melhorias nas consultas escritas em linguagens naturais. A combinação da habilidade de compreensão de texto com o conhecimento de pesquisa leva a um modelo BERT pré-treinado aprimorado que pode beneficiar tarefas de pesquisa relacionadas em contextos em que os dados de treinamento são limitados.

[6] realizou um estudo de modelos de classificação para atribuição de autoria. Foi utilizado um modelo de linguagem neural semi-supervisionado conhecido como *embeddings* de documentos para lidar com a questão. Foram utilizados vetores de parágrafos que, ao constrário de redes neurais convolucionais ou LSTMs, são semi-supervisionados e menos propensos a parâmetros de aprendizagem, segundo os autores. Eles podem ser utilizados para construir meta-perfis para autores e também podem ser construídos a partir de *n-grams* de caracteres e *n-grams* de símbolos da mesma maneira. O método indicou melhorias significativas em relação às representações de BOW no experimento realizado.

[48] propôs um modelo de linguagem neural simples e eficiente para tarefas de classificação em nível de sentença. Foi utilizada uma LSTM, sobre vetores pré-treinados obtidos de um modelo de linguagem neural não supervisionado para capturar semântica e informações sintáticas em uma frase curta. Os resultados empíricos, segundo os autores, foram excelentes em vários conjuntos de dados usados como *benchmark*, como o conjunto de dados de análise de sentimento IMDB e o conjunto de dados *Stanford Sentiment Treebank*. Os resultados empíricos mostram que o modelo é comparável aos métodos neurais e supera os métodos tradicionais na tarefa de análise de sentimento.

Em [112], [40], [126], [53], foi observada a utilização de algum método de representação textual em alguma aplicação, alguma tarefa específica/caso concreto. [112] apresentou uma metodologia de recuperação de informações para identificação de fontes relevantes a respostas de questionamentos biomédicos feitas em linguagem natural no contexto da pandemia do covid-19. A metodologia apresenta múltiplos estágios e combina modelos de ponderação probabilística e algoritmos de ran-

queamento baseados em redes neurais profundas para impulsionar a classificação de documentos relevantes. A combinação de BOW e modelos de linguagem neural profunda superou significativamente o *baseline* 'Okapi Best Match 25', recuperando em média 83% dos documentos relevantes entre os 20 primeiros. Os autores concluíram que os resultados indicam que a recuperação em vários estágios suportada pela aprendizagem profunda pode melhorar a identificação adequada da literatura para questões relacionadas ao COVID-19 colocadas em linguagem natural.

[40] propôs uma abordagem baseada em aprendizagem profunda para gerar sequências de uso de *API* para uma determinada consulta de linguagem natural. Obter uma sequência de uso de uma dada *API* com base em uma consulta de linguagem natural relacionada à essa *API* é útil para ajudar desenvolvedores a implementarem determinada funcionalidade. Dada uma consulta, as abordagens existentes utilizam modelos de recuperação de informações para pesquisar sequências nas *APIs* correspondentes. Essas abordagens tratam as consultas e *APIs* como pacotes de palavras e carecem de um entendimento profundo da semântica da consulta. No modelo proposto, em vez de usar a perspectiva do BOW, ele aprende a sequência de palavras em uma consulta e a sequência de *APIs* associadas. O 'DeepAPI' adapta um modelo de linguagem neural denominado RNN Encoder-Decoder. Ele codifica uma sequência de palavras (a consulta do usuário) em um vetor de contexto de comprimento fixo e gera uma sequência de *API* com base no vetor de contexto. A abordagem foi avaliada empiricamente com mais de 7 milhões de trechos de código anotados coletados do *GitHub*. Os resultados mostraram que a abordagem gera sequências *API* altamente precisas e supera as abordagens relacionadas.

[126] analisou como as representações vetoriais distribuídas de palavras podem ser usadas em um modelo de recuperação de informação e quais são seus benefícios. Foi utilizado *embeddings* de palavras neurais dentro do conhecido modelo de linguagem de tradução para recuperação da informação. Nesse modelo as relações semânticas implícitas entre as palavras são capturadas, produzindo estimativas mais precisas da relevância do documento. Os *embeddings* usados para estimar modelos de linguagem neural produzem traduções que diferem de abordagens de modelos de linguagem de tradução anteriores e essas diferenças proporcionam melhorias na eficácia da recuperação. Os resultados mostraram que os *embeddings* nem precisam ser produzidos a partir do mesmo corpus que está sendo usado para recuperação.

[53] estendeu o 'word2vec' no que foi chamado de codificadores de contexto. Multiplicando a matriz treinada de *embeddings* 'word2vec' com o vetor de contexto médio das palavras, *embeddings* de palavras fora do vocabulário e representações para palavras com múltiplos significados podem ser criados com base nos contextos locais das palavras. Esses *embeddings* foram utilizados como recursos na tarefa de reconhecimento de entidade nomeada.

Em [90], [43], [49] e [114] observamos alguns exemplos de propostas para o aperfeiçoamento dos

métodos de representação textual por meio de novos algoritmos para tarefas no campo do NLP, sendo [77], [79], [58], [119], [92], [97], [27] e [125] alguns dos trabalhos mais populares dos últimos anos nesse contexto. Considere que esses estudos, descritos nos próximos parágrafos, referem-se a aperfeiçoamentos(ou tentativas) do momento de sua publicação, evidentemente. Foi observada, também, a ordem cronológica das publicações.

[90] realizou estudo objetivando aprimorar *embeddings* das palavras para que os mesmos pudessem representar não só informações contextuais, mas, também, refletir o sentimento dos textos. A metodologia abordou esse problema adicionando um aprendizado semi-supervisionado de discriminação de sentimentos usando informações parciais de sentimentos de documentos. O método não apenas refletiu a informação parcial do sentimento, mas também preservou as estruturas locais induzidas a partir dos objetivos de aprendizagem da representação distribuída original, considerando apenas as relações de sentimento entre documentos vizinhos. Os resultados da visualização de um conjunto de dados de *reviews* da Amazon indicaram o aprimoramento da separação de classes de sentimento quando as representações de documento do método proposto são comparadas a outros métodos. Os autores também observaram que a previsão de sentimento dessas novas representações também pareciam ser consistentemente superiores a outras representações nos conjuntos de dados de *reviews* da *Amazon* e do Yelp Academic Review (YAR).

[43] realizou estudo sobre dois desafios da modelagem probabilística de tópicos para melhor estimar a probabilidade de uma palavra em um determinado contexto. A primeira questão é quando quando não há estrutura de linguagem no contexto. Isso ocorre com os modelos de tópicos que ignoram a ordem das palavras resumindo um dado contexto como um BOW e, consequentemente, a semântica das palavras no contexto é perdida. Para lidar com essa questão foi realizada incorporação da estrutura da linguagem combinando um modelo de tópico auto-regressivo neural com um modelo de linguagem baseado em LSTM, o LSTM-LM, em uma única estrutura probabilística. O LSTM-LM aprende uma representação de espaço vetorial de cada palavra levando em consideração a ordem das palavras em padrões de uso, enquanto o modelo de tópico auto-regressivo neural aprende simultaneamente uma representação latente de todo o documento. O LSTM-LM modela características complexas de linguagem, como a sintaxe e a semântica, enquanto o modelo de tópico auto-regressivo identifica a estrutura temática subjacente em uma coleção de documentos. Em suma, foi realizada a união desses dois paradigmas complementares, combinando um modelo de tópico e um modelo de linguagem em uma estrutura probabilística unificada. A segunda questão da modelagem de tópicos probabilísticos endereçada pelo trabalho é quando há contexto limitado e/ou corpus de treinamento reduzido. Foi realizada a incorporação de conhecimento externo em modelos de tópicos auto-regressivos neurais por meio de uma abordagem de modelagem de linguagem: foram utilizados *embeddings* de palavras como entrada de um LSTM-LM com o objetivo de melhorar o

mapeamento de tópicos de palavras em um corpus menor ou de texo curto. Os resultados superaram consistentemente modelos de tópico generativos de última geração naquele momento em termos de generalização (perplexidade), interpretabilidade (coerência de tópico) e aplicabilidade (recuperação e classificação) em 7 conjuntos de dados de textos longos e 8 de textos curtos de diferentes domínios.

[49] realizou estudo em que foi proposto um modelo de linguagem neural que se baseia nas redes neurais convolucionais e nas redes neurais recorrentes bidirecionais sobre vetores de palavras pré-treinados. O objetivo foi o de superar as abordagens daquele momento (é anterior ao ELMo, 'Transformer', BERT, GPT e XLNet) que se utilizam de modelos de linguagem neural, já considerando que as mesmas lidam com dois pontos fracos principais do BOW; ignoram a semântica das palavras e a ordem das mesmas. Na abordagem foram utilizadas camadas bidirecionais como substitutas das camadas de *pooling* nas redes neurais convolucionais, tendo como objetivo reduzir a perda de informações locais de maior detalhe e capturar dependências de longo prazo nas sequências de entrada. O modelo foi validado em dois conjuntos de dados de análise de sentimento de referência, o 'Stanford Large Movie Review - IMDB' e o 'Stanford Sentiment Treebank - SSTb'. Foi observada uma vantagem competitiva em comparação com modelos de linguagem neural nesses conjuntos.

[114] propôs um novo modelo de linguagem neural que incorpora a ordem das palavras e dos caracteres nas representações vetoriais das palavras. O modelo produz vários espaços vetoriais com subestrutura significativa, como ficou evidenciado pelo desempenho de 85,8% em uma tarefa recente de analogia de palavras, excedendo as melhores pontuações de analogia de palavras sintáticas publicadas por uma margem de erro de 58%. O mdelo também inclui vários métodos de treinamento paralelo, mais notavelmente permitindo que uma rede *skip-gram* com 160 bilhões de parâmetros seja treinada durante a noite em 3 CPUs multi-core, 14x maior do que a maior rede neural anterior ao estudo (publicado em 2015).

[77] propôs duas novas arquiteturas de modelo para computar representações vetoriais contínuas de palavras de conjuntos de dados muito grandes. A qualidade das representações foi medida em uma tarefa de similaridade de palavras e os resultados foram comparados com as técnicas de melhor desempenho anteriores baseadas em diferentes tipos de redes neurais. Foram observadas consideráveis melhorias na precisão a um custo computacional muito menor. Também indicaram que aqueles vetores forneciam desempenho de última geração no conjunto de testes para medir semelhanças sintáticas e semânticas de palavras.

[79] apresentou extensões que melhoravam tanto a qualidade dos vetores quanto a velocidade de treinamento do trabalho apresentado em [77]. Realizando uma subamostragem das palavras frequentes, foi obtida uma aceleração significativa e representações mais regulares. Foi descrita, também, uma alternativa simples ao *softmax* hierárquico chamada amostragem negativa. Uma limitação inerente das representações de palavras, naquele momento, era sua indiferença à ordem

das palavras e sua incapacidade de representar frases idiomáticas. Por exemplo, os significados de "Canada" e "Air" não podiam ser facilmente combinados para obter "Air Canada". Motivados por esse exemplo, foi apresentado um método simples para encontrar frases no texto e indicaram que é possível aprender boas representações vetoriais para milhões de frases.

[58] explorou um simples e eficiente baseline para classificação de texto. Os experimentos indicaram que o classificador de texto rápido proposto, 'FastText', geralmente está no mesmo nível dos classificadores de aprendizado profundo em termos de precisão e muitas ordens de magnitude mais rápido para treinamento e avaliação. Foi indicado que o 'FastText' poderia ser treinado em mais de um bilhão de palavras em menos de dez minutos usando uma CPU multicore padrão e classificar meio milhão de frases entre 312 mil classes em menos de um minuto.

[119] apresentou o 'Transformer'. No estudo foi proposta uma nova arquitetura de rede simples, o 'Transformer', baseada exclusivamente em mecanismos de atenção, dispensando recorrência e convoluções inteiramente. Os experimentos em duas tarefas de tradução automática mostraram que esses modelos são superiores em qualidade, sendo mais paralelizáveis e exigindo significativamente menos tempo para treinar. O modelo atingiu 28,4 Bilingual Evaluation Understudy Score (BLEU) na tarefa de tradução de inglês para alemão do WMT 2014, melhorando os melhores resultados existentes, incluindo ensembles, em mais de 2 BLEU. Na tarefa de tradução de inglês para francês do WMT 2014, o modelo proposto estabeleceu uma nova pontuação BLEU de última geração de modelo único de 41,8 após treinamento por 3,5 dias em oito GPUs, uma pequena fração dos custos de treinamento dos melhores modelos da literatura. O estudo mostrou que o 'Transformer' generaliza bem para outras tarefas, analisando tanto com dados de treinamento grandes quanto limitados.

[92], estudo do ELMo, apresentou um novo tipo de representação de palavras profundamente contextualizado que modela (1) características complexas do uso de palavras (por exemplo, sintaxe e semântica) e (2) como esses usos variam entre contextos linguísticos (ou seja, para modelar a polissemia). Os vetores de palavras apresentados são funções aprendidas dos estados internos de um modelo de linguagem bidirecional profundo, que é pré-treinado com um grande corpus de texto. O estudo mostrou que essas representações podem ser facilmente adicionadas aos modelos existentes e melhorar significativamente o estado da arte em seis problemas desafiadores do NLP, incluindo resposta a perguntas, vinculação textual e análise de sentimentos. Também apresentaram uma análise mostrando que expor as partes internas profundas da rede pré-treinada é importante, pois permite que os modelos *downstream* misturem diferentes tipos de sinais de semi-supervisão.

[97] apresentou o GPT, *Generative Pre-trained Transformer*. Foi indicado que ganhos em diferentes tarefas no NLP podem ser obtidos por pré-treinamento generativo de um modelo de linguagem em um corpus diversificado de texto não rotulado, seguido de ajuste fino discriminativo em cada tarefa específica. Em contraste com as abordagens anteriores, fizeram uso de transformações de

entrada com reconhecimento de tarefas durante o ajuste fino para obter uma transferência eficaz, exigindo alterações mínimas na arquitetura do modelo. Foi indicada a eficácia da abordagem em uma ampla gama de *benchmarks* para a compreensão da linguagem natural. O modelo geral proposto, agnóstico de tarefa, supera os modelos treinados discriminativamente que usam arquiteturas criadas especificamente para cada tarefa, melhorando significativamente o estado da arte em 9 das 12 tarefas estudadas. Por exemplo, foram alcançadas melhorias absolutas de 8,9% no raciocínio de senso comum (Stories Cloze Test), 5,7% na resposta a perguntas (RACE) e 1,5% na vinculação textual (MultiNLI).

