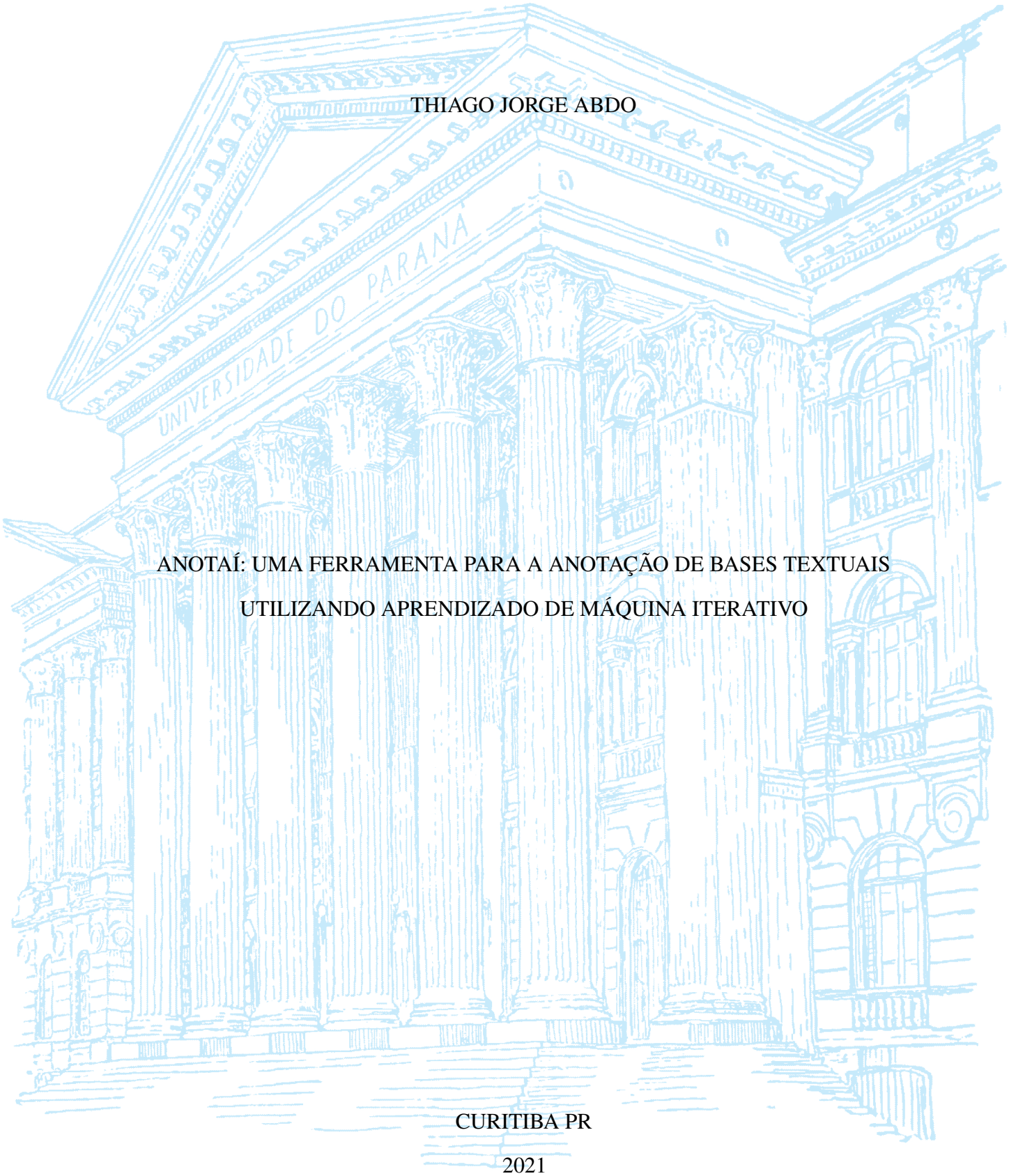
UNIVERSIDADE FEDERAL DO PARANÁ

THIAGO JORGE ABDO



ANOTAÍ: UMA FERRAMENTA PARA A ANOTAÇÃO DE BASES TEXTUAIS UTILIZANDO APRENDIZADO DE MÁQUINA ITERATIVO

CURITIBA PR

2021

THIAGO JORGE ABDO

ANOTAÍ: UMA FERRAMENTA PARA A ANOTAÇÃO DE BASES TEXTUAIS UTILIZANDO APRENDIZADO DE MÁQUINA ITERATIVO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Fabiano Silva.

CURITIBA PR

2021

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

A135a Abdo, Thiago Jorge
ANOTAÍ [recurso eletrônico] : uma ferramenta para a anotação de bases textuais utilizando aprendizado de máquina iterativo / Thiago Jorge Abdo. – Curitiba, 2021.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2021.

Orientador: Fabiano Silva.

1. Aprendizado do computador. 2. Algoritmos computacionais. I. Universidade Federal do Paraná. II. Silva, Fabiano. III. Título.

CDD: 006.31

Bibliotecária: Vanusa Maciel CRB- 9/1928

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **THIAGO JORGE ABDO** intitulada: **ANOTAÍ: Uma Ferramenta para a Anotação de Bases Textuais Utilizando Aprendizado de Máquina Iterativo**, sob orientação do Prof. Dr. FABIANO SILVA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 22 de Abril de 2021.

Assinatura Eletrônica
26/04/2021 14:48:20.0
FABIANO SILVA
Presidente da Banca Examinadora

Assinatura Eletrônica
23/04/2021 09:44:14.0
ADELAIDE HERCÍLIA PESCATORI SILVA
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ - DELLIN/UFPR)

Assinatura Eletrônica
05/05/2021 11:47:46.0
ANDRÉ RICARDO ABED GRÉGIO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
23/04/2021 10:10:59.0
CARLA CANDIDA RIZZOTTO
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ - DECOM)

A minha família e aos meus amigos.

AGRADECIMENTOS

Agradeço a minha família, por não medirem esforços para que pudesse levar meus estudos adiante.

Gostaria também de agradecer às pessoas especiais e aos meus amigos, dos que conheço desde o ensino fundamental aos que conheci apenas durante a graduação, no mestrado ou então no Clube de Dança da UFPR. Obrigado pela paciência, pelo incentivo, pelas danças e pelos bons momentos juntos.

Gostaria de agradecer ao Centro de computação científica e Software Livre (C3SL) pela bolsa que me ajudou durante o mestrado e pelas experiências de vida e profissional que jamais teria em uma empresa privada.

Por fim agradeço, também, a meu orientador, Fabiano Silva, pela paciência, dedicação e ensinamentos que possibilitou que realizasse este trabalho.

RESUMO

Esse trabalho tem como objetivo analisar o uso de diferentes técnicas e algoritmos de aprendizado de máquina aplicados a análise de sentimentos, em especial o estudar do impacto de bases de dados pequenas ou em construção em algoritmos de aprendizado de máquina. Além disso, é proposto um protótipo de uma ferramenta com o intuito auxiliar a criação de novas bases de dados, a qual faz uso das técnicas de aprendizado de máquina analisadas. Para isso é utilizada uma base de dados em construção proposta pelo Departamento de Comunicação da UFPR que consiste em 4333 comentários extraídos de plataformas na internet manualmente anotados em 22 classes diferentes. Para decidir qual técnica de aprendizado de máquina é o ideal para o protótipo de ferramenta e como elas se comporta com uma base de dados pequena foi elaborado 4 experimentos. Os experimentos foram feitos utilizando algoritmos de aprendizagem profunda (Bert) e algoritmos com aprendizagem de custo computacional mais baixo (W2V, Glove e classificadores não lineares) e mostraram que as técnicas de aprendizado de máquina profunda possuem um desempenho maior em relação àquelas que não possuem aprendizado profundo. Entretanto, quando o custo computacional é considerado, as técnicas clássicas de aprendizado de máquina se mostraram mais adequadas para apoiar a ferramenta proposta. Além disso, foram testadas as técnicas de aprendizado de máquina interativo e ativo, para auxiliar no processo de criação de bases de dados na ferramenta. A ferramenta foi desenvolvida com tecnologias bem estabelecidas, possui código aberto e possibilita a construção de bases de texto por um grupo de pesquisadores.

Palavras-chave: análise de sentimentos. aprendizado de maquina. aprendizado profundo.

ABSTRACT

The purpose of this work is to analyse the use of different machine learning approaches and algorithms applied to sentiment analysis and to propose a tool that can be used to create new databases leveraging the techniques that were analyzed. In particular, study the impact with a database with few examples or a database that is being constructed. The database used was proposed by the Departamento de Comunicação da UFPR and is being built with comments extracted from platforms on the internet to achieve this goal. This database has 4333 comments that were annotated in 22 classes. Four experiments were made to decide which machine learning algorithm is the best for the proposed tool and how they perform with a small database. The experiments were done using deep learning algorithms (Bert) and other algorithms that have a lower learning cost (W2V, Glove, and nonlinear classifiers) These experiments showed that deep learning algorithms have a clear performance advantage over techniques that do not have deep learning, but they also have a much higher computational cost, making them inadequate to the proposed tool. Active machine learning and interactive machine learning techniques were also tested to assist the creation of new databases. The tool was developed following current development standards, is open-source, and can be used to speed up the construction of text databases by groups of researchers.