O GPT é uma arquitetura baseada em 'Transformer' e procedimentos de treinamento para tarefas no NLP. O treinamento possui duas etapas bem definidas; primeiro, um objetivo de modelagem de linguagem é usado nos dados não rotulados para aprender os parâmetros iniciais de um modelo de rede neural. Em seguida esses parâmetros são adaptados a uma tarefa alvo usando o objetivo supervisionado correspondente.

[27] apresentou um novo modelo de representação de linguagem chamado BERT. Diferentemente dos modelos de representação de linguagem vistos até aquele momento [73; 96], o BERT foi projetado para pré-treinar representações bidirecionais profundas de texto não rotulado, condicionando conjuntamente o contexto esquerdo e direito em todas as camadas. Como resultado, o modelo BERT pré-treinado pode ser ajustado com apenas uma camada de saída adicional para criar modelos de última geração para uma ampla gama de tarefas, como resposta a perguntas e inferência de linguagem, sem modificações substanciais na arquitetura específica da tarefa. O BERT é conceitualmente simples e empiricamente poderoso. Ele obteve resultados de última geração em onze tarefas de processamento de linguagem natural, incluindo elevar a pontuação GLUE para 80,5% (7,7% de melhoria absoluta), precisão MultiNLI para 86,7% (4,6% de melhoria absoluta), SQuAD v1.1 pergunta respondendo Teste F1 a 93,2 (melhoria absoluta de 1,5 pontos) e Teste SQuAD v2.0 F1 a 83,1 (melhoria absoluta de 5,1 pontos).

[125] apresentou o XLNet. É um método generalizado de pré-treinamento auto-regressivo que permite aprender contextos bidirecionais, maximizando a probabilidade esperada sobre todas as permutações da ordem de fatoração, e supera as limitações do BERT (que negligencia a dependência entre as posições mascaradas e sofre de uma discrepância no ajuste de pré-treinamento) graças à sua capacidade de formulação auto-regressiva. O XLNet integra ideias do Transformer-XL, modelo auto-regressivo de última geração, no pré-treinamento. Empiricamente, em configurações de experimento comparáveis, foi observado que o XLNet supera o BERT em 20 tarefas, geralmente por uma grande margem, incluindo resposta a perguntas, inferência de linguagem natural, análise de sentimentos e classificação de documentos.

Não enquadrado em nenhuma das três segmentações expostas nesse capítulo (i.e., comparação

direta entre métodos de representação textual, aplicação de um dos métodos a um caso concreto ou proposta de evolução dessas representações), [7] realizou pesquisa explorando os avanços passados e recentes na classificação semântica de texto e organiza as abordagens existentes em cinco categorias fundamentais; abordagens baseadas em conhecimento de domínio, abordagens baseadas em corpus, abordagens baseadas em aprendizagem profunda, abordagens aprimoradas de sequência de palavras/caracteres e abordagens linguísticas enriquecidas. Além disso, ele destaca as vantagens dos algoritmos de classificação semântica de texto (mais recentes, baseados em aprendizagem profunda) sobre os algoritmos tradicionais de classificação de texto. No entanto foi observado que determinar o tipo de abordagem é uma tarefa desafiadora que depende da disponibilidade e do tamanho do conjunto de dados e das bases de conhecimento, e da natureza do problema que está sendo investigado. A escolha do método de classificação de texto semântico adequado depende de uma série de fatores inter-relacionados, pois cada categoria de método tem certos benefícios e malefícios quando comparados a outros.

A Tabela III.1 apresenta um resumo dos trabalhos indicando o Tópico de Pesquisa, Algoritmos/Abordagem Conceitual e Métricas de Avaliação utilizadas.

O presente estudo está comparando diferentes abordagens na representação de palavras/documentos e seus usos em diferentes algoritmos de aprendizagem de máquina. Essas representações e algoritmos são utilizados numa tarefa de classificação de polaridade, opcionalmente, binária e com conjuntos de dados variados. Algoritmos de aprendizagem de máquina tradicionais estão sendo utilizados com BOW usando TF-IDF e LSA, enquanto os algoritmos de aprendizagem profunda estão sendo utilizados com WE e CWE.

Estudo	Tópico de Pesquisa	Algoritmos / AC	Medidas de Avaliação
[8]	Classificação de Documento	kNN, MNB, SVM, MLP e OCSVM comparação BOW e Embeddings	F1, mae, mse e r2
[112]	Recuperação da Informação	BM25, DFR e LMD BERT, RoBERTa e XLNet combinação BOW e Embeddings	Precisão, NDCG, mean average precision (MAP) e binary preference (Bpref)
[90]	Aperfeiçoamento da representação textual / algorítmico	Reg. Log e SVM LSI e LDA DM, DBOW e semi-DBOW SSLE	acurácia
[25]	Recuperação da Informação	BERT comparação BOW e Embeddings Contextuais	acurácia
[43]	Aperfeiçoamento da representação textual / algorítmico	LSTM-LM combinação modelo de tópicos (BOW) e MLN (Embeddings)	IR-precisão e F1
[7]	Classificação de Documento	Survey	Survey
[6]	Atribuição de autoria	Reg Log, SVM, Naive Bayes e Mult Percept. comparação de BOW c/ document embedding	média do F1-Score
[48]	Classificação de Documento	LSTM comparação BOW e Embeddings	taxa de erro
[53]	Reconhecimento de Entidade Nomeada	MLN e Embeddings	F1
[49]	Aperfeiçoamento da representação textual / algorítmico	NB, SVM, RNN, CNN, BOW, variações e o modelo proposto	acurácia
[40]	Recuperação da Informação e aperfeiçoamento algorítmico	GRU (DeepApi), Lucene + UP-Miner, SWIM e outros	acurácia, BLEU score e outras
[126]	Recuperação da Informação	NTLM-skipgram, NTLM-CBOW, Dirichlet LM, TLM-MI e variações	MAP, P@10 e outras
[114]	Aperfeiçoamento da representação textual / algorítmico	PENN e DIEM	acurácia e outras
[77]	Aperfeiçoamento da representação textual / algorítmico	RNNLM, NNLM, Proposição: CBOW e CSkip-gram	acurácia e outras
[79]	Aperfeiçoamento da representação textual / algorítmico	Negative Sampling e Skip-gram	acurácia e outras
[58]	Aperfeiçoamento da representação textual / algorítmico	n-grams, VDCNN, char-CNN e outros Proposição: FastText	acurácia e outras
[119]	Aperfeiçoamento da representação textual / algorítmico	ByteNet, ConvS2S, Deep-Att e outros Proposição: Transformer	acurácia, BLEU score e outras
[92]	Aperfeiçoamento da representação textual / algorítmico	biLM e Cove Proposição: ELMo	acurácia, F1, e outras
[97]	Aperfeiçoamento da representação textual / algorítmico	ESIM + ELMo, GenSen, BiLSTM + ELMo e outros Proposição: GPT	correlação de Mathews, de Pearson, acurácia e F1
[27]	Aperfeiçoamento da representação textual / algorítmico	BiLSTM+ELMo+Attn, GPT Proposição: BERT	acurácia, correlação de Spearman e F1
[125]	Aperfeiçoamento da representação textual / algorítmico	GPT, BERT CNN, ULMFiT Proposição: XLNet	acurácia, F1 e outras

Tabela III.1: Resumo dos estudos

Capítulo IV Metodologia

Este capítulo visa apresentar a proposta desta dissertação. O objetivo é o de comparar o desempenho preditivo de modelos *n-gram* (os classificadores que fazem parte do grupo MAMT) com os MLN, utilizando, para representação de palavras/documentos, o BOW com TF-IDF e com LSA nos MAMT, e os WE e os CWE nos MLN. É possível utilizar *embeddings* nos MAMT, mas a opção nesta pesquisa foi a de utilizar as representações de palavras/documentos mais recentes para os MLN, baseados em redes neurais profundas, também mais recentes. Foram consideradas, nessa opção, as restrições no ambiente de execução indicadas na Seção VI.1 e, também, a possibilidade de observar/avaliar essas formas de representação e algoritmos sob uma perspectiva cronológica dos momentos em que foram propostos. Nos parágrafos seguintes o racional e o fluxo de trabalho utilizados para a comparação dessas abordagens são apresentados.

Inicialmente os conjuntos de dados passam pela fase de pré-processamento, fase em que os exemplos textuais são preparados para o adequado processamento pelos grupos de algoritmos utilizados. No pré-processamento são removidos marcadores e formatação (pontuação, acentos e conversão para minúsculo), ocorre a formação de *tokens* e a filtragem.

Após o pré-processamento nós temos duas fases. Na primeira, os exemplos textuais são convertidos numa representação numérica/vetorial para que possam ser utilizados pelos algoritmos. No caso do grupo MAMT, os exemplos são convertidos no BOW com o TF-IDF ou no BOW com LSA. No BOW com o TF-IDF as relações semânticas entre os termos não são preservadas. No BOW com LSA nós temos uma representação implícita da semântica do texto com base na co-ocorrência observada dos termos. Para o grupo MLN, os exemplos textuais são convertidos em WE e CWE. Por meio dos *embeddings* as distribuições dos termos nos documentos (dos observados durante o treinamento ou daqueles que serviram de base para os *embeddings* pré-treinados) são consideradas para a constituição semântica dos mesmos. Nesse caso, e também no BOW com LSA, temos como base a hipótese distribucional, onde há a consideração de que palavras que ocorrem em contextos similares tendem a ter significados similares [72; 117] (ver Seção II.4).

Na segunda fase os exemplos, já convertidos, são utilizados, em cada caso, pelos algoritmos do grupo MAMT ou pelos algoritmos do grupo MLN para treinamento e teste. Os desempenhos preditivos dos modelos de cada grupo são avaliados por meio de medidas de avaliação, sendo seus resultados registrados para observação posterior.

A Figura IV.1 ilustra o esquema:

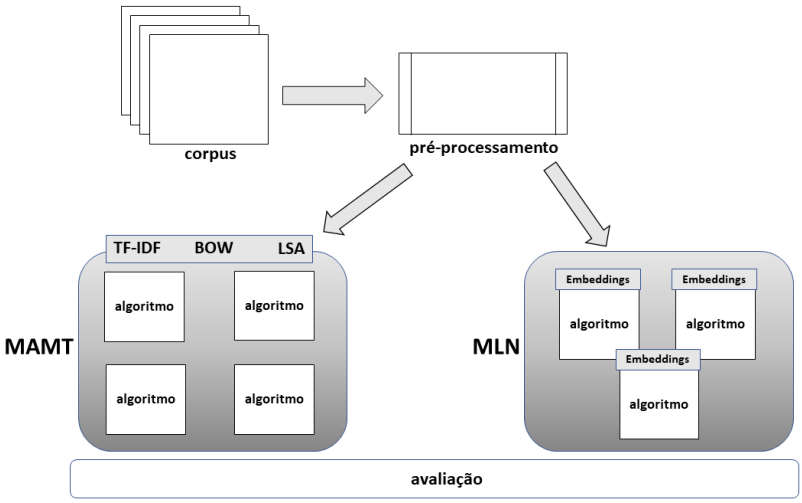

Figura IV.1: Esquema de comparação dos métodos de representação de palavras. Fonte: Elaboração Própria.

Capítulo V Análise Experimental

Este capítulo discorre sobre os experimentos realizados nesta dissertação. A Seção V.1 mostra a abordagem escolhida para aplicar a metodologia proposta para a comparação do desempenho preditivo dos algoritmos e dos métodos de representação de palavras/documentos utilizados. São indicados os algoritmos, seus hiperparâmetros, ambiente de execução, tecnologias e o fluxo de trabalho. A Seção V.2 apresenta os conjuntos de dados usados na pesquisa, suas características e particularidades, além das etapas realizadas no pré-processamento para uso no trabalho. A Seção V.3 apresenta os resultados obtidos com a abordagem e os conjuntos de dados, além de discuti-los.

V.1 Instância de Solução

O experimento foi realizado no *Google Colab* (https://colab.research.google.com/) usando o tensorflow (https://www.tensorflow.org/) [1], scikit-learn (https://scikit-learn.org/stable/) [91] e nltk (https://www.nltk.org/) com o python (https://www.python.org/). No *Google Colab*, na execução dos experimentos, o modelo de GPU mais utilizado foi o *Tesla V100* (ver Tabela V.3). O modelo da placa é escolhido pelo próprio *Google Colab* no momento em que a máquina é instanciada. Os modelos de linguagem *n-gram* do grupo MAMT são baseados nos algoritmos KNN, MNB, SVM e RF. Os modelos utilizados no MLN são o BERT e a Bi-LSTM. Os modelos dos dois grupos são utilizados numa tarefa de classificação de polaridade binária com quatro conjuntos de dados.

Essa pesquisa não tem como objetivo o ajuste fino dos hiperparâmetros em cada algoritmo, uma vez que esse tipo de busca exigiria - se realizada com a devida diligência - um processo de testagem e fluxo de trabalho próprio, o que normalmente resulta em estudos específicos. O *Hyperparameter optimization* é o problema de escolher um conjunto de hiperparâmetros ideais para um algoritmo de aprendizado; avaliar o desempenho de um modelo treinado com um conjunto específico de hiperparâmetros do espaço de busca depende dos recursos computacionais disponíveis, da natureza do algoritmo de aprendizagem, do tamanho do problema e a avaliação exige um tempo considerável, não sendo uma tarefa trivial [35].

Apesar disso, evidentemente, foram buscados estudos que pudessem servir de referência a esse tipo de escolha e utilizados conjuntos de hiperparâmetros em algoritmos como o GS e o HGS para a identificação dos melhores *setups* de uso. Essa foi a abordagem utilizada com os algoritmos do grupo

MAMT. Com os algoritmos pertencentes ao MLN a escolha dos hiperparâmetros foi baseada nos estudos de origem dos mesmos e de estudos que se propuseram a fazer essa avaliação. As subseções abaixo indicam os tipos de processamentos, adotados para esses algoritmos, que foram utilizados com os conjuntos de dados deste trabalho, além de indicar os hiperparâmetros e variáveis de uso comum utilizados em cada grupo.

V.1.1 Tipos de abordagens

Foram utilizados quatro conjuntos de dados: Buscapé, Polarity, Stanford Sentiment Treebank v2 (SST2) e YAR. Esses conjuntos são descritos na Seção V.2. Em razão das limitações de recursos disponíveis e da homogeneidade dos exemplos dentro de cada conjunto, observada com execuções preliminares e que indicaram desvios padrões muito baixos nas medidas de avaliação utilizadas, foi adotada estratégia de treinamento e teste dos algoritmos baseada na divisão dos conjuntos em 75% dos exemplos para treino (sendo usados, no MAMT, também para avaliação e escolha dos melhores hiperparâmetros) e 25% para teste. Após a separação, os mesmos segmentos (75% treino e 25% teste) de cada conjunto foram usados em todos os algoritmos dos grupos MAMT e MLN. Com os algoritmos do MLN os dados destinados ao treinamento foram novamente divididos entre treino e validação, sendo 75% para treino e 25% para validação. Nos algoritmos do MAMT, durante a etapa de treinamento, foram utilizados o HGS ou o GS com validação cruzada k-fold, conforme Seção II.7. Os detalhes de cada grupo estão especificados nas subseções próprias.

Os exemplos textuais dos conjuntos foram convertidos, para uso nos algoritmos do MAMT, no TF-IDF e no LSA tanto com *1-gram* quanto com *2-gram*, em cada caso. Na vetorização dos exemplos dos conjuntos para o TF-IDF foram removidas apenas as *stop words* e estabelecido o *ngram range*, assumindo valor (1, 1) ou (2, 2). Não foram utilizadas nenhuma restrição de termos do vocabulário baseada na frequência dos mesmos, sendo o vocabulário determinado a partir dos exemplos do conjunto de treinamento em cada processamento (conjunto de dados utilizado). Para o LSA foi utilizado o algoritmo de decomposição de valor singular truncada, útil na redução de dimensionalidade e que lida bem com matrizes de contagem de termos/TF-IDF, podendo trabalhar com matrizes esparsas de forma eficiente. Foi utilizado $ncomponents = 100$, conforme recomendação do *scikitlearn* para o LSA.

Para utilização nos algoritmos do MLN os exemplos textuais dos conjuntos foram convertidos no CWE (pré-treinados) ou no WE (codificados durante o treinamento), respectivamente, para utilização nos dois modelos BERT e na Bi-LSTM. A Tabela V.1 resume os métodos de representação textual e os seus usos nos algoritmos.

Os conjuntos de dados foram processados separadamente. Antes do processamento (utilização pelos algoritmos), os conjuntos de dados passam pelo pré-processamento e pela conversão numa

Algoritmo	TF-IDF 1-gram	TF-IDF 2-gram	LSA 1-gram	LSA 2-gram	WE	CWE
KNN	X	X	X	X		
MNB	X	X	X	X		
SVM	X	X	X	X		
RF	X	X	X	X		
BERT Small						X
BERT Base						X
Bi-LSTM					X	

Tabela V.1: Resumo dos métodos de representação textual utilizados

representação numérica adequada ao uso pelos algoritmos. Nessa etapa os textos dos exemplos são convertidos para o BOW com o TF-IDF e são utilizados no MAMT. O fluxo de execução é repetido, fazendo a conversão, dessa vez, ao BOW com o TF-IDF e, em seguida, para o LSA, sendo o conjunto, novamente, utilizado no MAMT. O fluxo de execução é repetido e os exemplos textuais do conjunto, agora, são endereçados aos algoritmos do MLN. Os textos são convertidos em *Embeddings* e utilizados nesses modelos. Os conjuntos de dados, após as conversões, em cada caso, são usados no grupo MAMT e no grupo MLN numa tarefa de classificação de polaridade binária. Os três resultados, MAMT com TF-IDF, MAMT com LSA e MLN com *Embeddings* são registrados para avaliação posterior.

A Figura V.1 ilustra o esquema:

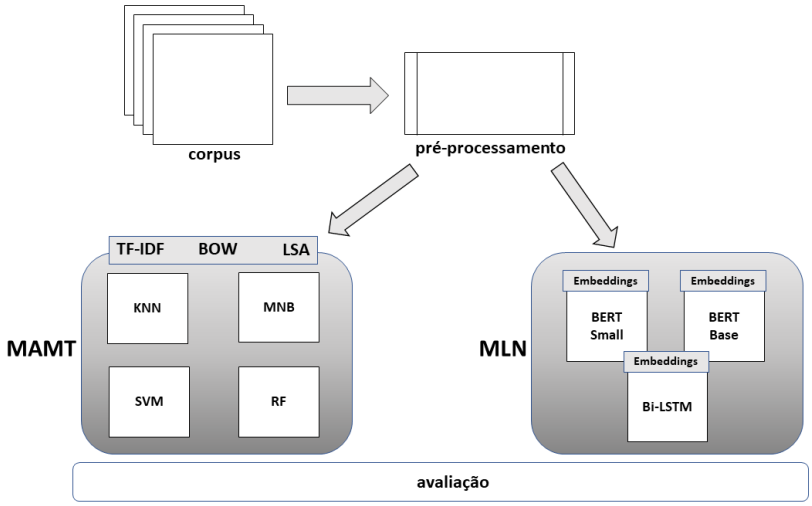

Figura V.1: Esquema de comparação dos métodos de representação de palavras (instância de solução). Fonte: Elaboração Própria.

Resumidamente, na abordagem nós temos o MAMT que utiliza os algoritmos KNN, MNB, SVM e RF, com o BOW usando o TF-IDF. Usando o BOW com TF-IDF não há a consideração de que as ordens dos termos são relevantes na constituição semântica dos documentos. Os mesmos algoritmos do MAMT foram utilizados com o LSA também, e, nesse caso, ao extrair tópicos dos documentos, há a presunção de que palavras com significados similares ocorrerão em contextos si-

milares (hipótese distribucional). O mesmo ocorre com o MLN, composto pela Bi-LSTM e pelos dois BERTs, utilizando WE e CWE, dado que os *embeddings* também se apoiam na hipótese distribucional e entendem que palavras que ocorrem em contextos similares tendem a ter significados similares [72; 117]. Todos os algoritmos e abordagens são avaliados numa tarefa de classificação de polaridade binária, por opção.

V.1.2 MAMT

Para escolha dos hiperparâmetros em cada algoritmo do grupo foi utilizado o HGS com os conjuntos de dados maiores (i.e., SST2 e YAR), e o GS com os conjuntos menores (i.e., Buscapé e Polarity). Essa opção teve por base a forma como esses algoritmos, o HGS e o GS, lidam com o espaço de busca, os consequentes impactos observados no tempo de execução e as restrições verificadas no uso do *Google Colab*.

Os hiperparâmetros utilizados tanto no HGS quanto no GS para o KNN foram k variando de 5 a 15 e o peso atribuído aos vizinhos próximos variando entre *uniform* e *distance*. Com o MNB foram utilizados o *alpha* com valor fixo "1" e o parâmetro relacionado a aprendizagem de probabilidades anteriores de classe, o *fit prior*, variando entre *False* e *True*. Foram utilizados o *kernel* variando entre [*linear*, *rbf* e *sigmoid*] e o C entre os valores [1, 10, 15, 20] para o SVM. Por fim, com o RF foram utilizados o número de estimadores variando entre os valores [75, 100, 125] e o *max samples* entre [0.25, 0.50, 0.75]. A Tabela V.2 resume os hiperparâmetros utilizados em cada algoritmo.

Algoritmo	Hiperparâmetros testados
KNN	k=[5 a 15] e weights=[uniform, distance]
MNB	alpha=1 e fit prior=[False, True]
SVM	kernel=[linear, rbf e sigmoid] e C=[1, 10, 15, 20]
RF	n estimators=[75, 100, 125] e max samples=[0.25, 0.50, 0.75]

Tabela V.2: Resumo dos hiperparâmetros testados

Com o HGS foi utilizado *fator* 20 para o KNN, 2 para o MNB, 12 para o SVM e 9 para o RF. Esses valores foram definidos considerando o espaço de busca de cada algoritmo durante o treinamento e com o objetivo de induzir o HGS a utilizar todos os exemplos do conjunto de treinamento apenas na última iteração, com o conjunto de hiperparâmetros vencedor. Essa escolha foi necessária para adequar a execução dos experimentos às limitações do *Google Colab*, explicadas no Capítulo VI, onde as limitações do estudo estão descritas.

Foram considerados, na tomada de decisão em relação ao espaço de busca dos hiperparâmetros, os valores padrão dos parâmetros das funções implementadas no *scikit-learn* [91] e os estudos [34; 17; 62; 106; 22]. Os mesmos hiperparâmetros foram testados nos quatro conjuntos, tanto na utilização do HGS com os conjuntos maiores (i.e., YAR e SST2), quanto na utilização do GS com os conjuntos

menores (i.e., Buscapé e Polarity).

Os algoritmos do grupo MAMT tiveram seus hiperparâmetros escolhidos e treinados, nos conjuntos maiores, YAR e SST2, utilizando validação cruzada estratificada com $k = 2$ no HGS. Os mesmos algoritmos do grupo MAMT tiveram seus hiperparâmetros escolhidos e treinados, nos conjuntos menores, Buscapé e Polarity, utilizando a mesma validação cruzada estratificada com $k = 5$ no GS. Essas escolhas buscaram garantir, assim, ao menos 20% dos exemplos em cada divisão para validação na etapa de treinamento, além de contornar limitações no tempo de execução do *Google Colab*. Conforme já mencionado, essa etapa utiliza 75% dos dados disponíveis, restando os outros 25% como conjunto de teste e aferição das medidas de desempenho preditivo.

Para avaliar o desempenho dos modelos na validação cruzada, tanto no HGS quanto no GS, foi utilizado o *f1 weighted*. Essa medida é uma média ponderada pelo suporte (o número de instâncias verdadeiras de cada rótulo) entre os rótulos. Desse modo o desequilíbrio dos rótulos é considerado.

V.1.3 MLN

Os modelos do grupo MLN são a Bi-LSTM, o *BERT Small* e o *BERT Base*. A Bi-LSTM possui uma camada de codificação, que faz o pré-processamento, camada de *Embeddings*, camada Bidirecional LSTM, uma camada densa com função de ativação *Relu* e uma última camada densa com função de ativação linear. Nesse trabalho não foi adotada a utilização de *embeddings* pré-treinados na Bi-LSTM, sendo os mesmos criados por meio dos dados que foram utilizados na rede durante o processamento de cada conjunto. Esses *embeddings* criados pelo *TensorFlow* possuem os relacionamentos identificados entre as palavras que foram fornecidas à rede e a qualidade dessas representações, evidentemente, irá variar de acordo com a quantidade de exemplos de cada conjunto e a qualidade dos mesmos.

O *BERT Small* (i.e., *bert en uncased L-4 H-512 A-8'* [116; 1]), está estruturado com uma camada de entrada, camada de pré-processamento, camada de codificação, camada de *dropout* e uma camada densa com função de ativação linear. Esse modelo utiliza *embeddings* pré-treinados em inglês na *Wikipedia* e *BooksCorpus* [116; 1]. Na criação dos *embeddings* as entradas de texto foram normalizadas de modo *"uncased"*, tornando-as minúsculas antes da *tokenização* e todos os marcadores de acento foram removidos, conforme descrição contida em [116; 1].

O *BERT Base* (i.e., *BERT Base cased L-12 H-768 A-12* [27; 1]), está estruturado com uma camada de entrada, camada de pré-processamento, a camada de codificação, camada de dropout e uma camada densa com função de ativação linear. Esse modelo utiliza *embeddings* pré-treinados para vários idiomas na *Wikipédia multilíngue* [27; 1]. Na criação dos *embeddings* as entradas de texto foram normalizadas de modo *"cased"*, o que significa que a distinção entre letras maiúsculas e minúsculas, bem como marcadores de acento, foi preservada, conforme descrição contida em [27; 1].

Considerando o que foi mencionado na Seção II.5.6 sobre a limitação quantitativa de *tokens* nas entradas (cada um dos exemplos textuais dos conjuntos), é importante esclarecer que não foi adotada nenhuma estratégia de escolha de *tokens* nos exemplos dos conjuntos processados. Essa escolha baseia-se nas características observadas nos conjuntos, conforme descrição feita na Seção V.2. Tanto no *BERT Small* quanto no *BERT Base* a camada de codificação foi utilizada com $trainable = True$, permitindo o ajuste dos *embeddings* durante o treinamento com o processamento das entradas. Foi verificado que o desempenho preditivo desses modelos era melhor com essa opção.

Foram utilizados nos treinamentos desses algoritmos um *lote*, *época* e taxa de aprendizado comuns entre a Bi-LSTM, *BERT Small* e o *BERT Base*; $lote = 32$, $época = 2$ nos conjuntos maiores e a taxa de aprendizado $= 3e - 5$. Nos conjuntos menores, Buscapé e Polarity, foram utilizadas 12 épocas no treinamento em razão da menor quantidade de exemplos. A definição desses hiperparâmetros utilizou como referência os estudos [27], [116], [111] e [92]. Foram utilizados, também, o *AdamW* como algoritmo de otimização e a *Binary Cross Entropy* como função de custo tanto na Bi-LSTM quanto nos dois BERTs. O intuito, evidentemente, é o de observar o desempenho preditivo desses modelos tendo uma base comum.

A Tabela V.3 indica o número de vezes que cada modelo de GPU foi utilizado pelos MLN no *Colab*. Elas estão ordenadas pela capacidade de desempenho computacional. A indicação do tipo de GPU, em cada conjunto, também está nas Tabelas V.7 e V.8, da Bi-LSTM e dos modelos BERT, respectivamente.

modelo	qtd
TESLA T4	6
TESLA P100	3
TESLA V100	**18**
TESLA A100	6

Tabela V.3: Quantitativos dos tipos de GPUs usadas nas execuções no Colab

V.2 Conjuntos de Dados

Nesta seção são apresentados os conjuntos de dados utilizados no estudo. Foram utilizados diferentes conjuntos de dados com o intuito de observar o desempenho preditivo dos algoritmos com as diferentes técnicas de representação textual. Nos parágrafos seguintes há uma breve descrição dos quatro conjuntos: Buscapé, Polarity, SST2 e YAR; são indicadas a origem, o processo de rotulagem e características. Os exemplos dos conjuntos foram preparados para serem utilizados numa tarefa de classificação de polaridade binária - rotulados como 'positivo' ou 'negativo' - sendo isso uma opção adotada na utilização dos mesmos. No final desta seção a Tabela V.4 os apresenta de forma consolidada.

O conjunto de dados Buscapé foi criado a partir de avaliações em português de usuários do site brasileiro (https://www.buscape.com.br/) em [61], sendo um conjunto arbitrariamente pequeno. Os avaliações possuem estrelas, representando o nível de satisfação com o produto, associadas a cada texto. As mesmas variam de 0 a 5 e foram utilizadas para atribuição de rótulos aos exemplos. Comentários com 0 a 2 estrelas receberam rótulo 'negativo', e comentários com 3 a 5 estrelas receberam rótulo 'positivo'.