Keywords: sentiment analisys. machine learning. deep learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
W2V	Word-to-vec
BOW	Bag-of-Words
KNN	K-nearest neighbors
PLN	Processamento de linguagem natural
gloVe	Global Vectors
SVM	Máquina de vetor de suporte
MLP	Perceptron multicamada
NB	Naive Bayes
TF-IDF	Frequência do termo–inverso da frequência nos documentos
LSTM	Long-short Term memory
RF	Floresta Aleatória

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento do número de pessoas que possuem acesso à internet, o número de dados sendo gerado todos os dias tem aumentado. Apenas na plataforma de vídeos Youtube, em 2015, foram recebidas 400 horas de vídeo por minuto (Córdova, 2019). Em 2018 cerca de 29 milhões de mensagens foram trocadas por minuto apenas no Whatsapp (Barbosa, 2018). Dessa forma, é inviável para pessoas públicas acompanharem a opinião de internautas e para governos acompanharem como a população tem reagido às políticas públicas. Levando isso em consideração, sistemas de processamento de linguagem natural automáticos ganham cada vez mais importância na decisão de empresas e governos.

Uma das formas de realizar processamento de linguagem natural automática é com aprendizado de máquina, que consiste em mostrar exemplos para um sistema com o intuito de que ele aprenda a classificar esses exemplos. Os tipos de aprendizado podem ser divididos em duas grandes áreas: supervisionado e não supervisionado. No aprendizado supervisionado o sistema faz o processo de aprender, também chamado de treino, recebendo exemplos e a qual classe que eles pertencem. Após o processo de treino ele tem o objetivo de classificar novos exemplos. No aprendizado não supervisionado o sistema realiza o processo de treino apenas recebendo exemplos e, no fim deste processo, o sistema deverá ser capaz de separá-los em classes e conseguir classificar novos exemplos. Dependendo do algoritmo é necessário informar quantas classes existem, ou em quantas o sistema deve tentar separar.

Para que sistemas de processamento de linguagem natural automáticos baseados em aprendizado de máquina possam existir, é necessária a existência de grandes bancos de dados. Esses bancos de dados, no caso de aprendizado supervisionado, precisam ter cada entrada analisada manualmente para poder ser utilizada como exemplo para um sistema de aprendizado de máquina. A grande maioria dos bancos de dados existentes para a criação e análise de performance de sistemas de aprendizado de máquina não está em português. Para exemplificar esse fato podemos comparar a quantidade de bases de dados em português e inglês no *Metatext* [1], um buscador de bases de dados de processamento de linguagem natural, temos 621 bases de dados em inglês e apenas 26 bases em português.

Neste trabalho é proposto um protótipo de ferramenta para apoiar a criação de novos bancos de dados anotados. Esse protótipo conta com sistemas de apoio inteligentes com o objetivo de aumentar a velocidade com que os exemplos são analisados manualmente. O sistema inteligente utiliza recursos de processamento de linguagem natural com aprendizado de máquina para assistir o processo de análise. Para analisar a performance do sistema inteligente foi utilizada uma base de dados classificada em diferentes óticas: desde se o comentário é a favor ou contra o tema do conteúdo do qual ele foi extraído, até se este apresenta características racionais ou de posicionamento.

O protótipo é projetado para ser hospedado em servidores na nuvem e pode ser utilizado na maioria dos computadores e tablets, acessível através de navegadores Web. O sistema inteligente, ou mecanismo de inferência, tem duas funções: sugerir a classificação de exemplos ainda não classificados e escolher quais exemplos devem ser classificados primeiro.

Para isso serão avaliadas diferentes estratégias de aprendizado de máquina, com o objetivo de obter uma estratégia que apresente um bom desempenho e um custo computacional compatível com o ambiente onde o protótipo é executado. Atualmente as estratégias que mostram o melhor resultado em análise de sentimentos são as que contam com aprendizado de máquina

[1] `https://metatext.io/datasets`, acessado 09/05/2021

profundo, (Devlin et al., 2019). Entretanto, essas estratégias possuem um custo computacional elevado para treino e para avaliar exemplos. Estratégias que não fazem uso de aprendizado de máquina profundo como a combinação de extratores de características são: Word2Vec, GloVe; com classificadores: KNN, SVM e RF; possuem um desempenho menor, mas possuem um custo computacional muito menor.

O protótipo foi desenvolvido utilizando técnicas de desenvolvimento web para aumentar a modularidade e manutenibilidade. É possível acessar o protótipo final em *http://anotai.c3sl.ufpr.br* e o código-fonte do projeto está disponível em *http://gitlab.c3sl.ufpr.br/anotai*. Para o mecanismo de inferência foi decidido utilizar o classificador florestas aleatórias combinado com o extrator de características Word2Vec. Essa apresentou o melhor desempenho das testadas e o menor custo computacional, se mostrando a ideal para ser utilizada com o ambiente do protótipo de ferramenta. Além disso, o protótipo permite que mais de um pesquisador trabalhe ao mesmo tempo em uma mesma base de dados, o que pode diminuir a chance de erros humanos.

Este trabalho está organizado da seguinte forma: no capítulo 2 são apresentados os principais conceitos do campo de processamento de linguagem natural; no capítulo 3 é apresentada uma revisão de trabalhos na área de análise de sentimentos e uma breve discussão sobre ferramentas existentes para a criação de bancos de dados; no capítulo 4 são apresentados os experimentos e testes das funções do mecanismo de inferência; no capítulo 5 é apresentado o protótipo de ferramenta para criação de bases de dados e um breve manual das funções existentes; e no capítulo 6 são apresentadas as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são introduzidas as definições e os conceitos do campo de processamento de linguagem natural. Em particular, são apresentadas as principais características da área de análise de sentimentos com aprendizado de máquina.

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

Em (Liddy, 2001), Processamento de Linguagem Natural (PLN) é definido como um conjunto de técnicas para analisar e representar toda e qualquer forma de comunicação entre humanos, com o objetivo de atingir capacidade humana de processamento de linguagem para um conjunto de tarefas ou aplicações. PLN pode ser aplicado em vários sistemas, sendo alguns deles:

- Verificação de ortografia e gramática: Correção automática de texto.
- Reconhecimento de caracteres automatizado: Reconhecimento de texto e transformação em áudio para pessoas com problemas de visão.
- Tradução automática de textos: Tradução de textos sem a necessidade de interferência humana.
- Recuperação de informação: Extração de informações de textos ou frases.
- Classificação de documentos: Divisão de textos em classes pré-definidas.
- Agrupamento de documentos: Divisão de textos em classes não definidas.
- Respostas automatizadas para perguntas: Robôs para responder perguntas frequentes, por exemplo.
- Sumarização: Criação de resumos automaticamente.
- Segmentação de texto/frases: Dividir o texto/frases de acordo com sua função semântica/sintática.
- Sistemas de diálogos: Robôs com capacidade de conversar e não apenas responder perguntas.
- Geração de textos: Gerar textos de forma automática utilizando ou não uma frase ou paragrafo de inspiração.

Neste trabalho o foco é em classificação de documentos, mais especificadamente em análise de sentimentos.

2.1.1 Modelo de Linguagem

Linguagens formais, como linguagens de programação, possuem uma definição precisa de modelo de linguagem. Uma linguagem pode ser definida como uma sequência de caracteres e por um conjunto de regras de como agrupar esses caracteres em palavras e as palavras em frases, chamado de gramática. Por exemplo, "print(2 + 2)" é uma sequência válida em um programa da

linguagem Python, mas "2)+(2 print" não é. Como a quantidade de combinações possíveis de sequências de carácteres para um programa em uma linguagem de programação é infinita, eles não podem ser enumerados. (Russell e Norvig, 2009)

Linguagens naturais também são um tipo de linguagem formal. No entanto, linguagens naturais possuem ambiguidade, uma mesma palavra ou frase podem ter significados diferentes dependendo do contexto. Por exemplo a palavra "pasta"pode estar sendo referida a "pasta de dente", "pasta de trabalho", conjugação do verbo pastar ou até mesmo sinônimo de macarrão. Logo, tanto os modelos de linguagem estatísticos – como Glove (Pennington et al., 2014) – quanto os modelos com redes neurais – como BERT (Devlin et al., 2019) –, serão detalhados no capítulo 4, são apenas aproximações para que um computador possa modelar linguagens naturais.

2.1.2 Desafios

Os desafios em aberto para a área de PNL afetam o desempenho de sistemas de processamento de linguagem natural em todo as tarefas ou aplicações citadas no começão da seção. Em (Kaddari et al., 2021), os autores afirmam que com o progresso de anos recentes ainda temos muitos desafios a serem resolvidos, alguns são intrínsecos a PNL mas outros estão relacionados com as técnicas recentes de aprendizado de máquina, sendo alguns deles:

- Desafios relacionados com o entendimento de linguagem: conseguir entender a língua e como ela se relaciona com o mundo, aprendendo diferentes estilos de escrita e expressão como ironia e sarcasmo.
- Desafios relacionados com a geração de linguagem: falta de coerência e consistência em textos gerados, principalmente em textos mais longos.
- PNL para linguagem com poucos recursos disponíveis: PNL em línguas que não possuem grandes bases de dados disponíveis implica em resultados piores do que as que possuem.
- Bases de dados e métricas de avaliação: existe a necessidade de melhores maneiras de executar avaliação comparativa e métricas de avaliação que possam revelar as limitações dos métodos atuais de PNL.
- PNL explicável: na maioria dos métodos atuais não existe como explicar quais fatores levaram a um decisão, isso é particularmente um problema em áreas como PNL jurídico.
- Detecção de emoção: modelos atuais de PNL não são capazes de interpretar emoções, isso pode ser um problema em sistemas de diálogos quando uma resposta com empatia pode ser necessária por exemplo.
- Entendimento multi-sensorial: Modelos atuais de PNL são predominantemente focados com apenas características linguísticas, sem incluir outras características que utilizamos como: visão, audição e olfato.
- Desafios relacionados com diálogos: além dos desafios dos dois principais componentes de um sistema de diálogos são entendimento de linguagem e geração de linguagem, temos que: ter conhecimentos gerais para poder manter uma conversa, incorporar emoções e sentimentos nas respostas, ter respostas específicas e não apenas as genéricas e manter a coerência com respostas dadas anteriormente.