O conjunto possui 230 exemplos rotulados como 'negativos' e 1.508 exemplos rotulados como 'positivos', totalizando 1.738 exemplos. O que corresponde, respectivamente, a 13% de exemplos negativos e 86% de positivos. Temos, assim, um conjunto pequeno e desbalanceado.

O segundo conjunto de dados, o Polarity, é composto por exemplos em inglês do site (http://reviews.imdb.com/Reviews). No site são encontradas classificações e críticas para filmes e programas de TV. O conjunto estava rotulado e foi usado pela primeira vez em [87]. O conjunto é pequeno e balanceado, tendo 1.000 exemplos 'positivos' e 1.000 exemplos 'negativos', totalizando 2.000 exemplos.

O terceiro conjunto de dados, o SST2 [109], é composto por exemplos em inglês, frases simples, extraídas de críticas de filmes. O conjunto possui, indicando sentimento associado a cada um dos exemplos, uma avaliação do usuário variando de 0.0 à 1.0. Os exemplos foram separados em 3 grupos, sendo consideradas 'negativos' os exemplos com notas menores ou iguais a 0.4, 'neutros' os que eram maiores que 0.4 e menores que 0.6 e, como 'positivos', os exemplos com notas maiores ou iguais a 0.6. Os exemplos 'neutros', totalizando 111.725, foram removidos. Foram removidas, também, as avaliações com texto inferior a 3 caracteres. O conjunto ficou com 69.607 (54,60%) exemplos 'positivos' e 57.875 (45,39%) 'negativos', totalizando 127.482 exemplos.

O quarto conjunto de dados, o YAR, foi criado e disponibilizado no site (https://www.yelp.com /dataset) a partir de avaliações de usuários sobre negócios. O conjunto original possui 8.635.403 exemplos que, além do texto e de outros dados, possuem uma avaliação do usuário que varia de 1 a 5.

Foram utilizados oito subconjuntos, aqui chamados de versões, desse conjunto na pesquisa. Em todas as versões os exemplos foram separados em 2 grupos, sendo considerados 'negativos' os exemplos com nota 1 e 2, e 'positivos' os exemplos com nota 4 e 5. Os exemplos com nota 3, considerados 'neutros', foram excluídos. Também foi mantida a representação percentual das classes observadas no conjunto original, aproximadamente 74,5% de exemplos 'positivos' e 25,5% de exemplos 'negativos' em cada versão.

A principal versão, o Yelp-1M, possui 1.000.871 exemplos. Essa quantidade considera o objetivo do estudo e as limitações observadas no *Google Colab*, descritas no Capítulo VI. São 1.000.871 exemplos, sendo 748.456 exemplos 'positivos' e 252.415 exemplos 'negativos'. O segundo conjunto,

o Yelp-750, possui 752.446 exemplos, sendo 559.748 ‘positivos’ e 192.698 ‘negativos’. O terceiro conjunto, o Yelp-500, possui 502.127 exemplos, sendo 383.388 ‘positivos’ e 118.739 ‘negativos’. O quarto conjunto, o Yelp-250, possui 251.164 exemplos, sendo 192.876 ‘positivos’ e 58.288 ‘negativos’.

Temos, então, o Yelp-1M com 1.000.871 exemplos, o Yelp-750 com 752.446 exemplos, o Yelp-500 com 502.127 exemplos e o Yelp-250 com 251.164 exemplos. Além dessas quatro versões, foram utilizadas outras quatro, mas com representações balanceadas entre as classes. O Yelp-1M-B possui 1.001.496 exemplos, sendo 500.748 exemplos ‘positivos’ e 500.748 exemplos ‘negativos’. O Yelp-750-B possui 751.686 exemplos, sendo 375.843 ‘positivos’ e 375.843 ‘negativos’. O Yelp-500-B possui 500.034 exemplos, sendo 250.017 ‘positivos’ e 250.017 ‘negativos’. O Yelp-250-B possui 250.796 exemplos, sendo 125.398 ‘positivos’ e 125.398 ‘negativos’.

Dois dos quatro conjuntos de dados, o YAR e o Polarity, possuem exemplos textuais compostos por muitas frases, textos longos, sendo os outros formados por exemplos compostos por frases simples, textos curtos. Foi avaliada a necessidade da utilização de alguma estratégia de adequação do tamanho dos exemplos desses conjuntos ao BERT. No YAR, maior conjunto do estudo, foi observado que apenas 1% dos exemplos do conjunto original (91.815 de 8.635.403) tinham mais de 510 *tokens*. No Polarity, conjunto balanceado e pequeno com 2.000 exemplos, foi observado que 1.543 exemplos (aproximadamente 77% do total) possuem mais de 510 tokens. Considerando que estamos utilizando oito subconjuntos do YAR, cujo conjunto original possui apenas 1% dos exemplos com mais de 510 *tokens*, que o Buscape e o SST2 não possuem exemplos acima dessa quantidade, e que apenas o Polarity possui 77% de seus exemplos com mais de 510 *tokens*, sendo um dos dois conjuntos pequenos utilizados no estudo, não foi adotada nenhuma estratégia própria para o truncamento dessas entradas. Não há impacto - nesse contexto (11 conjuntos de dados; 8 versõs do YAR, SST2, Polarity e Buscape) - em permitir que o próprio BERT faça o truncamento das entradas durante o processamento, tal como indicado na Seção II.5.6.

Foram utilizados, no total, onze conjuntos de dados: oito versões do YAR - sendo quatro desbalanceadas (como o conjunto original) e quatro versões balanceadas - o SST2, o Polarity e o Buscapé.

V.2.1 Pré-processamento dos conjuntos

Todos os conjuntos passaram pela etapa de pré-processamento. Após o carregamento e pré-processamento, os conjuntos possuem apenas as colunas ‘text’, com o texto dos exemplos e ‘y’, com o rótulo associado a cada exemplo (de acordo com os critérios explicados nas descrições dos conjuntos). Na fase de pré-processamento, em nível léxico, ocorreu a remoção de marcadores e formatação, sendo retirados sinais de pontuação, acentos e realizada a conversão para minúsculo.

Para os algoritmos do grupo MAMT os registros foram *tokenizados* e, nessa etapa, foi realizada a filtragem com a remoção de palavras funcionais (*stop words*). Como mencionado na Subseção

conjunto	exemplos positivos	exemplos negativos	total
Buscapé	1.508	230	1.738
Polarity	1.000	1.000	2.000
SST2	69.607	57.875	127.482
Yelp-250	192.876	58.288	251.164
Yelp-250-B	125.398	125.398	250.796
Yelp-500	383.388	118.739	502.127
Yelp-500-B	250.017	250.017	500.034
Yelp-750	559.748	192.698	752.446
Yelp-750-B	375.843	375.843	751.686
Yelp-1M	748.456	252.415	1.000.871
Yelp-1M-B	500.748	500.748	1.001.496

Tabela V.4: Conjuntos de dados e suas quantidades

V.1.1, a *tokenização* usando o TF-IDF foi feita sem qualquer restrição vocabular impositiva. Nas execuções em que a vetorização utilizou o LSA, como descrito na Subseção V.1.1, foi utilizado $ncomponents = 100$. Nesse caso, os exemplos, após a vetorização com o TF-IDF, passam por novo processamento usando o algoritmo de decomposição de valor singular truncada para, assim, termos o LSA. Todas as entradas textuais (cada um dos exemplos de cada conjunto) foram convertidas em caracteres minúsculos. Para os algoritmos do grupo MLN, os BERTs e a Bi-LSTM, os exemplos dos conjuntos passaram pelas mesmas etapas descritas acima, exceto pela etapa de *tokenização* e remoção de *stop words*.

V.3 Resultados e Discussão

Para comparar diferentes representações do dado textual e seus usos em diferentes algoritmos numa tarefa de classificação de polaridade binária, foram utilizados o BOW com TF-IDF e com LSA no MAMT, e os WE e CWE no MLN. O MAMT é composto pelos algoritmos KNN, MNB, SVM e RF. O MLN é composto pela Bi-LSTM e duas versões do BERT.

As formas de representação do dado textual mais recentes, também mais robustas, o WE e o CWE, apresentam evidente ganho na representação numérica de termos linguísticos, como dito na Subseção II.4.3. No entanto, é interessante entender em que medida o uso dessas formas de representação e dos algoritmos mais recentes (usados em tarefas de classificação, já como modelos treinados), de fato, representam a melhor escolha quando comparados a algoritmos tradicionais (anteriores aos chamados 'modelos de linguagem neural', os modelos n-gram) com o BOW.

O uso do BOW com TF-IDF ou LSA nos algoritmos do MAMT, e o WE e CWE nos algoritmos do MLN, busca fornecer uma perspectiva cronológica dessas proposições, tanto das representações textuais quanto dos algoritmos. Foram utilizados quatro conjuntos de dados, dois com um volume de dados maior, como o SST2 e o YAR, e dois pequenos, como o Buscapé e o Polarity, conforme descrito na Seção V.2.

Após a realização dos experimentos a Tabela V.5 indica os modelos que apresentaram melhor desempenho preditivo por conjunto, considerando a medida 'F1' como referência. O '1G T' observado na segunda linha, na coluna 'modelo', significa '1-gram TF-IDF'.

dados	modelo	acc	f1	precision	recall	mat. confusão	tempo(s)
Buscapé	BERT Base	85.52	92.06	87.74	96.82	7 51 12 365	98.9s
Polarity	SVM 1G T	84.80	84.80	84.90	84.80	207 43 33 217	331.05s
SST2	BERT Base	92.29	92.99	93.30	92.68	13311 1158 1273 16129	1340.54s
Yelp-250	BERT Base	96.04	97.45	97.54	97.35	13389 1183 1278 46941	4691.75s
Yelp-500	BERT Base	96.40	97.69	97.78	97.61	27560 2125 2294 93553	9189.68s
Yelp-750	BERT Base	96.74	97.83	97.94	97.73	45295 2880 3175 136762	13743.97s
Yelp-1M	BERT Small	96.92	97.93	97.74	98.13	58853 4251 3496 183618	9542.66

Tabela V.5: Melhor desempenho preditivo por conjunto de dados

Podemos observar que o *BERT Base* apresentou o melhor desempenho preditivo em praticamente todos os conjuntos. O *BERT Base* só não apresentou o melhor resultado no Polarity e no Yelp-1M. No entanto, nesse último ele pode ser considerado vencedor também, já que a diferença dele para o *BERT Small* (vencedor) foi de apenas 0.0003.

Em todos os conjuntos, exceto o Polarity, os BERTs ficaram na frente dos demais modelos. No Polarity o grupo MLN ficou atrás de muitos modelos do MAMT nas diferentes configurações; '1-gram TF-IDF', '2-gram TF-IDF', '1-gram LSA' e '2-gram LSA'. No Polarity o vencedor foi o SVM.

Como os BERTs utilizam o CWE pré-treinado, era esperado que os referidos modelos apresentassem bom desempenho preditivo. Os pontos mecionados acima sobre o desempenho preditivo observado no Polarity chamaram a atenção. Por que o SVM conseguiu superar os MLN no Polarity? Por que muitos modelos do MAMT em diferentes configurações (n-gram e TFIDF ou LSA) conseguiram superar os BERTs?

Como os BERTs tiveram desempenhos preditivos consistentemente melhores em todos os outros conjuntos, foi criado uma versão do YAR (onde os BERTs venceram) com igual número de exemplos do Polarity, dois mil exemplos textuais, e balanceado, tal como o Polarity, que possui dois mil exemplos, sendo mil positivos e mil negativos. O objetivo com esse conjunto, aqui chamado de 'YELP 2m-B', era avaliar o desempenho preditivo do SVM, vencedor do Polarity, e dos BERTs, vencedores do YAR. Eventualmente, alguma característica dos dados textuais presentes no Polarity poderia estar afetando o desempenho dos BERTs, ao mesmo tempo em que poderia estar sendo corretamente *capturada* pelos modelos do MAMT.

Foi considerada a possibilidade de que termos que servissem a atribuição do rótulo de classe (positivo ou negativo), por parte dos modelos, eventualmente estivessem mais presentes no intervalo do meio para o fim de cada exemplo. Em outras palavras, estariam os termos mais positivos ou negativos dos exemplos mais presentes no intervalo que vai do meio ao fim de cada exemplo? Isso poderia gerar essa diferença no desempenho, uma vez que há uma limitação quantitativa de 510 *tokens* nos BERTs (conforme descrito na Seção II.5.6).

O Polarity, como descrito na Seção V.2, possui 2.000 exemplos, sendo que em 1.543 exemplos (aproximadamente 77% do total) nós temos mais de 510 *tokens*. Em razão disso, foi criado outro conjunto chamado 'Polarity 510-T', a partir do Polarity, com 2.000 exemplos também, balanceado, mas onde todos os exemplos só possuem os 510 primeiros *tokens*. Permitindo, assim, que o SVM e os BERTs lidassem com o mesmo conjunto de termos. Buscou-se evitar, assim, que os BERTs processassem só os 510 primeiros *tokens* dos exemplos enquanto o SVM tivesse contato com todos em cada exemplo do conjunto.

A Tabela V.6 mostra os resultados nos três conjuntos; Polarity, Polarity 510-T e YELP 2m-B.

dados	modelo	f1	mat. confusão	tempo(s)
Polarity	**SVM**	**84.80**	207 43 33 217	331.05s
Polarity	BERT Small	65.98	177 73 91 159	79.99s
Polarity	BERT Base	69.22	175 75 78 172	132.93s
Polarity 510-T	**SVM**	**80.00**	189 61 39 211	268.05s
Polarity 510-T	BERT Small	67.21	176 74 86 164	196.26s
Polarity 510-T	BERT Base	67.77	139 111 65 185	530.93s
YELP 2m-B	SVM	89.20	216 34 20 230	54.94s
YELP 2m-B	**BERT Small**	**91.09**	225 25 20 230	147.72s
YELP 2m-B	BERT Base	89.96	215 35 17 233	573.6s

Tabela V.6: SVM e BERT - Polarity 510-T e YELP 2m-B

No 'Polarity 510-T' observamos que o SVM continuou com desempenho preditivo melhor que os BERTs. O F1 do SVM perdeu quase 5 pontos (em relação ao Polarity original), enquanto que os BERTs apresentaram variações menores para mais ou para menos. Ou seja, a diferença de desempenho preditivo do SVM diante dos BERTs persistiu mesmo fornecendo aos três modelos os mesmos 510 primeiros *tokens* de cada exemplo do conjunto. Não se tratando, assim, de alguma particularidade dos exemplos (presente na parte do meio para o fim dos mesmos) que estivesse sendo

capturada pelo SVM e não *capturada* pelos BERTs. Os outros três modelos do MAMT também desempenharam melhor que os MLN usando *1-gram* TF-IDF.

No 'YELP 2m-B' observamos que o desempenho preditivo do *BERT Small* foi melhor que o do SVM em quase 2 pontos. O *BERT Base* também foi melhor que o SVM, marginalmente, com diferença de 0.76. Ou seja, muito embora a primeira possibilidade (baseada na limitação quantitativa de 510 *tokens*) não tenha se confirmado, com o processamento do 'YELP 2m-B' podemos observar que há algo nos exemplos textuais do Polarity (onde o SVM e outros modelos do MAMT foram melhores) que os exemplos do YAR (onde os BERTs foram melhores) não possuem, fazendo com que naquele - e somente no Polarity - o SVM e outros modelos do MAMT apresentem desempenho preditivo melhor que os MLN. Há alguma diferença qualitativa entre os exemplos dos dois conjuntos.

A Tabela V.7 apresenta o desempenho preditivo da Bi-LSTM nos diferentes conjuntos testados.

dados	f1	mat. confusão	tempo(s ou h)	gpu
Buscapé	0.0	-	1005.06s	Tesla T4
Polarity	52.53	135 115 120 130	2008.2s	Tesla A100-SXM4-40GB
SST2	69.66	531 13938 653 16749	652.06s	Tesla A100-SXM4-40GB
Yelp-250	**86.87**	0 14572 0 48219	16478.96s ou 4,58h	Tesla V100-SXM2-16GB
Yelp-250-B	9.31	29799 1551 29742 1607	13701.41s ou 3,81h	Tesla V100-SXM2-16GB
Yelp-500	86.59	0 29685 0 95847	23594.04s ou 6,55h	Tesla T4
Yelp-500-B	25.95	57769 4736 52478 10026	25895.72s ou 7,19h	Tesla T4
Yelp-750	85.31	0 48175 0 139937	36293.94s ou 10,08h	Tesla V100-SXM2-16GB
Yelp-750-B	66.82	30081 63880 14760 79201	39423.85s ou 10,95h	Tesla T4
Yelp-1M	85.57	0 63104 0 187114	50739.85s ou 14,09h	Tesla P100-PCIE-16GB
Yelp-1M-B	50.16	98303 26884 74285 50902	56692.2s ou 15,74h	Tesla V100-SXM2-16GB

Tabela V.7: Desempenho Bi-LSTM

A Bi-LSTM apresentou comportamento irregular e disfuncional, não obtendo desempenho preditivo melhor do que os outros modelos em nenhum conjunto de dados. Nos conjuntos desbalanceados o modelo classificou com o rótulo da classe majoritária todos os exemplos, como podemos observar no 'Yelp-1M', 'Yelp-750', 'Yelp-500' e 'Yelp-250'. Como descrito na Seção V.2 esses conjuntos possuem 74,5% exemplos 'positivos' e 25,5% exemplos 'negativos'.

Nos conjuntos balanceados 'Polarity', 'SST2' (praticamente balanceado, com 54,60% exemplos 'positivos' e 45,39% 'negativos'), 'Yelp-1M-B' 'Yelp-750-B', 'Yelp-500-B' e 'Yelp-250-B' o modelo ficou com f1 menor, não classificando todos os exemplos com o rótulo da classe majoritária. A classificação nesses conjuntos ocorreu de forma irregular, em alguns o modelo classificou mais exemplos

como ‘negativos’ e, em outros, como ‘positivos’.

A Bi-LSTM utiliza os WE que são criados durante o treinamento, diferentemente dos CWE dos BERTs que são pré-treinados. Isso motivou o uso dos dois modelos, pois se baseiam em abordagens diferentes (não só no que se refere a abordagem de representação textual, mas a própria tecnologia inerente a arquitetura dos modelos) e seria interessante observar os seus desempenhos preditivos com os diferentes conjuntos. No entanto, o volume de dados utilizado talvez não tenha sido suficiente para o adequado treinamento da Bi-LSTM (com WE criados durante o treinamento) e isso pode ter impactado seu desempenho preditivo.

A Tabela V.8 apresenta o desempenho preditivo dos dois modelos BERT utilizados na pesquisa. O *BERT Small*, *bert en uncased L-4 H-512 A-8’*, e o *BERT Base*, *BERT Base cased L-12 H-768 A-12*, conforme descrição na Seção V.1.3.

dados	modelo	f1	mat. confusão	tempo(s e h)	gpu
Buscapé	BERT Small	90.96	7 51 20 357	52.56s	Tesla T4
Buscapé	BERT Base	92.06	7 51 12 365	98.9s	Tesla T4
Polarity	BERT Small	65.98	177 73 91 159	79.99s	Tesla A100-SXM4-40GB
Polarity	BERT Base	69.22	175 75 78 172	132.93s	Tesla A100-SXM4-40GB
SST2	BERT Small	91.44	13167 1302 1648 15754	844.38s	Tesla A100-SXM4-40GB
SST2	BERT Base	92.99	13311 1158 1273 16129	1340.54s	Tesla A100-SXM4-40GB
Yelp-250	BERT Small	97.32	13200 1372 1215 47004	2137.56s ou 0.59h	Tesla V100-SXM2-16GB
Yelp-250	BERT Base	97.45	13389 1183 1278 46941	4691.75s ou 1.30h	Tesla V100-SXM2-16GB
Yelp-250-B	BERT Small	95.29	30035 1315 1622 29727	2115.93s ou 0.59h	Tesla V100-SXM2-16GB
Yelp-250-B	BERT Base	95.51	30127 1223 1576 29773	4621.7s ou 1.28h	Tesla V100-SXM2-16GB
Yelp-500	BERT Small	97.65	27353 2332 2169 93678	4228.23s ou 1.17h	Tesla V100-SXM2-16GB
Yelp-500	BERT Base	97.69	27560 2125 2294 93553	9189.68s ou 2.55h	Tesla V100-SXM2-16GB
Yelp-500-B	BERT Small	95.92	60148 2357 2723 59781	4152.57s ou 1.15h	Tesla V100-SXM2-16GB
Yelp-500-B	BERT Base	96.02	60442 2063 2877 59627	9438.88s ou 2.62h	Tesla V100-SXM2-16GB
Yelp-750	BERT Small	97.82	45221 2954 3136 136801	6092.58s ou 1.69h	Tesla V100-SXM2-16GB
Yelp-750	BERT Base	97.83	45295 2880 3175 136762	13743.97s ou 3.82h	Tesla V100-SXM2-16GB
Yelp-750-B	BERT Small	96.32	90502 3459 3459 90502	6274.37s ou 1.74h	Tesla V100-SXM2-16GB
Yelp-750-B	BERT Base	96.37	90955 3006 3795 90166	13934.72s ou 3.87h	Tesla V100-SXM2-16GB
Yelp-1M	BERT Small	**97.93**	58853 4251 3496 183618	9542.66s ou 2.65h	Tesla P100-PCIE-16GB
Yelp-1M	BERT Base	97.91	59106 3998 3843 183271	18278.67s ou 5.08h	Tesla P100-PCIE-16GB
Yelp-1M-B	BERT Small	96.50	120778 4409 4367 120820	8479.07s ou 2.36h	Tesla V100-SXM2-16GB
Yelp-1M-B	BERT Base	96.55	121261 3926 4688 120499	18489.25s ou 5.14h	Tesla V100-SXM2-16GB

Tabela V.8: BERT - desempenho preditivo em todos os conjuntos

Com exceção do conjunto Polarity, podemos observar que em todos os demais conjuntos os BERTs tiveram desempenho preditivo acima de 90.00, mesmo num conjunto pequeno como o Bus-

capé. Com os conjuntos balanceados, nas versões do YAR, observamos uma diminuição no f1, em geral, entre 1 a 2 pontos. Mantendo, claro, o patamar elevado, acima de 95%.

Seus CWE pré-treinados, sendo representações densas e baseadas num grande volume de dados textuais, contribuem para esse desempenho. Diferentemente da Bi-LSTM, que cria suas representações na fase de treinamento, os CWE pré-treinados do BERT o ajudam no seu desempenho preditivo mesmo quando dispomos de poucos dados e lidamos com conjuntos pequenos.

O desempenho preditivo do *BERT Base* foi superior ao do *BERT Small* em praticamente todos os conjuntos (diferença marginal), exceto no 'Yelp-1M', onde o *BERT Small* levou a melhor, mas com uma diferença ínfima (0.02). É possível observar, também, que a diferença de desempenho do *BERT Base* para o *BERT Small* é maior, mesmo que marginalmente, nos conjuntos com menor volume de dados, Buscapé, Polarity e SST2. O que faz sentido, já que o *BERT Base* é um modelo mais robusto em sua configuração que o *BERT Small*, sendo esperado que apresente desempenho melhor.

Em alguns conjuntos o *BERT Base* teve um tempo de processamento um pouco inferior ao dobro do tempo que o *BERT Small* utilizou nos mesmos. Já em outros, o *BERT Base* teve um tempo de processamento superior ao dobro do tempo utilizado pelo *BERT Small*. O que é esperado. E, de forma geral, podemos observar que nos conjuntos YAR não balanceados os BERTs vão apresentando melhor desempenho preditivo, marginalmente, com a crescente oferta de dados.

A Tabela V.9 indica os modelos que apresentaram pior desempenho preditivo por conjunto, considerando a medida 'F1' como referência. Os '2G L' e '1G T' observado junto ao nome de alguns modelos, na coluna 'modelo', referem-se a '2-gram LSA' e '1-gram TF-IDF', respectivamente.

dados	modelo	f1	tempo(s)
Buscapé	Bi-LSTM	0.0	1005.06s
Polarity	KNN 2G L	50.20	2.81s
SST2	MNB 2G L	44.30	0.81s
Yelp-250	KNN 1G T	66.30	2530.15s
Yelp-500	KNN 1G T	65.90	11021.98s
Yelp-750	MNB 2G L	63.50	4.0s
Yelp-1M	MNB 2G L	64.00	129.4s

Tabela V.9: Piores desempenhos preditivos por conjunto de dados

A Bi-LSTM apresentou um desempenho preditivo disfuncional e, apesar de estar na tabela em razão do 'F1' no Buscapé, precisa ser analisada separadamente, como ocorreu na Tabela V.7. Podemos observar na Tabela V.9 que os modelos KNN e MNB apresentaram o pior desempenho preditivo em alguns conjuntos de dados. O *setup* '2-gram LSA' apareceu com frequência.

A Tabela V.10 indica os modelos que apresentaram melhor desempenho preditivo por conjunto, considerando a medida 'F1' como referência, e observando apenas os modelos do grupo MAMT, excluídos os MLN. Os '1G T', observado junto ao nome dos modelos na coluna 'modelo', refere-se

a ‘1-gram TF-IDF’.

dados	modelo	f1	mat. confusão	tempo(s)
Buscapé	MNB 1G T	84.40	12 46 11 366	46.68s
Polarity	SVM 1G T	84.80	207 43 33 217	331.05s
SST2	SVM 1G T	91.20	13081 1388 1401 16001	4061.94s
Yelp-250	SVM 1G T	95.00	12746 1826 1261 46958	16006.71s
Yelp-500	SVM 1G T	95.20	26188 3497 2474 93373	60637.95s
Yelp-750	SVM 1G T	95.50	43510 4665 3702 136235	56977.53s
Yelp-1M	SVM 1G T	95.60	56894 6210 4802 182312	91449s

Tabela V.10: Maiores desempenhos preditivos dos MAMT por conjunto de dados

O modelo do MAMT que apresentou o melhor desempenho preditivo em quase todos os conjuntos foi o SVM. Exceto no Buscapé, em que o MNB marcou o melhor desempenho.

A Tabela V.11 indica os *setups* (variações entre ‘1’ ou ‘2’ do ‘n-gram e entre TF-IDF e LSA) que apresentaram o melhor e o segundo melhor desempenho preditivo por conjunto, considerando a medida ‘F1’ como referência e a comparação direta entre cada modelo do MAMT com o seu par no outro *setup*. O *setup* que possui mais modelos com pontuação ‘F1’ maior que os seus pares no outro *setup* é considerado o melhor, sendo a comparação realizada entre os quatro *setups* possíveis (1-gram/2-gram com TF-IDF/LSA) em cada conjunto processado. Os ‘1G T’, observado na coluna ‘melhor’, e os ‘1G L’ e ‘2G T’, observados na coluna ‘segundo melhor’, referem-se, respectivamente, a ‘1-gram TF-IDF’, ‘1-gram LSA’ e ‘2-gram TF-IDF’.

dados	melhor	segundo melhor	range
Buscapé	1G T	1G L	80.50 a 84.40
Polarity	1G T	1G L	50.20 a 84.80
SST2	1G T	2G T	44.30 a 91.20
Yelp-250	1G T	1G L	66.30 a 95.00
Yelp-500	1G T	1G L	65.90 a 95.20
Yelp-750	1G T	1G L	63.50 a 95.50
Yelp-1M	1G T	1G L	64.00 a 95.60

Tabela V.11: Melhores setups dos MAMT em cada conjunto

O *setup* que apresentou o melhor desempenho preditivo foi o ‘1-gram TF-IDF’. Ele foi o melhor *setup* em todos os conjuntos observados - de forma consistente. O segundo melhor foi o ‘1-gram LSA’. Na coluna ‘range’ nós temos a variação entre o menor e o maior desempenho preditivo.