2.2 APRENDIZADO DE MÁQUINA

Aprendizado de máquina é criar programas que possuem o objetivo de otimizar o desempenho, fazendo o uso de dados. É útil em problemas onde é inviável escrever regras para resolvê-los, mas existe a possibilidade de coletar dados ou resoluções prévias. A tradução de texto falado para escrito é um caso que aprendizado de máquina é necessário. Para humanos não é considerada uma tarefa difícil, mas não conseguimos, sem aprendizado de máquina, criar um programa que consiga realizar essa tradução com o mesma consistência de um humano. (Alpaydin, 2004)

Um conjunto de dados (textos ou frases) coletados forma uma base de dados, que em processamento de linguagem natural também pode ser conhecida como corpus. Bases de dados podem ter apenas textos ou, no caso de análise de sentimentos, cada entrada da base de dados pode ser rotulada com a classe que ela possui.

Dependendo do problema estudado, é necessário que os exemplos da base de dados passem por algum tipo de extração de características antes de serem analisados pelos classificadores, esse processo é geralmente executado por um extrator de características. Esses classificadores são separados em duas categorias: linear e não linear. Classificadores lineares são aqueles que tentam encontrar uma função matemática linear para separar os exemplos dentro do respectivo espaço dimensional. Já os classificadores não lineares tentam encontrar uma função matemática não linear. Por exemplo, em processamento de linguagem natural, é preciso traduzir as palavras ou caracteres de alguma forma para um modelo que possa ser interpretado de forma matemática.

2.2.1 Aprendizado supervisionado

O aprendizado supervisionado, quando utilizado no contexto de classificação de documentos, tenta extrair uma forma de mapear uma entrada para uma anotação de saída, por exemplo: um programa que consegue dizer se o texto de entrada é positivo ou negativo. Também pode ser utilizado no contexto de regressão, que seria quando o objetivo é mapear a entrada para uma saída continua, ou seja, não tenta prever a qual classe a entrada pertence, mas qual o valor da saída, por exemplo, um programa que consegue prever qual a chance de uma pessoa gostar de algum livro.

Em ambos os contextos a qualidade do aprendizado está diretamente atrelada à qualidade dos exemplos de treino. Se a entrada possui exemplos ambíguos, ou seja, exemplos de diferentes classes muito parecidos, a precisão do sistema decorrente do aprendizado será afetada por isso.

Algoritmos comuns para o aprendizado supervisionado no contexto de classificação são classificadores, redes neurais, florestas aleatórias, classificadores Naive Bayes e vizinhos mais próximos (KNNs).

2.2.2 Aprendizado não supervisionado

Aprendizado não supervisionado é inspirado na capacidade de humanos e animais de aprender sem a necessidade de um professor. Através da exposição é possível analisar e reconhecer padrões. Por exemplo, a habilidade de separar maçãs e laranjas pela aparência para isso, não é necessário conhecer os nomes das frutas. (Hinton et al., 1999)

Em alguns algoritmos são compartilhados os aprendizados supervisionado e não supervisionado, como em redes neurais e algoritmos de vizinhos mais próximos. Aprendizado não supervisionado também é usado nos estágios inicias de muitas soluções que fazem uso do aprendizado profundo, como no caso do BERT (Devlin et al., 2019), ou como treinamento para o extrator de características, que é o caso do Word-to-Vec (Mikolov et al., 2013).

2.2.3 Aprendizado profundo

Aprendizado profundo é um modelo computacional composto de várias camadas de processamento que aprendem a representação de um dado com diferentes níveis de abstração. Esse método melhorou o estado da arte em vários domínios, como reconhecimento de fala e objetos. Para aprender a representação dos dados sem informação prévia são necessárias bases de dados maiores quando comparado com métodos que não envolvem aprendizagem profunda. Os modelos podem ser baseados em redes neurais convolucionais ou recorrentes. (LeCun et al., 2015)

As técnicas que envolvem aprendizado profundo precisam consumir uma grande quantidade de dados antes de poderem gerar uma boa representação e são mais complexas, tendo como consequência um treino mais custoso computacionalmente. Portanto, para viabilizar o treino dessas técnicas, é recomendado o uso de placas que possam realizar o processo de treino de forma paralela e eficiente, como placas de processamento de vídeo ou placas específicas para multiplicação de matrizes.

2.2.4 Algoritmos de aprendizado de máquina

Algoritmos são uma sequência de comandos que são executados em ordem para o computador poder atingir o objetivo do programa. Eles são geralmente definidos de forma independente de linguagem de programação e em aprendizado de máquina são usualmente definidos com fórmulas matemáticas. A maioria dos algoritmos de aprendizado de máquina podem ser utilizados sem muitas modificações no aprendizado supervisionado ou não supervisionado.

Uma máquina de vetor de suporte (SVM) tem o objetivo de encontrar uma linha que separa as duas classes de saída no espaço dimensional dos exemplos. O processo de treinamento desses classificadores pode não convergir se eles não possuírem uma linha nesse espaço dimensional que faça a separação das classes. Em problemas nos quais o conjunto de entrada não é linearmente separável, existe a possibilidade de aumentar a dimensionalidade dos exemplos, realizando uma transformação não linear (Boser et al., 1992).

Uma árvore de decisão pode ser interpretada como uma forma de gerar automaticamente um conjunto de regras para mapear o conjunto de entrada para a classe correta. O custo computacional para gerar uma árvore de decisão é baixo e é um dos classificadores que podem ser inspecionados, ou seja, é possível saber o motivo do algoritmo determinar a classe de saída. No algoritmo de treino mais comum não existe a certeza de que a melhor árvore está sendo gerada. A altura da árvore que está sendo gerada afeta diretamente a capacidade de aprendizado do algoritmo, e quanto maior a árvore, maior a chance destas decorarem a base de treino, aprendendo padrões que não existem na realidade. Para resolver os problemas das árvores de decisão foram propostas diferentes maneiras de combinar múltiplas delas, resultando em florestas de decisão (RF) (Hastie et al., 2001).

Classificadores Naive Bayes (NB) são classificadores probabilísticos baseados no teorema de Bayes (D'Agostini, 1995), que é a suposição de que todos os elementos do conjunto de entrada são independentes. Apesar dessa suposição, que nem sempre é verdadeira, os classificadores que fazem uso desse teorema se mostram bastante competitivos com outros classificadores, além de não sofrer com o problema de alta dimensionalidade do conjunto de entrada, uma vez que é linear para o tamanho do espaço dimensional da entrada.

Os algoritmos de vizinho mais próximo fazem a suposição de que exemplos pertencentes a uma mesma classe estão próximos no espaço dimensional da entrada. A implementação exata desse algoritmo é ineficiente, uma vez que é necessário calcular algum tipo de distância para todos os exemplos da base de treino. Após o cálculo da distância para todos os elementos da

base de treino, são escolhidos os K vizinhos mais próximos para realizar uma votação e decidir qual é a classe do elemento, no caso de um problema de classificação. Existem aproximações do algoritmo, que se mostram mais eficientes, que evitam realizar o cálculo da distância para todos os exemplos da base, porém essas implementações podem obter resultados diferentes do que seria obtido com a implementação exata. (Han et al., 2012)

Redes neurais, ou perceptron multicamada (MLP), são a união de perceptrons conectados em série e/ou paralelo. Perceptron, ou neurônio, é um classificador linear binário que mapeia a entrada para a saída através de uma função matemática de primeiro grau. (Rosenblatt, 1958)

Fazendo combinações de camadas de perceptrons com outras funções matemáticas (chamadas de funções de ativação), é possível gerar um classificador não linear, as redes neurais. É possível criar diferentes configurações de redes neurais variando: quantidade de conexões, quantidades de camadas, funções de ativação e quantidades de perceptrons em cada camada. Não existe uma forma determinística de qual configuração é ideal para cada problema.