Diferentemente do que algumas referências relatam (menções em livros e artigos) a abordagem com ‘2-gram’ não apresentou vantagens quando comparada ao ‘1-gram’. De forma geral, os modelos

que usaram '2-gram', nos diferentes conjuntos de dados, tiveram desempenho preditivo abaixo dos seus pares que usaram '1-gram'. O LSA também não superou o TF-IDF, mesmo possuindo as vantagens descritas na Subseção II.4.2. Como dito na Subseção V.1.1, não foi utilizada nenhuma restrição de termos do vocabulário baseada na frequência dos mesmos no TF-IDF, nem no '1-gram' e nem no '2-gram'. Eventualmente, utilizando alguma restrição vocabular o resultado pode ser outro. A Tabela V.12 apresenta os problemas de convergência dos algoritmos do MAMT durante o processamento. Em cada linha da referida tabela nós temos a indicação de um *setup* e modelo que não concluiu a etapa de treinamento ao lidar com o conjunto de dados indicado na coluna 'dados' no tempo limite do *Colab*, 24 horas, descrito na Seção VI.1. As linhas estão organizadas de acordo com o *setup* e o modelo.

dados	setup	modelo	hiperparâmetros usados	origem	execução
Yelp-1M	1G T	SVM	kernel: linear C: 1	vencedor do Yelp-500	concluída
Yelp-750	1G T	SVM	kernel: linear C: 1	vencedor do Yelp-500	concluída
Yelp-1M	2G T	SVM	-	-	não executado
Yelp-750	2G T	SVM	-	-	não executado
Yelp-500	2G T	SVM	kernel: sigmoid C: 10	vencedor do Yelp-250	falhou
Yelp-1M	2G T	RF	-	-	não executado
Yelp-750	2G T	RF	n estimators: 125 max samples: 0.75	vencedor do Yelp-250	falhou
Yelp-500	2G T	RF	n estimators: 125 max samples: 0.75	vencedor do Yelp-250	concluída
Yelp-1M	2G L	SVM	kernel: rbf C: 20	vencedor do Yelp-750	concluída

Tabela V.12: SVM e RF - hiperparâmetros usados para contornar a limitação do Colab

Ao observar a Tabela V.12 é preciso considerar que os algoritmos do MAMT não possuíram, para a escolha de hiperparâmetros, a mesma quantidade de hiperparâmetros. A presença nessa tabela não indica que os dois algoritmos, SVM e RF, são piores - nesse aspecto - do que o KNN e MNB. Ela serve apenas como uma referência para falar sobre problemas na execução, limitações observadas no *Colab* e como os mesmos foram contornados no decorrer da pesquisa.

O SVM e o RF tiveram problemas para concluir o processamento (treinamento e teste com o conjunto de dados), como podemos observar, com *setup* '2-gram TF-IDF'. Nas execuções, sempre que possível, o treinamento (com escolha de hiperparâmetros) e teste eram executados de forma consecutiva. Quando o algoritmo não terminava as duas etapas no prazo limite do *Colab*, num primeiro momento, optava-se pela realização do treinamento, armazenagem do modelo e, depois, o carregamento do mesmo para a etapa de testes com a parte restante do conjunto de dados.

Nesses casos relacionados na Tabela V.12 nem com a execução separada entre treinamento e teste foi possível concluir o treinamento antes das 24 horas de uso limite do *Colab*. Assim, foram utilizados hiperparâmetros que venceram no conjunto imediatamente abaixo no número de exemplos do mesmo algoritmo e *setup* ('1-gram TF-IDF' com '1-gram TF-IDF', '1-gram LSA' com '2-gram LSA' e assim por diante), separando a execução de treinamento da de teste. E caso o algoritmo e *setup* em questão, no conjunto de dados imediatamente inferior, também não tivesse concluído o treinamento e teste, pegava-se o do conjunto seguinte, com menor volume de dados ainda. Na etapa de treinamento do MAMT nós temos, no SST2 e YAR, a utilização do HGS, como descrito na Subseção V.1.2.

Desse modo, como indicado na Tabela V.12, nós temos casos em que a execução com o conjunto de hiperparâmetros vencedor de conjunto 'anterior' (de volume menor) foi concluída com sucesso, casos em que ocorreu a falha - não convergência no prazo limite do *Colab*, dentro das 24 horas - e casos cuja a execução não foi tentada com essa abordagem, já que o conjunto antecessor, com volume menor, não concluiu com êxito.

Ou seja, mesmo usando o HGS, mesmo separando execuções de treinamento e armazenagem do modelo, e recarregamento do modelo e teste, mesmo utilizando hiperparâmetros vencedores do mesmo algoritmo e *setup* em conjunto com volume imediatamente inferior, em algumas execuções os algoritmos não convergiram no tempo limite de uso do *Colab*, dentro das 24 horas.

A Tabela V.13 apresenta os tempos total e médio de execução de cada modelo.

modelo	tempo total(s ou h)	tempo médio(s ou h)
KNN	28.561,42s ou 7,93h	1,13h
MNB	61,11s	8,73s
SVM	229.497,8s ou 63,75h	**9,11h**
RF	162.564,57s ou 45,16h	6,45h
BERT SMALL	22.977,96s ou 6,38h	0,91h
BERT BASE	47.476,44s ou 13,19h	1,88h
BI-LSTM	130.772,11s ou 36,33h	5,19h

Tabela V.13: Tempo total e médio de execução por modelo

O modelo do MAMT com o melhor desempenho preditivo foi o SVM, como indicado na Tabela V.10. No entanto é interessante observar na Tabela V.13 que ele também possui o maior tempo de processamento (treinamento e teste). O SVM e o RF tiveram, respectivamente, o maior e o segundo maior tempo de processamento.

A Bi-LSTM teve o maior tempo entre os MLN e o terceiro maior tempo geral. Isso era esperado já que seus WE, como dito anteriormente, são criados durante o treinamento da rede. O *BERT Base*, também como esperado, ficou com tempo maior do que o *BERT Small* em razão de sua arquitetura mais robusta.

Os tempos indicados na Tabela V.13 servem apenas como referência, não podendo ser interpreta-

dos de forma rígida/cartesiana. Ocorreram variações entre *GPUs* nos MLN, variações da quantidade de hiperparâmetros testados em cada algoritmo no MAMT para escolha do melhor conjunto e particularidades afins (nao foi realizada busca por hiperparâmetros nos MLN, por exemplo).

As Tabelas V.14, V.15, V.16 e V.17 apresentam os melhores desempenhos preditivos de cada modelo do MAMT por conjunto de dados, além do *setup* utilizado. '1G T', '1G L', '2G T', '2G L', significam, respectivamente, '1-gram TF-IDF', '1-gram LSA', '2-gram TF-IDF' e '2-gram LSA'.

É possível observar na Tabela V.14 que o LSA teve um bom desempenho preditivo com o KNN (comparativamente com os demais modelos do MAMT), sendo o melhor tipo de representação para lidar com as versões do YAR, o conjunto maior.

dados	representação	f1	tempo(s ou h)
Buscapé	1G T	83.30	45.03s
Polarity	2G T	71.20	4.79s
SST2	1G T	86.10	202.92s
Yelp-250	1G L	85.50	889.37s ou 0.25h
Yelp-500	2G L	85.90	4089.27s ou 1.14h
Yelp-750	1G L	86.20	8345.95s ou 2.32h
Yelp-1M	**1G L**	**86.50**	14984.09s ou 4.16h

Tabela V.14: KNN - desempenho preditivo em todos os conjuntos

Observamos na Tabela V.15 que o MNB apresentou o melhor desempenho preditivo usando '1-gram TF-IDF'. O tempo de convergência e conclusão do processamento foi o melhor de todos os algoritmos - de longe. No entanto, é importante observar que o espaço de busca de hiperparâmetros, durante o treinamento (usando HGS ou GS - a depender do conjunto), está bastante limitado quando comparado aos outros algoritmos do MAMT. Desse modo, para além de questões relacionadas a forma como o algoritmo processa a entrada, é esperado que ele tenha um tempo menor. Mesmo considerando isso, a percepção é a de que o MNB tem bom tempo de convergência, em razão da diferença vertiginosa para os demais algoritmos do MAMT.

dados	representação	f1	tempo(s)
Buscapé	1G T	0.844	46.68s
Polarity	1G T	0.818	0.23s
SST2	1G T	0.866	0.51s
Yelp-250	1G T	0.899	1.54s
Yelp-500	1G T	0.906	3.06s
Yelp-750	1G T	0.909	4.34s
Yelp-1M	**1G T**	**0.91**	4.75s

Tabela V.15: MNB - desempenho preditivo em todos os conjuntos

Observamos na Tabela V.16 que o SVM, modelo com melhor desempenho preditivo do MAMT, obteve seu melhor desempenho em todos os conjuntos de dados utilizando o '1-gram TF-IDF'.

Na Tabela V.17 observamos que o RF, no 'Yelp-250' e 'Yelp-500', respectivamente, teve os seus

dados	representação	f1	tempo(s ou h)
Buscapé	1G T	0.828	33.62s
Polarity	1G T	0.848	331.05s
SST2	1G T	0.912	4061.94s
Yelp-250	1G T	0.95	16006.71s ou 4,45h
Yelp-500	1G T	0.952	60637.95s ou 16,84h
Yelp-750	1G T	0.955	56977.53s ou 15,83h
Yelp-1M	**1G T**	**0.956**	91449s ou 25,40h

Tabela V.16: SVM - desempenho preditivo em todos os conjuntos

dois maiores tempos de execução dentre todos os conjuntos. Justamente com execuções que utilizam o '2-gram TF-IDF'.

dados	representação	f1	tempo(s ou h)
Buscapé	1G L	0.821	26.89s
Polarity	1G T	0.822	43.42s
SST2	1G T	0.895	893.58s
Yelp-250	2G T	0.89	63272.88s ou 17,58h
Yelp-500	2G T	0.899	57642s ou 16,01h
Yelp-750	**1G T**	**0.899**	17451.62s ou 4,85h
Yelp-1M	1G T	0.897	23234.18s ou 6,45h

Tabela V.17: RF - desempenho preditivo em todos os conjuntos

É possível observar, também, nas Tabelas que indicam o desempenho preditivos dos modelos com os diferentes conjuntos de dados, a melhoria gradual do desempenho com a maior oferta de dados. Em geral, os desempenhos preditivos vão melhorando na medida em que mais dados são oferecidos aos modelos, tanto modelos *n-gram*, do MAMT, quando o MLN.

Capítulo VI Conclusão

O desenvolvimento tecnológico observado nos últimos anos tem acarretado mudanças estruturais na forma como nos relacionamos. As melhorias nas possibilidades de geração, coleta e manipulação de dados que observamos nas últimas décadas nos permitiu buscar desenvolvimentos algorítmicos e de aplicações que possibilitassem a extração de informações úteis a tomada de decisão, impactando a gestão de negócios, governos e pessoas. O volume de dados não estruturados - por se tratar da forma mais fácil de dado que pode ser criado - cresce a cada ano, acompanhando o desenvolvimento tecnológico e toda a capilaridade que os dispositivos atuais oferecem. Nesse cenário, o problema da mineração de texto tem ganhado atenção crescente em razão do grande volume de dados textuais que são criados na web e em outras aplicações centradas em informações [4].

As representações de palavras/documentos utilizadas em meio computacional apresentam nítida evolução, permitindo que os computadores atuais possam lidar com a linguagem natural de uma forma cada vez mais segura, possibilitando uma elevação das possibilidades de interpretação, manipulação e resposta da linguagem humana. O BOW com TF-IDF (e variantes), o BOW com LSA, o WE e o CWE, são técnicas de representação do dado textual que representam marcos nas melhorias das formas de representação ao longo dos anos, superando limitações e abrindo novas possibilidades no horizonte do NLP.

Esse estudo teve como objetivo a comparação de diferentes abordagens na representação do dado textual, sendo seus usos observados em diferentes algoritmos numa tarefa de classificação de polaridade binária. Foram utilizados, no MAMT, o BOW com TF-IDF e o BOW com LSA e, no MLN, o WE e o CWE. Foram consideradas, nessa opção, as restrições de uso observadas no *Colab* e, também, a possibilidade de avaliação considerando uma perspectiva cronológica dessas formas de representação e seus usos em algoritmos propostos nas últimas décadas.

Pensando na questão elaborada no Capítulo I dessa pesquisa - em que medida as formas de representação textual e algoritmos mais recentes, baseados em redes neurais profundas, superam os resultados dos modelos *n-gram* em diferentes conjuntos e com diferentes volumes de dados na tarefa de classificação? É interessante observar a superioridade das abordagens (forma de representação e algoritmos) mais recentes, conforme apresentado na Tabela V.8. Os BERTs, nas duas versões utilizadas, tiveram desempenho preditivo consistentemente acima dos demais modelos, exceto no conjunto Polarity, como descrito na Seção V.3. Quanto a esse último, há alguma diferença no

conjunto capturada pelos modelos *n-gram* que o modelos BERT não capturaram, tendo esses últimos desempenho preditivo inferior àqueles (ver Tabela V.6). Aliás, os modelos BERT, KNN e MNB apresentaram quedas no seu desempenho preditivo ao lidar com esse conjunto (tendo como referência o desempenho no conjunto Buscapé e no SST2). O SVM e o RF lidaram bem com o Polarity, considerando as mesmas referências.

Com 'F1' elevado, seus CWE pré-treinados e as particularidades de sua arquitetura, os modelos BERT mostraram-se uma ótima escolha - de forma geral. Por outro lado, o desempenho da Bi-LSTM foi disfuncional (Tabela V.7). Isso pode estar relacionado ao volume de dados necessário para o correto treinamento e criação de seus WE, criados durante o treinamento da rede (diferentemente dos utilizados pelos modelos BERT, pré-treinados). O uso de *embeddings* pré-treinados, aliás, se mostrou muito interesssante, na medida em que endereça essa dificuldade, diminuindo tempo de processamento e custos. A superioridade dos *embeddings* pré-treinados em relação aos *embeddings* sem pré-treinamento é evidente. Em relação ao volume de dados, inclusive, é perceptível a melhoria gradual do desempenho preditivo de todos os demais modelos com a maior oferta de dados. De forma geral - e mesmo que marginalmente de um volume para outro - os desempenhos vão melhorando na medida em que mais dados são oferecidos aos modelos, tanto modelos *n-gram*, do MAMT, quando o MLN.

É interessante observar que os modelos *n-gram* obtiveram desempenhos muito satisfatórios. O SVM, por exemplo, o melhor do MAMT, obteve 'F1' acima de 91.00 em todos os conjuntos grandes utilizados (i.e., o SST2 e o YAR em suas diferentes versões). Chegando a ter melhor desempenho preditivo no conjunto Polarity do que os MLN (Tabelas V.5 e V.16). Os modelos *n-gram* desempenharam consistentemente melhor com '1-gram TF-IDF', tendo o '1-gram LSA' o segundo melhor desempenho (Tabela V.11). Os desempenhos preditivos dos modelos *n-gram* utilizando '2-gram' foram inferiores ao '1-gram'. Além de não terem apresentado bons desempenhos preditivos quando comparados as versões do *1-gram* (seja com TF-IDF ou LSA), o SVM e o RF, usando '2-gram', tiveram problemas de convergência no tempo limite de uso do *Colab* (Tabela V.12).

O tempo de execução (treinamento e teste) dos modelos foi indicado na Tabela V.13. Os três modelos com maior tempo total e médio foram, em ordem decrescente, o SVM, o RF e a Bi-LSTM. Os modelos *n-gram* SVM e RF ficaram com tempo médio acima dos BERTs. O KNN obteve tempo de execução melhor que o BERT BASE. O MNB ficou com tempo muito abaixo de todos os demais modelos, seja *n-gram* ou MLN. É importante esclarecer que os modelos *n-gram* do MAMT foram submetidos a busca por hiperparâmetros enquanto os do MLN não. E o MNB utilizou um espaço de busca de hiperparâmetros muito limitado, quando comparado aos demais modelos *n-gram*. Além disso, conforme descrito na Subseção V.1.1, não foi utilizada nenhuma restrição vocabular nos modelos *n-gram*. A Tabela V.13, em razão dessas particularidades, serve como uma referência

apenas.

Os problemas de convergência dentro do tempo limite de uso do *Colab*, observações sobre o tempo total e médio de execução dos modelos, dentre outras observações e comparações entre os modelos *n-gram* do MAMT e os MLN, precisam ser feitas (se forem feitas) com cautela no contexto do presente estudo. Isso porque as especificidades dos algoritmos de cada grupo, a existência ou inexistência de busca por melhores hiperparâmetros na execução, dentre outros, pode tornar a comparação injusta e ineficaz.

Embora as duas versões do BERT, como citado acima, tenham tido desempenho preditivo consistentemente acima dos demais modelos, determinar o tipo de abordagem não é algo simples. Como indicado acima, no Polarity os modelos BERT não obtiveram os melhores desempenhos. Há muitas vantagens em utilizar algoritmos/modelos de classificação semântica de texto, os MLN, baseados em aprendizagem profunda, como o BERT (inclusive a Bi-LSTM, desde que superados os problemas aqui identificados e com os ajustes devidos) quando comparamo-os com os algoritmos tradicionais de classificação de texto, os modelos *n-gram*, utilizados no MAMT [60; 7] (ver Seção II.4). Em comparação com os modelos de linguagem *n-gram*, os MLN podem lidar com textos muito mais longos, possuem capacidade de generalização maior sobre contextos de palavras semelhantes e conseguem ser mais precisos na previsão de palavras [60]. Ao mesmo tempo, segundo os mesmos autores, possuem uma complexidade maior, são mais lentos, precisam de mais energia para treinar e são menos interpretáveis que os modelos *n-gram*.

No entanto, em razão dos fatores elencados nessa pesquisa (nessa Conclusão e na Seção V.3), determinar o tipo de abordagem não é simples, pois depende de diferentes fatores, tais como: conjunto de dados (características e volume), recursos disponíveis, bases de conhecimento, características do algoritmo, dentre outros. Em razão de suas especificidades, cada tipo de abordagem possui vantagens e desvantagens, fazendo com que a escolha do método mais adequado (técnica de representação textual/modelo de linguagem e algoritmo de aprendizagem) dependa de diferentes fatores inter-relacionados. Nesse sentido, a percepção vai ao encontro do observado por [7]. A superioridade dos modelos de classificação semântica de texto, os MLN, é evidente. No entanto, os modelos *n-gram* podem ser usados em muitas tarefas, em especial as menores, já que possuem desempenho preditivo muito satisfatório, mesmo considerando as abordagens mais recentes nesse campo.

VI.1 Limitações

O presente estudo, ao avaliar o uso das diferentes formas de representação textual e diferentes algoritmos, utilizou uma tarefa de classificação binária. Nada impede, contudo, que uma tarefa de classificação multiclasse possa servir de base para a avaliação.

Os algoritmos do grupo MAMT podem ser utilizados, no tocante a representação do dado textual, com *embeddings* também. Foi uma opção da pesquisa utilizá-los apenas no grupo MLN. Como dito na IV, essa opção considerou as restrições no ambiente de execução indicadas nessa Seção VI.1 e, também, a possibilidade de observar/avaliar essas formas de representação e algoritmos sob uma perspectiva cronológica dos momentos em que foram propostos; BOW com TF-IDF e o BOW com LSA no MAMT, e WE e CWE no MLN.

Não foram removidas as *stop words* nos exemplos dos conjuntos quando processados pelos MLN. E todos os exemplos foram convertidos em caracteres minúsculos. Como o *BERT Small* não faz distinção entre maiúsculas e minúsculas nos seus *embeddings*, foi feita essa opção. No entanto, os *embeddings* do *BERT Base* preserva essa diferença e, também, os marcadores de acento, conforme descrição contida em [27; 1]. Eventualmente remover as *stop words* e processar o texto atentando para o tipo de *embeddings* que está sendo utilizado pode contribuir para alguma melhoria, ainda que marginal, no desempenho preditivo do modelo.

Esse estudo buscou estabelecer uma base comum entre os algoritmos do grupo MLN, tanto no pré-processamento do texto, quanto na escolha de hiperparâmetros (tamanho do *lote*, *época* etc.), o que, claro, desconsidera particularidades de cada algoritmo utilizado e podem não representar a melhor escolha em cada caso. O mesmo pode ser dito em relação aos algoritmos do grupo MAMT. Por exemplo, na vetorização dos exemplos textuais, utilizando o TF-IDF, não foi utilizada nenhuma restrição de termos do vocabulário baseada na frequência dos mesmos. Alguns algoritmos, como o SVM e RF, podem apresentar dificuldades de convergência relacionados ao volume de dados e ao modo como processam suas entradas. Lidar com um vocabulário reduzido, fazendo uso de restrições baseadas na frequência para limitar o vocabulário, eventualmente, pode ajudar a evitar esse tipo de problema e, também, no desempenho preditivo dos modelos.

Buscar essa base comum no pré-processamento do texto e na escolha de hiperparâmetros, quando possível e entre os algoritmos de cada grupo, teve o intuito de buscar uma comparação justa, facilitando a observação e compreensão dos desempenhos preditivos dos modelos e das formas de representação textual utilizados. Mas é importante salientar a importância de observar os aspectos particulares de cada algoritmo (em cada um dos grupos) e abordagem em cada caso, para uso específico.

Como dito em V.1, essa pesquisa não teve como objetivo o ajuste fino dos hiperparâmetros em cada algoritmo. Alguns estudos serviram de base para escolha do espaço de busca dos hiperparâmetros, no caso do MAMT, e para escolha dos hiperparâmetros utilizados no MLN. Esse tipo de busca exigiria - se realizada com a devida diligência - um processo de testagem e fluxo de trabalho próprio, o que normalmente resulta em outros estudos, mais específicos. Espaço de buscas maiores, ou com outras variações de valores, eventualmente, podem ser interessantes no MAMT. A realização

de testes com outros hiperparâmetros no MLN é algo a ser pensado, evidentemente. Variações no número de *épocas*, *lote*, taxa de aprendizado e outros seriam interessantes para avaliar eventuais ganhos no desempenho preditivos desses modelos.

Seria interessante utilizar um volume de dados maior para o treinamento da Bi-LSTM. O maior conjunto de dados utilizado nesse trabalho, uma versão do YAR, possui 1.000.871 exemplos. O conjunto original, no entanto, possui 8 milhões de exemplos. Seria interessante fazer experimentos com a Bi-LSTM utilizando segmentos maiores que o maior segmento do YAR utilizado nesse trabalho, já que o desempenho preditivo da mesma, em todos os conjuntos utilizados, não foi bom. A Bi-LSTM utiliza WE criados durante a fase de treinamento do modelo e o volume de dados utilizado, claro, é importante.

Não foram utilizadas, também, as versões do YAR balanceadas com o MAMT, tal como foi utilizado com os MLN. Com o MAMT foram utilizados apenas as versões primárias dos conjuntos de dados, desbalanceadas.

Foram utilizados, no total, onze conjuntos de dados, como descrito em V.2. No entanto, apenas um deles, o Buscapé, possui textos em Português. Todos os demais estão em Inglês. E o Buscapé é um dos dois conjuntos pequenos (com 1.738 exemplos) utilizados na pesquisa. Será que há diferenças (significativas) no desempenho preditivo se utilizarmos o Português como idioma alvo da abordagem?

O *Google Colab* é uma ferramenta muito interessante, mas possui grandes limitações. Mesmo no maior plano oferecido (Colab Pro+ R$ 258,00) os *notebooks* permanecem em execução por, no máximo, 24 horas em 'segundo plano'. E, como é de se esperar, a depender do algoritmo, do tipo de representação textual e do volume de dados utilizados, isso não é suficiente para as fases de treinamento e teste. Alguns dos algoritmos simplesmente não convergiram até esse limite. O volume de dados utilizado na pesquisa, evidentemente, precisou atentar-se a essas limitações.

Os parâmetros utilizados no HGS e no GS buscavam, evidentemente e na medida do possível, o melhor ajuste considerando essa limitação. No caso do HGS, por exemplo, o *fator* usado em cada algoritmo do MAMT tinha como objetivo induzir o HGS a utilizar todos os exemplos do conjunto de treinamento apenas na última iteração, com o conjunto de hiperparâmetros vencedor.

O GS - sendo uma busca exaustiva em todas as combinações possíveis de hiperparâmetros e utilizando todo o volume de dados disponível - só foi utilizado nos conjuntos menores, Buscapé e Polarity. Outro exemplo foi o *cross-validation*. Tanto o HGS quanto o GS usaram o *cv* observando o volume de dados e essa limitação no tempo de uso diário do *Google Colab*.

Outra limitação no *Google Colab* é a impossibilidade de escolha do modelo de *GPU*. O usuário, a cada vez que ativa um ambiente de execução, tem acesso a um modelo de GPU escolhido pelo *Google Colab*. Evidentemente, não é o cenário mais adequado, podendo prejudicar as comparações

baseadas no tempo de execução e derivadas que façam uso da mesma.

O *Google Colab* implementou uma alteração no serviço no final de Setembro/começo de Outubro de 2022. Antes, o usuário pagava um valor fixo (Colab Pro+ R$ 258,00, por exemplo) e poderia utilizar diferentes *notebooks* simultaneamente com execução em segundo plano, tendo acesso aos melhores modelos de *GPU* e com uso ilimitado, exceto pela quantidade de horas de até 24 horas/dia por *notebook*.

Antes era possível ficar utilizando diferentes *notebooks* simultaneamente onde cada um tinha uma limitação de 24 horas, todos os dias do mês, com um único pagamento. Com a alteração o sistema de cobrança mudou. Foi aplicada a cobrança por unidades de computação. O consumo por hora varia de acordo com o *setup* escolhido para o *notebook*; com ou sem GPU, GPU *premium* ou não - ainda assim sem poder escolher o modelo.

A maior parte das execuções ocorreram no modelo de cobrança anterior. Se tivessem ocorrido no novo modelo, os custos teriam que ser considerados de forma mais cuidadosa ao fazer a opção pelo *Google Colab*. Atualmente, com o maior plano do serviço, o Colab Pro+ R$ 258,00, o usuário tem 500 unidades de computação. Para treinar qualquer algoritmo mais robusto, esses mais atuais, baseados em aprendizagem profunda e/ou tendo um volume de dados muito grande, o usuário facilmente irá consumir esse crédito e terá que comprar mais para continuar usando. E, ainda assim, terá que atentar para a limitação das 24 horas diárias de uso por *notebook*.

Além disso, foi observado que o uso de vários *notebooks* simultaneamente ficou mais restrito (não foi possível ligar um que usava GPU enquanto usava dois sem GPU executando em segundo plano). Outro problema do *Google Colab* é a imprecisão das regras do serviço. Não são transparentes, dificultando um planejamento mais detalhado e projeções de custos. Apenas com a utilização e teste é que foi possível observar as diferentes quantidades de unidades de computação cobradas por cada tipo de placa; GPUs p100 = 4 unidades/hora, v100 = 5 unidades/hora, a100 = 13,08 unidades/hora e 0,19 unidades/hora por *notebook* sem GPU, por exemplo. Foram percebidas variações eventuais nessas quantidades.

VI.2 Trabalhos Futuros

Seria interessante observar o desempenho preditivo dessas abordagens e modelos numa tarefa de classificação multiclasse. Ela pode oferecer novas perspectivas quanto ao que foi mostrado nesse trabalho. Além disso, o uso de WE no MAMT, igualmente, pode oferecer perspectivas além das mostradas nessa pesquisa.

Em pesquisas futuras pode ser útil - a depender do objetivo - manter uma aderência maior as especificidades de cada algoritmo; considerando o *Cased* do WE pré-treinado utilizado pelo modelo, por exemplo, ou *setups* (‘2-gram TF-IDF’ etc.) com alguma limitaçâo vocabular baseada na frequên-

cia de uso, dentre outros. Alguns algoritmos, como o SVM e RF, podem apresentar dificuldades de convergência relacionados ao volume de dados e ao modo como processam suas entradas. Lidar com um vocabulário reduzido, fazendo uso de restrições vocabulares pode, eventualmente, ajudar a evitar esse tipo de problema, melhorando o tempo de execução e/ou o desempenho preditivo dos modelos - mesmo que de forma marginal. Ainda mais considerando eventuais limitações no tempo de execução, como as enfrentadas nessa pesquisa.

O aumento do espaço de busca dos conjuntos de hiperparâmetros nos algoritmos do MAMT seria interessante. A realização de testes com outros hiperparâmetros no MLN é algo a ser pensado, evidentemente. Variações no número de *épocas*, *lote*, taxa de aprendizado e outros seriam importantes para avaliar eventuais ganhos no desempenho preditivos desses modelos.

Se a Bi-LSTM com *embeddings* sem pré-treinamento, em razão de alguma perspectiva comparativa, fosse utilizada novamente, seria interessante usar conjuntos de dados mais volumosos para observar seu comportamento. Eventualmente o ajuste para o seu desempenho preditivo adequado dependa de um volume de dados maior do que o que foi utilizado nessa pesquisa.

Uma abordagem futura que lidasse, também, com grandes conjuntos de textos em Português seria útil para avaliar o desempenho preditivo dessas abordagens e algoritmos. A língua portuguesa é uma das línguas mais faladas no mundo [108; 36], sendo a quinta mais utilizada na internet e a terceira em redes sociais como Facebook e Twitter [36].

Além disso, realizar execuções das versões balanceadas do YAR com o MAMT (só realizado com os MLN nessa pesquisa) e utilizar o BERT com diferentes conjuntos de dados textuais cujos exemplos possuam mais de 510 *tokens* (em razão da limitação quantitativa do BERT) são opções que podem, eventualmente, oferecer novas perspectivas e percepções sobre essas abordagens e algoritmos.

Seria interessante utilizar, em estudos futuros, um meio de processamento onde a flexibilidade de escolha da GPU e as limitações do tempo de execução não fossem como as encontradas nessa pesquisa. Apesar de interessante, com vantagens no uso, o *Colab* possui muitas restrições no serviço e ele precisa ser avaliado com cautela em cada caso.

Referências

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] Agarwal, A., Biadsy, F., and Mckeown, K. Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 24–32, 2009.