Para treinar e descobrir qual a função matemática que cada perceptron deve representar para solucionar o problema, é realizado um treinamento com o algoritmo *BackPropagation* (Rumelhart et al., 1988). Esse processo de treino consiste em propagar o erro da classificação de exemplos para os perceptrons da rede. Porém, nesse algoritmo, é necessário calcular o erro médio da classificação de todos os exemplos de uma base de dados, o que é, geralmente, muito lento. Na época que esse processo de treino foi proposto ele era computacionalmente inviável. O Gradiente Descendente Estocástico é o algoritmo que é utilizado na prática, uma aproximação do *BackPropagation*, que divide os exemplos em grupos para realizar o processo de treino ao invés de utilizar o erro médio de todas as classificações. Essa aproximação, em conjunto com a evolução da capacidade computacional, foi o que possibilitou o uso de redes neurais. (Bottou, 1991)

As redes *long short-term memory* (LSTM) são uma variação de redes neurais que tentam trazer melhor desempenho para problemas em que a temporalidade ou a sequência são importantes. A LSTM é uma rede neural que faz parte do grupo de redes neurais recorrentes. As redes neurais recorrentes são redes que possuem conexões que podem formar ciclos ou possuem camadas conectadas com mais de duas camadas. Essa característica permite que essas redes possuam a capacidade de capturar comportamentos temporais. Como essas redes possuem ciclos e, as vezes, a saída do exemplo anterior é utilizado para prever o próximo, pode ficar impossível computacionalmente de propagar o erro de classificação de exemplos pela rede. Para resolver esse problema essas redes possuem mecanismos de controle para diminuir a influência de inferências antigas. No caso da LSTM, esse mecanismo não é estático e é aprendido durante o processo de treino. Essa variação é significativamente importante em áreas como o processamento de fala e composição de música. (Hochreiter e Schmidhuber, 1997)

Outra variação de redes neurais, os codificadores automáticos, possuem o objetivo de encontrar representações com dimensões menores dos exemplos de uma base de dados. Um codificador automático faz isso tentando encontrar redes neurais que conseguem copiar a entrada inicial para a saída. Porém, eles possuem mecanismos para impedir que cada camada interna apenas aprenda a reproduzir a entrada. Um dos mecanismos mais utilizados para garantir que o codificador não está apenas reproduzindo a entrada é forçar que a saída do codificador esteja em um espaço dimensional diferente do que a entrada inicial. Geralmente são construídos utilizando dois módulos, um para codificar a entrada e outro para decodificar. Existem diversas variações que podem ser aplicadas para remover ruídos, transformar a entrada em algo matematicamente interpretável, extrair a parte do conteúdo que é mais importante, entre outros. Uma das variações utilizadas em técnicas de processamento de linguagem natural é chamada de transformador. O transformador é uma variação que, após o treino, utiliza apenas a parte do codificador automático

que codifica a entrada, com o objetivo de converter o exemplo para algo que possa ser processado por outros algoritmos de aprendizado de máquina. (Hinton et al., 2011)

2.2.5 Avaliação e treinamento

Com a quantidade de diferentes métodos, algoritmos e problemas de aprendizado de máquina se faz necessário existirem formas de comparações entre eles que sejam o mais justo possível.

Na definição da maioria das métricas de avaliação, são utilizados os conceitos de classificação positiva, negativa, falsa e verdadeira de um exemplo para uma classe. Quando um exemplo é classificado com a classe "X" ele possui classificação positiva para essa classe e negativa para as outras. Essa classificação pode ter sido feita de forma correta, nesse caso ela é uma classificação verdadeira, caso ela tenha sido feita de forma incorreta, ela é uma classificação falsa. Unindo esses dois conceitos chegamos ao conceito de uma classificação falsa-positiva, falsa-negativa, verdadeira-positiva e verdadeira-negativa.

A forma de avaliação mais simples é a taxa de acerto, ou acurácia. Ela é calculada dividindo a quantidade de exemplos que deram o resultado esperado (verdadeiro positivo ou negativo) pela quantidade total de exemplos. É muito utilizada pela simplicidade, mas possui problemas principalmente quando é utilizada em problemas que a base de dados não é balanceada, podendo mascarar o resultado real e não ser a melhor opção. Por exemplo, em uma base de dados que possuí 100 exemplos e 90 são da classe A e 10 são da classe B, se um algoritmo acertar somente os 90 exemplos da classe A ele possui taxa de desempenho de 90%, mas ele errou 100% dos exemplos da classe B, logo nesse exemplo temos um algoritmo que sempre diz que os exemplos são da classe A. Uma das formas de resolver isso é com a taxa de macro-desempenho, que calcula a taxa de desempenho por classe e então calcula a média delas.

Outra forma de avaliação que é muito utilizada é a taxa F1. A taxa F1 é calculada a partir da média harmônica da precisão e da revocação. A precisão e a revocação de um classificador em relação a uma classe podem ser calculadas utilizando os conceitos de falso negativo, falso positivo, verdadeiro negativo e verdadeiro positivo. Seja FP a quantidade de falso-positivos, FN a quantidade de falso-negativos, TP a quantidade de verdadeiros-positivos e FP a quantidade de falso-positivos, temos que:

$$precisão = \frac{TP}{TP + FP} \tag{2.1}$$

$$revocação = \frac{TP}{TP + FN} \tag{2.2}$$

$$F1 = 2 * \frac{precisão * revocação}{precisão + revocação} \tag{2.3}$$

Durante o processo de treinamento ou a criação de um novo processo de aprendizado de máquina é necessário fazer vários ajustes em parâmetros que afetam o sistema que está sendo treinado. Para ter uma avaliação mais próxima da realidade é convencionada a separação da base de dados em 3 partes: treino, validação e teste. A base de dados separada para treino é geralmente a maior parte da base de dados e é utilizada no processo de aprendizado de um sistema de aprendizado de máquina. A base de dados de validação é uma parte da base de dados que o sistema não vê durante o processo de treinamento e é utilizada para validar os parâmetros utilizados e verificar a performance da base. A base de dados de teste é uma parte da base de dados que o sistema nunca vê, ela existe para garantir que o sistema tem uma performance similar

mesmo com exemplos que não foram utilizados para o ajuste de parâmetros, simulando o uso na vida real.

O *K-fold cross validation* tem o objetivo de atingir o melhor desempenho possível na base de teste sem a necessidade de realizar testes nela, visto que isso iria contra a convenção citada acima. Para atingir esse objetivo, a base de dados de treino e validação é unificada e divida em "K" partes. Então são feitos "K" ciclos de treino e validação, cada vez utilizando uma das partes para validação e as outras para treino. Todos esses ciclos são realizados utilizando a mesma parametrização, assim tentando garantir que o sistema está ajustado de forma genérica e não apenas para um conjunto de exemplos. (Blum et al., 1999)

2.3 APRENDIZADO DE MÁQUINA ITERATIVO

Aprendizado de máquina é uma ferramenta poderosa, contudo, demanda o uso de recursos que muitas vezes não estão disponíveis. O ciclo tradicional para aplicar aprendizado de máquina conta com as seguintes etapas: coletar dados, selecionar características para representar os dados, pré-processar e/ou transformar os dados, escolher uma representação e um algoritmo de treino. Essas etapas precisam de diferentes especialistas para terem maior chance de atingir um bom resultado no final, o que as torna ainda mais custosas.

Para facilitar esse processo é possível utilizar um processo chamado aprendizado de máquina iterativo (AMI), aumentando o número de ciclos, mas diminuindo a duração de cada um deles, fazendo-os forma mais rápida, focada e incremental. Nesse método cada mudança é testada em uma etapa do ciclo, permitindo que o especialista em coleta de dados possa ver em tempo real o efeito de novos conjuntos de dados. (Amershi et al., 2014)

Com isso o AMI democratiza ainda mais o aprendizado de máquina, tornando possível serem criadas interfaces/processos nos quais um usuário que não tem conhecimento sobre aprendizado de máquina possa treinar modelos para uma tarefa específica da qual ele possui conhecimento. O AMI permite isso estruturando o processo de treinamento como um problema de interface humano-computador. (Dudley e Kristensson, 2018)

2.4 APRENDIZADO ATIVO

Aprendizado ativo é um processo no qual um algoritmo escolhe qual será a ordem que os exemplos serão consumidos pelo algoritmo de aprendizado, ou quais serão anotados quando utilizados no processo de criação de uma nova base de dados. O principal o objetivo é atingir níveis de desempenho similares com um aprendizado de máquina supervisionado tradicional mas fazendo o uso de menos exemplos. Aprendizado ativo é especialmente útil em ambientes onde fazer a classificação de exemplos é custosa. (Mohri et al., 2018)

Aprendizado ativo tem se mostrado efetivo com testes empíricos. Mas é necessário tomar cuidado com o seu uso, uma vez que a base de treino se torna diretamente conectada com o algoritmo utilizado para escolher os exemplos. Além disso, os exemplos não são mais escolhidos na distribuição normal dos dados, mas de uma distribuição que depende do algoritmo de escolha. Existem testes que mostram que, quando é trocado o algoritmo de aprendizado, o desempenho pode ser menor do que se o mesmo número de exemplos fosse escolhido de forma aleatória. (Settles, 2009)

2.5 CONSIDERAÇÕES

Nesse capítulo foi introduzida a área de processamento de linguagem natural e apresentada a subárea foco desse trabalho, que é aprofundada no capítulo 3 e com foco em aprendizado de máquina. Foram apresentados os conceitos bases de aprendizado de máquina e os tipos de aprendizado que são retomados no capítulo 3, utilizados na ferramenta de apoio apresentada no capítulo 5 e para os experimentos do capítulo 4. Além disso, no capítulo 4 foram apresentados dois processos de aprendizado de máquina que são utilizadas para apoiar a ferramenta apresentada no capítulo 5.

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados trabalhos de análise de sentimentos no contexto de aprendizado de máquina, uma breve análise de ferramentas para fazer a anotação de bases de texto e processos de aprendizado de máquina.

3.1 ANÁLISE DE SENTIMENTOS

Análise de sentimentos, ou também conhecida como mineração de opinião, é o campo que estuda as opiniões, sentimentos e outros atributos através do texto escrito. Com o crescimento do número de críticas na internet, fórum de discussões, blogs e micro-blogs é inviável ler e analisar todas as fontes que estão influenciando diversas pessoas no mundo. Nossas escolhas e opiniões são em grande parte influenciadas pelas pessoas do nosso círculo de convívio. Sistemas de análise de sentimentos estão cada vez mais comuns por causa da quantidade de fontes existentes que influenciam pessoas a tomarem decisões (Liu, 2012).

Na figura 3.1 temos as etapas de técnicas de aprendizado de máquina em análise de sentimentos: pré-processamento, extração de características e classificação. Na segunda e terceira etapas é quando temos o uso de algoritmos de aprendizado de máquina. A primeira etapa, a de pré-processamento, é um conjunto de regras que possui o objetivo de reduzir os ruídos da entrada, no caso da análise de sentimentos esse conjunto de regras vai depender da origem dos dados, como no caso de comentários extraídos da internet, onde podemos fazer o uso de corretor de erros de digitação ou remoção de links e formatação de texto. A etapa de extração de características tem o objetivo de realizar transformações do texto para alguma estrutura que possa ser interpretada matematicamente. A etapa de classificação possui o objetivo de classificar o exemplo já transformado pela etapa anterior.

As estratégias de aprendizado de máquina para a extração de características que têm melhor funcionado no cenário de análise de sentimentos são as que possuem aprendizado profundo, como no BERT (Devlin et al., 2019) e Elmo (Peters et al., 2018). Embora essas técnicas possam ter um melhor desempenho, elas possuem alguns problemas comuns em técnicas de aprendizado: primeiro que não sabemos realmente como a extração das características acontece; segundo, essas estratégias estão suscetíveis a exemplos adversários (Szegedy et al., 2014), que são exemplos gerados automaticamente para enganarem sistemas inteligentes que geralmente não são enganaria humanos. Além dos problemas comuns à maioria das estratégias, as de aprendizado profundo têm um custo computacional mais elevado em relação às que possuem aprendizado mais simples como W2V e GloVe.

Para realizar a análise de sentimentos utilizando técnicas de aprendizado de máquina é combinada a extração de características com um classificador automático. A quantidade de características extraídas por essas técnicas é muito grande, variando de algumas dezenas a milhares, sendo inviável criar regras de separação de classes manualmente, para isso é utilizado

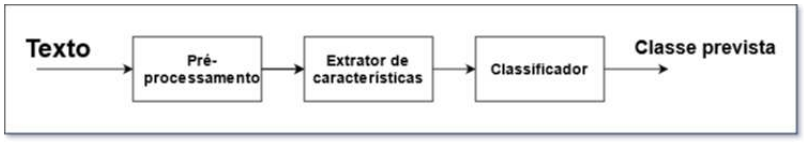


Figura 3.1: Etapas de um processo de máquina que realiza análise de sentimentos

algum classificador automático. Ainda que utilizando técnicas de extração de características e classificação automáticas, é necessário encontrar a melhor parametrização para cada problema.

3.1.1 Extratores de características

Um dos modelos mais simples para extração de características de textos é o Bag-of-words (BOW). Para gerar um vetor de representação nesse modelo, é contado quantas vezes cada palavra do *corpus* existe na frase, não exigindo um processo de treinamento. Outro modelo simples de contagem de palavras é a frequência do termo-inverso da frequência nos documentos (TF-IDF), modelo similar ao BOW, com a diferença de que não é contada a quantidade de vezes que cada palavra aparece na frase, mas sim incrementado um valor que define a importância dessa palavra em relação ao *corpus* utilizado no processo de treinamento. O maior problema desses dois modelos é que eles não guardam a ordem das palavras na frase.

Em (Mikolov et al., 2013), o modelo word-to-vec (W2V) foi proposto por engenheiros da Google para aprender as estruturas de linguagem de um *corpus*. No processo de aprendizado não supervisionado o modelo aprende a gerar uma representação vetorial para cada palavra a partir dos diferentes contextos[1] em que a palavra aparece no *corpus*. As palavras são posicionadas no espaço vetorial de forma que palavras com contextos parecidos fiquem próximas umas das outras no espaço. Esse modelo faz uso de uma rede neural para aprender a probabilidade de cada palavra aparecer próxima de outra no *corpus*. Em (Pennington et al., 2014), o modelo GloVe foi proposto para gerar uma representação vetorial a partir de contagem. Nele, para aprender a representação vetorial, são feitas reduções de dimensionalidade de uma matriz que conta co-ocorrências de pares de palavras. Esse modelo faz o uso de informações estatísticas da bases de dados para reduzir o tempo de duração do processo de treino, realizando o processo de treino somente nos elementos que são diferentes de zero na matriz de co-ocorrência. Mesmo com o treinamento mais rápido o modelo consegue um desempenho similar ao W2V e em alguns casos é melhor.

Em (Peters et al., 2018) foi proposto o extrator de características Elmo. Utilizando aprendizado profundo, esse modelo possui duas estruturas internas independentes, uma com LSTM's conectadas no sentido da sentença e outra conectada no sentido contrário.

Esse modelo conta com um treino em duas etapas: pré-treino e ajuste fino. O pré-treino é feito utilizando aprendizado não supervisionado e faz o uso de grandes massas de dados. Já a etapa de ajuste fino é feita utilizando aprendizado supervisionado e precisa de menos dados para atingir um bom desempenho. Esse tipo de treino em duas etapas possibilita pessoas que não teriam como executar o treino completo de realizar apenas o ajuste fino e conseguir bons resultados.

Uma versão pré-treinada com 5,5 bilhões de símbolos pode ser encontrada no `https://allennlp.org/elmo`. O modelo tem o objetivo de criar um vetor de representação para cada sentença. Nesse modelo, o contexto de cada palavra cria um vetor de representação diferente, ou seja, uma mesma palavra possui vetores diferentes dependendo da frase na qual estão inseridas. Esse modelo, na época do artigo, foi capaz de superar o estado da arte em 6 tarefas.

Em (Radford et al., 2018) foi proposto o extrator de características OpenAI-GPT utilizando aprendizado profundo, que tem como estrutura interna transformadores. Esse modelo utiliza apenas a parte do codificador do transformador, pois o objetivo é apenas extrair características do texto de entrada.

Esse modelo também conta com o processo de treino divido em duas etapas. A estrutura interna dele é um codificador de 12 camadas seguido por uma rede neural com 3072 dimensões

[1] O contexto utilizado pelo W2V também é conhecido por n-gramas

internas. O OpenAI-GPT, na época do artigo, foi capaz de superar o estado da arte em 9 das 12 tarefas testadas.

Em (Devlin et al., 2019) foi proposto o extrator de características BERT utilizando aprendizagem profunda para gerar um modelo de representação natural. Esse modelo inova em ser uma estrutura conectada de forma bidirecional e ter um pré-treino em duas etapas. Além de ser o primeiro modelo de aprendizado profundo capaz de aprender mais de uma língua ao mesmo tempo.

Além da conexão bidirecional desse modelo, todos os neurônios entre duas camadas subsequentes da rede são conectados, dessa forma todos eles têm acesso ao contexto da frase inteira. Com isso, o significado de todas as palavras são afetadas pelo contexto da frase inteira e não apenas das palavras que estão antes dela na frase. Em modelos como o OpenAI GPT (Radford et al., 2019), que é conectado da esquerda para direita, ou como o Elmo (Peters et al., 2018), que possui duas estruturas internas, cada neurônio da rede tem acesso a apenas uma parte do contexto. Dessa forma, o modelo do BERT possui um custo computacional mais elevado para treinar, mas apresenta também melhora considerável em relação a esses dois modelos.

Na primeira etapa de pré-treino o modelo tenta acertar as palavras que foram removidas da frase, de forma que ele aprenda como as palavras são interligadas. Na segunda, avalia se uma frase é sequência de outra, de forma que nessa etapa é aprendido como as frases são interligadas.

Após o pré-treino o modelo estará pronto para passar por um ajuste fino, o qual depende da função para a qual o modelo será empregado. Isso faz com que, apesar do pré-treino ser computacionalmente difícil, ele possa ser executado apenas uma vez, possibilitando ao ajuste fino ser repetido para cada base de dados (ele pode ser concluído em poucos minutos dependendo da base de dados e do hardware disponível).

3.1.2 Classificadores

Em estratégias que fazem o uso de classificadores para definir o sentimento de um documento é possível combinar um ou mais classificadores com um ou mais extratores de características. Nesse caso, é possível fazer inúmeras combinações para se obter um sistema classificador melhor. A combinação que geralmente produz um bom resultado é a combinação de SVM e de um W2V.

Em (Rui et al., 2013) foi utilizado um sistema que conta com dois classificadores, SVM e Naive Bayes, e feita a extração de características usando uma técnica chamada dicionário de intenção. Na base de dados introduzida pelo artigo, essa técnica conseguiu uma precisão média de 71% na classificação do sentimento de documentos. É importante notar que a técnica usada para a extração de características é manual, no caso do dicionário de intenção, e não robusta em relação a palavras desconhecidas.

Em (Kang et al., 2012) foi comparado o uso de SVM e Naive Bayes para a classificação de críticas de restaurantes, além de propor um algoritmo modificado do Naive Bayes, alterado para ser melhor do que o usual no caso de problemas não balanceados. A base de dados utilizada foi gerada a partir de críticas de restaurantes disponíveis na internet de forma binária. A partir do artigo podemos notar que, no caso da base de dados proposta, o algoritmo sugerido ficou pior do que o Naive Bayes na classificação de uma das duas classes, mas melhorou na média da precisão. Não existe menção à disponibilização da base de dados criada para validar os experimentos.

Em (Walker et al., 2012) foi criada uma base de dados a partir de um site de debates, cada documento da base consiste em título de um debate e um argumento. O objetivo é determinar se o argumento é a favor ou contra o debate. Nele são testados classificadores treinados de duas formas, um treinado para ser genérico e outro para ser especialista em um assunto. Além disso, foi feito um teste para determinar qual seria a capacidade de um humano de reconhecer

os argumentos e os autores chegaram a conclusão de que, dependendo do assunto, a precisão humana é de 73% a 85%, enquanto a precisão da máquina foi entre 60% e 75%.