[3] Aggarwal, C. C. and Zhai, C. An introduction to text mining. In *Mining text data*, pages 1–10. Springer, 2012a.

[4] Aggarwal, C. C. and Zhai, C. *Mining text data*. Springer Science & Business Media, 2012b.

[5] Aghababaei, S. and Makrehchi, M. Mining twitter data for crime trend prediction. *Intelligent Data Analysis*, 22(1):117–141, 2018.

[6] Agun, H. V. and Yilmazel, O. Document embedding approach for efficient authorship attribution. In *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*, pages 194–198. IEEE, 2017.

[7] Altinel, B. and Ganiz, M. Semantic text classification: A survey of past and recent advances. *Information Processing and Management*, 54(6):1129–1153, 2018.

[8] Araujo, A., Gôlo, M., and Marcacini, R. Opinion mining for app reviews: an analysis of textual representation and predictive models. *Automated Software Engineering*, 29(1), 2022.

[9] Atak, E., Karaoğlu, D., Serttürk, S., Koyuncu Irmak, D., and Yenenler-Kutlu, A. Performing the comparative analysis to understand the functional roles of genes in different pathways of cardiomyopathy disease. *Meta Gene*, 30, 2021.

[10] Baldi, P., Brunak, S., Frasconi, P., Soda, G., and Pollastri, G. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

[11] Becker, K. and Tumitan, D. Introdução à mineração de opiniões: Conceitos, aplicações e desafios. *Simpósio brasileiro de banco de dados*, 2013.

[12] Bekkerman, R. and Allan, J. Using bigrams in text categorization. Technical report, Technical Report IR-408, Center of Intelligent Information Retrieval, UMass ..., 2004.

[13] Bellan, P., Dragoni, M., and Ghidini, C. A qualitative analysis of the state of the art in process extraction from text. *AIxIA 2020 Discussion Papers Workshop*, pages 19–30, 2020.

[14] Bengio, Y., Ducharme, R., and Vincent, P. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[15] Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[16] Biswas, B., Mukhopadhyay, A., Bhattacharjee, S., Kumar, A., and Delen, D. A text-mining based cyber-risk assessment and mitigation framework for critical analysis of online hacker forums. *Decision Support Systems*, 2021.

[17] Bouaziz, A., Dartigues-Pallez, C., Costa Pereira, C. d., Precioso, F., and Lloret, P. Short text classification using semantic random forest. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 288–299. Springer, 2014.

[18] Camacho-Collados, J. and Pilehvar, M. T. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788, 2018.

[19] Cambria, E., Schuller, B., Xia, Y., and Havasi, C. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent systems*, 28(2):15–21, 2013.

[20] Chen, J., Yang, Y., and Liu, H. Mining bilateral reviews for online transaction prediction: A relational topic modeling approach. *Information Systems Research*, 32(2):541–560, 2021.

[21] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[22] Colas, F. and Brazdil, P. Comparison of svm and some older classification algorithms in text classification tasks. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 169–178. Springer, 2006.

[23] Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[24] Crain, S. P., Zhou, K., Yang, S.-H., and Zha, H. Dimensionality reduction and topic modeling: From latent semantic indexing to latent dirichlet allocation and beyond. In *Mining text data*, pages 129–161. Springer, 2012.

[25] Dai, Z. and Callan, J. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 985–988, 2019.

[26] Deerwester, S., Dumais, S., Landauer, T., et al. Indexing by latent semantic analysis journal of the american society for information science, 1990.

[27] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[28] Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[29] Erk, K. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653, 2012.

[30] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996a.

[31] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. Advances in knowledge discovery and data mining. American Association for Artificial Intelligence, 1996b.

[32] Feldman, R., Sanger, J., et al. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007.

[33] Forman, G. Feature selection for text classification. *Computational methods of feature selection*, 16:257–274, 2007.

[34] Frank, E. and Bouckaert, R. R. Naive bayes for text classification with unbalanced classes. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Knowledge Discovery in Databases: PKDD 2006*, pages 503–510, Berlin, Heidelberg. Springer Berlin Heidelberg, 2006.

[35] Ghawi, R. and Pfeffer, J. Efficient hyperparameter tuning with grid search for text categorization using knn approach with bm25 similarity. *Open Computer Science*, 9(1):160–180, 2019.

[36] Gomes, N., Lopes, S., and Araújo, S. Mobile learning: a powerful tool for ubiquitous language learning. *New perspectives on teaching and working with languages in the digital era*, pages 189–199, 2016.

[37] Gong, Y. and Liu, X. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25, 2001.

[38] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

[39] Graves, A. and Schmidhuber, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

[40] Gu, X., Zhang, H., Zhang, D., and Kim, S. Deep api learning. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, pages 631–642, 2016.

[41] Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. Knn model-based approach in classification. In *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems* , pages 986–996. Springer, 2003.

[42] Gupta, A., Dengre, V., Kheruwala, H., and Shah, M. Comprehensive review of text-mining applications in finance. *Financial Innovation*, 6(1), 2020.

[43] Gupta, P., Chaudhary, Y., Buettner, F., and Schütze, H. Texttovec: Deep contextualized neural autoregressive topic models of language with distributed compositional prior, 2019.

[44] Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Haryana, India; Burlington, MA, 3 edition, 2011a.

[45] Han, J., Pei, J., and Kamber, M. *Data mining: concepts and techniques*. Elsevier, 2011b.

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

[47] Hasan, M., Nasreen, M., and Chowdhury, M. Gender-inclusive disaster management policy in bangladesh: A content analysis of national and international regulatory frameworks. *International Journal of Disaster Risk Reduction*, 41, 2019.

[48] Hassan, A. and Mahmood, A. Deep learning for sentence classification, 2017a.

[49] Hassan, A. and Mahmood, A. Efficient deep learning model for text classification based on recurrent and convolutional layers. volume 2017-December, pages 1108–1113, 2017b.

[50] Hinton, G. E., McClelland, J., and Rumelhart, D. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. *Distributed Representations*, pages 77–109, 1986.

[51] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[53] Horn, F. Context encoders as a simple but powerful extension of word2vec. *arXiv preprint arXiv:1706.02496*, 2017.

[54] Hu, X. and Liu, H. Text analytics in social media. In *Mining text data*, pages 385–414. Springer, 2012.

[55] Huang, A. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56, 2008.

[56] Huang, Q., Mao, J., and Liu, Y. An improved grid search algorithm of svr parameters optimization. In *2012 IEEE 14th International Conference on Communication Technology*, pages 1022–1026, 2012.

[57] Joshi, K., Gupta, A., Mittal, S., Pearce, C., Joshi, A., and Finin, T. Alda: Cognitive assistant for legal document analytics. volume FS-16-01 - FS-16-05, pages 149–152, 2016.

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

[59] Jung, J.-O., Crnovrsanin, N., Wirsik, N. M., Nienhüser, H., Peters, L., Popp, F., Schulze, A., Wagner, M., Müller-Stich, B. P., Büchler, M. W., et al. Machine learning for optimized individual survival prediction in resectable upper gastrointestinal cancer. *Journal of Cancer Research and Clinical Oncology*, pages 1–12, 2022.

[60] Jurafsky, D. and Martin, J. H. Speech and language processing (3rd draft ed.), 2019.

[61] Kansaon, D., Brandão, M. A., Reis, J. C. S., Barbosa, M., Matos, B., and Benevenuto, F. Portuguese Comparative Sentences: A Collection of Labeled Sentences on Twitter and Buscapé, 2020.

[62] Kim, S.-B., Han, K.-S., Rim, H.-C., and Myaeng, S. H. Some effective techniques for naive bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466, 2006.

[63] Kleine-Cosack, C. Recognition and simulation of emotions. *Archived from the original on May*, 28:2008, 2006.

[64] Kohavi, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[65] Langley, P., Iba, W., Thompson, K., et al. An analysis of bayesian classifiers. In *Aaai*, volume 90, pages 223–228. Proc. National Conference on Artificial Intelligence, 1992.

[66] Lewis, D. D. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50, 1992.

[67] Li, Z., Yang, F., and Luo, Y. Context embedding based on bi-lstm in semi-supervised biomedical word sense disambiguation. *IEEE Access*, 7:72928–72935, 2019.

[68] Lin, H. and Tegmark, M. Critical behavior in physics and probabilistic formal languages. *Entropy*, 19(7), 2017.

[69] Liu, B. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[70] Liu, B. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press, 2020.

[71] Liu, B. and Zhang, L. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.

[72] Malcolm, N. Wittgenstein's philosophical investigations. *The Philosophical Review*, 63(4):530–559, 1954.

[73] Matthew, E. Peters, mark neumann, mohit iyyer, matt gardner, christopher clark, kenton lee, luke zettlemoyer. deep contextualized word representations. In *Proc. of NAACL*, 2018.

[74] McCallum, A., Nigam, K., et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

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

[76] Mikolov, T. *Language modeling for speech recognition in czech*. PhD thesis, Masters thesis, Brno University of Technology, 2007.

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

[78] Mikolov, T., Kopecky, J., Burget, L., Glembek, O., et al. Neural network based language models for highly inflective languages. In *2009 IEEE international conference on acoustics, speech and signal processing*, pages 4725–4728. IEEE, 2009.

[79] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013b.

[80] Mikolov, T., Yih, W.-t., and Zweig, G. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013c.

[81] Miner, G., Elder IV, J., Fast, A., Hill, T., Nisbet, R., and Delen, D. *Practical text mining and statistical analysis for non-structured text data applications*. Academic Press, 2012.

[82] Mining, W. I. D. Data mining: Concepts and techniques. *Morgan Kaufinann*, 10:559–569, 2006.

[83] Moraes, R., Valiati, J., and Gavião Neto, W. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633, 2013.

[84] Nair, P., Gupta, D., and Devi, B. A survey of text mining approaches, techniques, and tools on discharge summaries. *Advances in Intelligent Systems and Computing*, 1086:331–348, 2021.

[85] Nenkova, A. and McKeown, K. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer, 2012.

[86] Otter, D. W., Medina, J. R., and Kalita, J. K. A survey of the usages of deep learning in natural language processing, 2019.

[87] Pang, B. and Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*, 2004.

[88] Pang, B. and Lee, L. Opinion mining and sentiment analysis. *Comput. Linguist*, 35(2):311–312, 2009.

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

[90] Park, S., Lee, J., and Kim, K. Semi-supervised distributed representations of documents for sentiment analysis. *Neural Networks*, 119:139–150, 2019.

[91] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[92] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[93] Picard, R. W. Affective computing: from laughter to ieee. *IEEE Transactions on Affective Computing*, 1(1):11–17, 2010.

[94] Pohl, M., Hashaam, A., Bosse, S., Staegemann, D., Volk, M., Kramer, F., and Turowski, K. Application of nlp to determine the state of issues in bug tracking systems. volume 2020-November, pages 53–61, 2020.

[95] Prabowo, R. and Thelwall, M. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157, 2009.

[96] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding with unsupervised learning, 2018a.

[97] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training, 2018b.

[98] Rao, C. R. and Vinod, H. D. *Conceptual Econometrics Using R*. Elsevier, 2019.

[99] Rao, G., Huang, W., Feng, Z., and Cong, Q. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57, 2018.

[100] Rosenthal, S., Mohammad, S. M., Nakov, P., Ritter, A., Kiritchenko, S., and Stoyanov, V. Semeval-2015 task 10: Sentiment analysis in twitter. *arXiv preprint arXiv:1912.02387*, 2019.

[101] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[102] Sahlgren, M. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53, 2008.

[103] Sathya, S. and Rajendran, N. A review on text mining techniques. *Int. J. Comput. Sci. Trends Technol.*, 3(5):274–284, 2015.

[104] Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[Scopus website] Scopus website. The largest abstract and citation database of peer-reviewed literature. `https://www.scopus.com`.

[106] Shah, K., Patel, H., Sanghvi, D., and Shah, M. A comparative analysis of logistic regression, random forest and knn models for the text classification. *Augmented Human Research*, 5(1):1–16, 2020.

[107] Sherstinsky, A. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404, 2020.

[108] Simons, G. F. and Fennig, C. D. Ethnologue: Languages of the world, twenty. *Dallas: SIL International. Retrieved from www. ethnologue. com. Accessed in*, 3:2018, 2018.

[109] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. Parsing With Compositional Vector Grammars. In *EMNLP*, 2013.

[110] Steinberger, J., Poesio, M., Kabadjov, M. A., and Ježek, K. Two uses of anaphora resolution in summarization. *Information Processing & Management*, 43(6):1663–1680, 2007.

[111] Sun, C., Qiu, X., Xu, Y., and Huang, X. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer, 2019.

[112] Teodoro, D., Ferdowsi, S., Borissov, N., Kashani, E., Alvarez, D., Copara, J., Gouareb, R., Naderi, N., and Amini, P. Information retrieval in an infodemic: The case of covid-19 publications. *Journal of Medical Internet Research*, 23(9), 2021.

[113] Trappey, A., Chen, P., Trappey, C., and Ma, L. A machine learning approach for solar power technology review and patent evolution analysis. *Applied Sciences (Switzerland)*, 9(7), 2019.

[114] Trask, A., Gilmore, D., and Russell, M. Modeling order in neural word embeddings at scale. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2266–2275, Lille, France. PMLR, 2015.

[115] Tsytsarau, M. and Palpanas, T. Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, 24(3):478–514, 2012.

[116] Turc, I., Chang, M.-W., Lee, K., and Toutanova, K. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*, 2019.

[117] Turney, P. D. and Pantel, P. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[118] Vapnik, V. *The nature of statistical learning theory*. Springer science & business media, 1999.

[119] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

[120] Wang, J., Peng, Y., Lin, Y., and Wang, K. Template based industrial big data information extraction and query system. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10387 LNCS:247–254, 2017.

[121] Wang, X., Yu, G., and Yu, Y. Exploring the marketing strategy for the promotion of a new mobile game: love and producer on sina weibo. *Social Network Analysis and Mining*, 11(1), 2021.

[122] Wilks, Y. and Bien, J. Beliefs, points of view, and multiple environments. *Cognitive Science*, 7(2):95–119, 1983.

[123] Wittgenstein, L. *Philosophical investigations*. John Wiley & Sons, 2010.

[124] Yang, Y., Guo, C., Huang, J., Pan, J., and Li, X. Application of data mining technology in acupuncture prescription compatibility. *Advances in Intelligent Systems and Computing*, 1385 AIST:440–447, 2021.

[125] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[126] Zuccon, G., Koopman, B., Bruza, P., and Azzopardi, L. Integrating and evaluating neural word embeddings in information retrieval. In *Proceedings of the 20th Australasian document computing symposium*, pages 1–8, 2015.