3.1.3 Aprendizado ativo

Aprendizado ativo tem o objetivo de atingir o mesmo desempenho de técnicas que não fazem o uso de aprendizado ativo e utilizando um número menor de exemplos. Para atingir isso, é pressuposto que a ordem na qual os exemplos são apresentados para o modelo de aprendizado de máquina é importante. Além de ser empregado para reduzir a quantidade de exemplos utilizados da base de dados, ele pode ser bastante útil para a criação de novas bases de dados, quando a anotação dos exemplos possui custos elevados.

No geral aprendizado ativo é separado em dois elementos principais, o algoritmo que escolhe os próximos exemplos a serem analisados e um oráculo que consegue prever a classe de cada exemplo. No caso de ser utilizado para reduzir a quantidade de exemplos analisados, o oráculo é a base de dados previamente anotada, já no caso de ser utilizado no processo de criação de uma base de dados, o oráculo pode ser um pesquisador responsável pela anotação de exemplos ou um experimento cujo resultado seja a classe do exemplo. O aprendizado ativo pode ser dividido em três tipos: sínteses de pertencimento, amostragem seletiva baseada em fluxo e aprendizado ativo baseado em conjuntos.

Sínteses de pertencimento é quando o algoritmo de seleção pode perguntar a um oráculo a classe de qualquer exemplo da base de dados e também exemplos gerados pelo algoritmo. Esse tipo de seleção é mais útil quando usado em problemas em que o oráculo não é um pesquisador humano, mas sim um experimento que pode ser automatizado. Amostragem seletiva baseada em fluxo é quando o algoritmo precisa escolher quais entradas serão classificadas pelo oráculo só olhando cada exemplo uma vez. Esse tipo de seleção pode ser utilizado em problemas onde existe um fluxo de dados constante e de grande volume, sendo inviável para guardar todas as entradas e analisá-las mais de uma vez. Aprendizado ativo baseado em conjuntos é quando o algoritmo precisa escolher quais exemplos de um conjunto serão classificados, podendo analisar o conjunto inteiro mais de uma vez, se for necessário. É o mais facilmente utilizado em problemas reais, pois geralmente é simples obter novos conjuntos de dados sem classificação. (Settles, 2009)

Em (Vitório et al., 2019) é apresentada uma comparação de diferentes técnicas de seleção de exemplos, utilizando um algoritmo que é classificado como aprendizagem ativa baseada em conjuntos. Nesse artigo foram comparadas as técnicas de amostragem aleatória, incerteza variável, incerteza variável aleatória, ganho de informação, entropia, entropia variável e entropia variável aleatória.

Para averiguar o desempenho foi utilizado o algoritmo Naive Bayes Multinomial e uma base de dados extraída a partir do Twitter e do Facebook durante o período das eleições presidenciais no Brasil. Um total de 3500 exemplos foram manualmente anotados para possibilitar os testes.

Os autores chegaram a conclusão de que o uso da entropia foi a que obteve melhor acurácia na maioria dos cenários testados por eles. Essa conclusão corrobora com outros experimentos da literatura, inclusive de experimentos que não são da área de análise de sentimentos

Em (Zimmermann et al., 2015) é proposto um algoritmo que permite o uso de aprendizado ativo sem a necessidade de fazer o treino do começo toda a vez que novos exemplos são adicionados. Nele é utilizado um aprendizado ativo por amostragem seletiva baseado em fluxo e, para a classificação, é utilizado o Nayve Bayes Multinomial.

Para medir a eficácia do algoritmo foi feito um teste com as seguintes técnicas de amostragem: completa, aleatória, desativada, por ganho de informação e por incerteza. A amostragem completa simula um ambiente em que todos os novos exemplos são rotulados e

utilizados para o treino. A amostragem aleatória simula um ambiente onde apenas alguns dos novos exemplos serão rotulados, simulando um cenário realista que não faz uso de aprendizado ativo. A amostragem desativada simula um ambiente que não conta com nenhum tipo de aprendizado após o treino inicial. As amostragens por ganho de informação e por incerteza são os dois cenários que fazem o uso de aprendizado ativo.

Os experimentos mostraram que a amostragem por ganho de informação teve o melhor desempenho, utilizando menos da metade das classificações que a amostragem completa. Já a amostragem por incerteza teve o pior desempenho entre as testadas.

Em (Kranjc et al., 2015) é proposta uma plataforma de código aberto que permite a criação de modelos e mineração de dados. É feito o uso de aprendizado ativo de amostragem seletiva baseado em conjuntos. Os autores afirmam estar consumindo dados de um fluxo virtualmente infinito de dados, mas são criados conjuntos a partir desse fluxo de exemplos para fazer a amostragem.

Nessa plataforma o classificador escolhido para realizar a análise de sentimentos foi a máquina de vetor de suporte. É possível customizar o processo de pré-processamento do texto da forma que seja melhor para a fonte de informações escolhidas. Para validar qual seria a melhor forma de escolher quais exemplos seriam classificados por um classificador humano, os autores elaboraram um teste fazendo uso de uma base de dados com 11 mil tweets criada por eles. Com esse teste os autores chegaram a conclusão de que uma combinação de amostragem aleatória e de incerteza é a que possui a melhor performance.

3.2 FERRAMENTA DE ANOTAÇÃO DE TEXTO

As ferramentas de anotação de texto podem possuir diferentes focos, como facilitar o cooperação de pesquisadores e fazer uso de uma técnica de inferência para permitir que o pesquisador trabalhe mais rápido. Geralmente fazer o uso desse tipo de ferramenta faz com que aumente a qualidade das anotações, pois quem está anotando pode focar apenas em anotar e não em armazenar e gerar relatórios (Bontcheva et al., 2010).

3.2.1 Família Gate

A família gate é um projeto de código aberto que possui programas como o Gate Teamware, GATE Developer, Gate Mímir, Gate Embedded e Gate Cloud.

Gate Teamware é uma plataforma web para fazer anotações em projetos de *corpus* de forma colaborativa. Gate Developer é análogo a um ambiente para desenvolvimento de processadores de linguagem que possui vários componentes que podem ser usados para criar um processador de linguagem complexo. Gate Mímir é semelhante a um banco de dados para texto e anotações. Ele permite que sejam criados índices e feita busca pelos dados armazenados. Gate Embedded é uma biblioteca de objetos para acesso aos serviços da família Gate. Gate Cloud é uma plataforma que provê análise de texto como um serviço, com ela é possível utilizar de sistemas treinados com bases conhecidas ou criar novos sistemas.

O Gate Teamware permite que um grupo de pesquisadores trabalhe na anotação de um mesmo projeto e que cada pesquisador tenha uma função específica dentro do projeto, seja como anotador, editor ou gerente de projeto, quem possuem as seguintes responsabilidades:

- Anotador: recebe uma quantidade de documentos e um guia de como fazer as anotações. Cada anotador passa por uma etapa de treino, onde ele recebe textos que já foram anotados, sem ver as anotações, e suas respostas são comparadas com as anotações reais

para ver se ele aprendeu a seguir o guia. Após esse período de teste o anotador começa a receber exemplos reais para anotar e contribuir com a base de dados.

- Editor: é como um gerente dos anotadores, ele tem a função de esclarecer dúvidas dos anotadores e verificar se eles estão seguindo o guia de anotações corretamente.
- Gerente de projeto: tem a função de definir novos guias de anotação, novos projetos de *corpus*, monitorar o progresso dos projetos e lidar com problemas de performance.

De acordo com o que foi reportado no artigo que descreve o projeto, a velocidade de anotação dos documentos cresceu de 15 a 20% e a qualidade das anotações continuou alta nos testes que foram executados, além de permitir que dois anotadores trabalhem em um mesmo documento de forma remota.

3.2.2 Prodigy

O projeto Prodigy pode ser encontrado na `https://prodi.gy`. É um projeto de código fechado mantido pelos criadores da biblioteca de código aberto Spacy (Honnibal et al., 2020). Prodigy tem como função ser uma ferramenta que auxilia na anotação de texto.

Durante o processo de anotar o texto, a ferramenta aprende a fazer as anotações e só pergunta ao usuário os rótulos que a ferramenta não tem certeza de como classificar. Além disso, essa ferramenta permite a customização das etapas de treino e anotação sendo possível criar diversas bases de dados. A ferramenta não permite uso em nuvem e fazer anotações em grupo. Não foi possível executar testes com a ferramenta, pois não existe versão de testes.

3.2.3 Aprendizado de máquina iterativo

A exposição de pessoas não especialistas ou sem conhecimento técnico ao aprendizado de máquina tem sido cada vez mais recorrente, como em aplicações de recomendações ou filtros de correio eletrônico. Essas pessoas, muitas vezes sem saber o que é ou como funciona o aprendizado de máquina, ajudam a melhorar o sistema fazendo correções nas recomendações ou interativamente melhorando o sistema através de ações como: gostar, desgostar, comprar e comentar sobre produtos. Esses usuários geralmente estão fazendo o uso de um sistema de AMI. (Dudley e Kristensson, 2018)

Em (Mishra et al., 2015) são apresentados dois problemas de processamento de linguagem natural com aprendizado de máquina tradicional: as percepções das pessoas sobre fatos está em constante evolução e técnicas que utilizam uma interpretação generalista de palavras pode ter problemas com significados em contextos específicos. Para abordar esses problemas é proposta uma ferramenta que faz uso de AMI, permitindo o treinamento incremental e ajuste manual das regras de classificação.

Essa ferramenta permitiu que os usuários levassem em consideração mudanças na língua, reduzindo as chances da taxa de acerto deteriorar com o passar do tempo e a mudança da língua. Os autores fizeram uma simulação do uso da ferramenta fazendo uso de um sistema treinado em uma base de dados pública originada a partir do Twitter e duas bases menores anotadas manualmente do mesmo contexto. Ao fim da simulação a métrica F1 melhorou em média 3% se comparado sem o uso da ferramenta e 30% se comparado sem o uso do sistema pré-treinado.

Em (El-Assady et al., 2017) é proposto um sistema de reconhecimento de entidades nominal, uma das alternativas de recuperação de informação, de forma visual e interativa. Para fazer o reconhecimento de cada entidade é utilizado aprendizado de máquina com a possibilidade de fazer correções e adaptações de listas de palavras para melhorar a qualidade do modelo

de forma iterativa. O sistema proposto separa as entidades em 10 classes gerais: pessoas, geo-localização, data/hora, palavras de contexto, indicador de polidez, medida, unidade de medida (apenas se não estiver próximo de um número), organizações, indicador de emoção positiva e indicador de emoção negativa.

O principal objetivo da ferramenta é possibilitar a visualização de diálogos em 6 diferentes ângulos: visualização do texto, visualização de entidades, grafo de entidades, grafo de interlocutores e grafos de conceitos. No artigo é feito um estudo de caso com 3 especialistas em ciência política. Os especialistas tiveram novas percepções, tanto de bases de dados anteriormente conhecidas e desconhecidas por eles.

Em (Yimam et al., 2015) é mostrado o impacto de um sistema de aprendizado de máquina no desenvolvimento de uma base de dados de reconhecimento de entidades médicas. Esse reconhecimento é similar ao reconhecimento de entidades nominais, ou seja, dada uma frase, o objetivo é reconhecer qual parte da frase é o transtorno médico e qual é a condição do paciente.

O sistema utilizado conta com o uso de AMI, mais especificamente focado na abordagem de incluir pessoas durante o treinamento, ou seja, durante o processo de anotação, um modelo treinado nas anotações passadas propõe a anotação do documento atual.

Ainda é simulada a eficácia de um sistema usando AMI. Em uma base já anotada o sistema escolhe quais exemplos serão utilizados para aprendizagem. Dessa forma, podemos ver com quantos exemplos anotados o sistema começa a ser útil para agilizar o processo de anotação.

Até 500 exemplos o sistema aumentou rapidamente a métrica F1, atingindo cerca de 50%. Depois disso novos exemplos causaram um impacto menor, sendo que com quase 18.000 a métrica F1 foi de aproximadamente 70%.

Em (Kim et al., 2015) é proposto um método de agrupamento eficiente e interativo, no qual o usuário não precisa ser um especialista em aprendizado de máquina para obter resultados aceitáveis. O sistema permite que sejam feitas mudanças diretas no comportamento do algoritmo para atingir os resultados desejados.

O canal de comunicação com o usuário não exige conhecimento de aprendizado de máquina, sendo feito através de exemplos de entradas e características dos exemplos para gerar os agrupamentos. No artigo é dado o exemplo de um sistema de recomendação de filmes. Uma pessoa pode não achar importante a duração dos filmes, mesmo que isso possa resultar em um bom agrupamento, e achar que o gênero e o número de prêmios do filme seja mais importante e customizar o agrupamento para refletir isso.

3.3 CONSIDERAÇÕES

Nesse capítulo foram apresentadas as mais comuns formas de análise de sentimentos, ferramentas de anotação de texto e técnicas e/ou ferramentas que fazem o uso de aprendizado de máquina em conjunto com o esforço humano.

Os trabalhos que têm abordagem similares ao trabalho aqui proposto: (Walker et al., 2012), (Yimam et al., 2015), (Bontcheva et al., 2010), (Zimmermann et al., 2015) e o Prodigy. No capítulo 5 é proposto um sistema utilizando tecnologias mais acessíveis do que (Bontcheva et al., 2010), que possuem compatibilidade com mais dispositivos. O foco é no suporte à criação de bases de dados e não tanto na criação de métodos de aprendizado de máquina ou a constante extração de novos exemplos, como em (Zimmermann et al., 2015). O sistema é voltado para o uso de técnicas de aprendizado de máquina em análise de sentimentos e não recuperação de informação. como em (Yimam et al., 2015) O aprendizado é feito de forma centralizada, não necessitando a instalação de programas e o código é aberto, diferentemente do Prodigy. No

capítulo 4 são avaliados métodos de análise de sentimentos mais recentes e em diferentes classes se comparados com (Walker et al., 2012).

4 AVALIAÇÃO EXPERIMENTAL

Neste capitulo é apresentado a avaliação experimental para determinar qual o mecanismo de inferência é o adequado a ferramenta proposta no capítulo 5. O mecanismo de inferência tem uma função importante em duas partes na ferramenta: aumentar a velocidade de anotação e selecionar os exemplos a serem anotados em seguida.

Para aumentar a velocidade de anotação o mecanismo de inferência precisa ser capaz de anotar exemplos de forma eficiente do ponto de vista de custo computacional, de hardware e de taxa de acerto. Para poder selecionar os exemplos a serem anotados a literatura mostra que o mecanismo de inferência precisa ser capaz de fornecer uma métrica de o quanto o modelo está certo da classe que ele previu para cada exemplo (Vitório et al., 2019).

O objetivo dos experimentos descritos nesse capítulo é verificar qual a técnica de aprendizado de máquina se encaixa melhor na ferramenta de anotação e o comportamento do aprimoramento de acerto conforme mais exemplos são anotados. Além disso para a execução dos experimentos foi utilizado o *cluster* de computação de alta performance do C3SL. Os experimentos de aprendizado profundo foram executados em nós do cluster que possuem 4 placas gráficas Nvidia® Tesla V100 com 32 Giga de memória HBM2. Os experimentos que possuem aprendizado mais simples foram executados nos nós do cluster que possuem 32 núcleos de processamento Intel® Xeon® CPU E5-4627 v2 e 256 gigas de memória RAM.

4.1 ESCOLHA DA TÉCNICA DE APRENDIZADO DE MÁQUINA

Para definir qual a técnica é a melhor para a ferramenta, fizemos dois experimentos. O primeiro experimento foi feito com 5 diferentes pré-treinos do BERT: pré-treinado apenas com os dados da base de dados, pré-treinado com textos extraídos da internet, pré-treinado com textos extraídos da internet em conjunto com os dados da base de dados, pré-treinado com textos em diferentes línguas pelo Google e pré-treinado com textos em português pelo Neuralmind. O segundo foi feito combinando os extratores de características W2V e GloVe com diferentes classificadores e utilizado 3 conjuntos de treino: apenas os textos da base de dados, textos da bases de dados em conjuntos com textos extraídos da internet e treinado com 17 bases de dados da língua portuguesa (misturando português europeu com brasileiro) (Hartmann et al., 2017).

O primeiro experimento tem como objetivo avaliar o desempenho de cada modelo em relação a métrica F1 e a tempo total de treino do algoritmo com aprendizado profundo BERT e verificar se existe impacto de um sistema treinado com mais de uma língua. O algoritmo BERT foi escolhido para o teste pois quando os testes foram elaborados o BERT era o estado da arte para diversos problemas de processamento de linguagem natural. O segundo experimento tem como objetivo avaliar o desempenho em relação à métrica F1 e o tempo total de treino para algoritmos que não possuem aprendizado profundo como W2V e GloVe.

A base de dados utilizada nos experimentos consiste em comentários extraídos da internet sobre cotas raciais. Alguns exemplos de comentários pode ser vistos na tabela 4.1 A extração desses comentários foi feita no dia da consciência negra, um dia em que esse assunto estava em alta. Essa base de dados teve 4333 comentários anotados manualmente em 22 diferentes atributos (tabela 4.2). A maioria das classes dessa base de dados não estão balanceadas.

Exemplos de comentários da base de dados
Xô mimimi. Vai estudar, porra
Igor Rodrigues mas você acha que o problema está na universidade? O problema está na educação básica, os educadores batem na mesma tecla há anos, é preciso mais investimento na educação básica, todos os governos sempre privilegiaram o ensino superior pois até então ele sempre foi elitista, o pobre não chegava nele, e o ensino básico sempre abandonado, se os alunos do ensino básico público tivessem uma boa educação as cotas nem seriam necessárias, os alunos de todas as classes sociais estudariam na mesma escola.
"Dar tratamento isonômico às partes significa tratar igualmente os iguais e desigualmente os desiguais, na exata medida de suas desigualdades". (NERY JUNIOR, 1999, p. 42). Essas interpretação da lei só cria mais segregação, mais atrapalha q ajuda.
#ConscienciaNegra Ter dia da consciencia negra e necessario sim!!! Em respeito a todos os negros que sofreram ou so... https://t.co/XKn16nmMnT

Tabela 4.1: Exemplos de comentários extraídos da base de dados

Atributo	Possíveis valores
Posicionamento	Neutro (1501), favorável (1298) e contrário (1245)
Racionalidade	Presente (1797) e ausente (1797)
Tema	Fora do tema (1181), relacional (1213), estrutural (937), e desconhecido (1002)
Tema reduzido	Fora do tema (fora do tema e desconhecido) (2183) e dentro do tema (2150)

Tabela 4.3: Classes escolhidas para realizar o experimento e quantidade de exemplos em cada atributo.

Classe	Possíveis classificações
Turno de fala	Novo e resposta
Finalidade relacional	Ausente e Presente
Meta-conversação	Ausente e Presente
Foco no conteúdo do debate	Ausente e Presente
Racionalidade	Ausente e presente
Ameaça	Ausente e presente
Sarcasmo	Ausente e presente
Analogia	Ausente e presente
Narração	Ausente e presente
Insulto	Ausente e presente
Uso de GIFs	Ausente e presente
Uso de Memes	Ausente e presente
Links externos	Ausente e presente
Uso de Emojis	Ausente e presente
Uso de Hashtags	Ausente e presente
Posicionamento	Neutro, favorável, contrário e não se aplica
Gênero do comentarista	Feminino, masculino, outro e não está claro
Tema	Fora do tema, relacional, estrutural e desconhecido
Fonte	Pessoal, mídia, religiosa, científica, outras e não se aplica
Estratégia persuasiva secundária	Retórica propositiva, retórica sedutora, retórica ético-moral, retórica crítica e não se aplica
Estratégia persuasiva dominante	Retórica propositiva, retórica sedutora, retórica ético-moral, retórica crítica e não se aplica
Função da Hashtag	Contexto geral da publicação, decoração, incitação, pertencimento coletivo, ampliação de contexto, afirmação e não se aplica
Forma	Declaração/afirmação, ponto de vista oposto, esclarecimento, questionamento, proposição de solução, chamada para ação e estabelecer conflito

Tabela 4.2: Tabela com as classes e possíveis classes da base de dados utilizada

As três classes que possuem mais exemplos por atributos foram escolhidas para continuar os testes: posicionamento, racionalidade e tema. Para garantir que as classes fossem balanceadas, foi necessário: remover o atributo "não se aplica" da classe "posicionamento", pois essa possuía apenas 289 exemplos; reduzir o atributo "ausente" da classe "racionalidade", pois essa possuía 2536 exemplos (cerca de 41% a mais do que o atributo presente). Além disso, criamos uma nova classe a partir da redução da classe tema. A tabela 4.3 mostra quais são as possíveis classificações para cada classe e a quantidade de elementos.

Cada classe foi tratada como se fosse uma base de dados. Cada base de dados foi, então, separada em 90% para treino e 10% para teste, todos os modelos utilizaram o mesmo conjunto de treino e teste. Além da base, em alguns modelos utilizamos um conjunto de textos extraídos da internet sobre cotas. Esses textos foram publicados na mesma época que os comentários da base foram coletados.

Para o primeiro experimento utilizamos o BERT com diferentes conjuntos de pré-treino. Fizemos o uso do BERT nesse teste por conta da facilidade de encontrar versões com o treino concluído na internet para diferentes idiomas, além de ser o estado da arte para processamento de linguagem natural quando os testes foram executados. Fizemos o uso de 5 modelos: pré-treino com apenas os dados da base, pré-treino com textos sobre cotas e da base de dados, multi-língua disponibilizado pelo Google, mono-língua em português brasileiro disponibilizado pelo Neuralmind (Souza et al., 2020). Todos os modelos passaram por um refinamento do treino apenas com a base de dados.

- Bert 1: pré-treino apenas com a base de dados
- Bert 2: pré-treino com textos sobre cotas extraídos da internet
- Bert 3: pré-treino com textos sobre cotas e base de dados
- Google: multi-língua disponibilizado pelo Google
- Neuralmind: mono-língua disponibilizado pelo Neuralmind

Nos modelos que realizamos o treino completo (Bert 1, 2 e 3), o pré-treino foi testado com variações dos seguintes parâmetros: tamanho das camadas internas, número de camadas, número de mecanismos de atenção e tamanho da entrada. Os valores testados para cada parâmetro são apresentados na tabela 4.4. Todos os modelos gerados com as combinações desses parâmetros foram testados no ajuste fino. Os tamanho de cada camada interna, número de camadas e tamanho da entrada determinam respectivamente: a quantidade de neurônios de cada camada interna, a quantidade de camadas internas e o tamanho máximo da entrada que será processada. Esses parâmetros são comuns à maioria de algoritmos de aprendizagem profunda. O número de mecanismo de atenção é um parâmetro específico do Bert e determina a quantidade de mecanismos de atenção por camada interna. O mecanismo de atenção é um instrumento do Bert que permite que cada palavra analisada por esse mecanismo possua uma conexão de alguma intensidade com outras palavras da frase/texto que está sendo analisado. Outra peculiaridade do Bert é que o tamanho da camada interna precisa ser múltiplo do número de mecanismos de atenção. Os valores iniciais foram inspirados nos valores que o modelo do Google utilizou para pré-treino mas testando variações e reduções por conta do tamanho da base de dados a ser testada e do custo computacional.

Parâmetro	**Valores testados**
Tamanho de cada camada interna	288, 516, 768
Número de camadas	4, 8 e 12
Número de mecanismos de atenção	4 e 12
Tamanho da entrada	256 e 512

Tabela 4.4: Tabela com a variação de cada parâmetro testado no pré-treino dos modelos Bert 1, 2 e 3

Para o ajuste fino foram realizadas as variações nos seguintes parâmetros: tamanho da entrada, taxa de aprendizado e número de épocas. O valores testados para cada parâmetro

são apresentados na tabela 4.5. Esses parâmetros são comuns a maioria dos algoritmos de aprendizagem de máquina. Os tamanhos da entrada foram escolhidos de acordo com a quantidade de elementos da base de dados que tinham no máximo o tamanho da entrada: 54,60% possuem tamanho de até 128, 77,75% de até 256 e, 91,14% de até 512. O número de épocas e a taxa de aprendizado foram definidas de forma empírica, foram realizados testes reduzidos com um número maior de épocas (7) e menor de taxa de aprendizado (2^{-7}) mas os testes completos foram executados apenas com os valores da tabela 4.5.

Parâmetro	**Valores testados**
Tamanho da entrada	128, 256 e 512
Taxa de aprendizado	2^{-3} e 2^{-5}
Número de épocas	3 e 5

Tabela 4.5: Tabela com a variação de cada parâmetro testado no ajuste fino de todos os modelos

Os modelos escolhidos para terem os resultados apresentados na tabela 4.8 são os modelos que tiveram os melhores resultados. Para os modelos que foi feito o pré treino e ajuste fino a melhor configuração de parâmetros está na tabela 4.6, para os modelos que apenas o ajuste fino foi realizado, na tabela 4.7.

	Bert 1	**Bert 2**	**Bert 3**
Tamanho da camada interna	768	288	768
Número de camadas	8	8	8
Número de mecanismos de atenção	4	4	4
Tamanho da entrada no pré-treino	512	512	256
Tamanho da entrada no ajuste fino	512	512	512
Taxa de aprendizado	2^{-5}	2^{-5}	2^{-5}
Número de épocas	5	5	5

Tabela 4.6: Tabela com o melhor conjunto de parâmetros para pré-treino e ajuste fino nos modelos em que o treino completo foi executado

	Google	**Neuralmind**
Tamanho da entrada no ajuste fino	512	512
Taxa de aprendizado	2^{-5}	2^{-5}
Número de épocas	3	3

Tabela 4.7: Tabela com o melhor conjunto de parâmetros para o ajuste fino nos modelos em que apenas o ajuste fino foi executado

	Bert 1	**Bert 2**	**Bert 3**	**Google**	**Neuralmind**
Tema reduzido	88,19%	84,95%	**89,81%**	89,35%	88,89%
Tema	59,76%	60,67%	58,81%	63,07%	**64,81%**
Posicionamento	56,92%	49,17%	59,43%	56,94%	**61,80%**
Racionalidade	74,02%	75,12%	75,39%	77,65%	**79,89%**

Tabela 4.8: Tabela com os valores da porcentagem de acerto de cada modelo (BERT) em cada atributo

Como é possível ver na tabela 4.8, o Neuralmind teve o melhor desempenho em três atributos, apenas no atributo de tema reduzido que ficou atrás do Google. Além disso o Bert 2 e 3 tiveram desempenho melhor do que o Google no atributo de posicionamento sendo que o uso de hardware, tempo e dados para o pré-treino e treino é menor.

Contudo, por conta do custo computacional elevado, algoritmos baseados no BERT não são o ideal para a ferramenta, uma vez que a ferramenta é hospedada em servidores web sem acesso a recursos como placas de video para acelerar o aprendizado. Para ilustrar como o custo computacional elevado é um problema para executar os testes em um hardware não otimizado como um servidor na nuvem, executamos os testes de ajuste fino dos modelos Google e Neuralmind em uma máquina virtual disponibilizada pelo C3SL. Os dois modelos demoraram mais de 28 horas para completar o ajuste fino, tiveram um consumo de 16 Giga de memória RAM e ocuparam todo o processamento da máquina virtual.

Máquinas virtuais com 16 Giga de memória RAM são altamente incomuns e caras, servidores em nuvem mais econômicos possuem menos de 1 giga de RAM. Como podemos ver nos testes de desempenho temos os tempos estimados para a execução de cada etapa, o uso de algoritmos baseados no BERT fica inviável.

Os testes demonstraram que o uso de técnicas como o BERT, que fazem o uso de aprendizagem profunda, não é possível para muitos cenários. O consumo de recursos para completar o treino e o custo de classificar um exemplo são muito maiores do que em alternativas clássicas. O uso de alternativas sem aprendizado profundo ou com aprendizado mais simples possuem o custo computacional mais baixo.

Para testar a viabilidade dessas técnicas foi elaborado um experimento utilizando a combinação de extratores de características e classificadores não lineares. Para obter os modelos testados utilizamos combinações dos classificadores (máquina de vetor de suporte, K vizinhos mais próximos e florestas aleatórias) com os extratores de características (W2V e GloVe). Cada extrator teve 3 combinações de conjuntos de treinos: apenas o conjunto de dados separado para treino da base de dados (representa um extrator sem conhecimento prévio); conjunto de textos sem classificação sobre o mesmo assunto da base de dados combinado com a base de dados (representa um extrator com conhecimento do assunto da base de dados); conjunto de 17 bases de dados na língua portuguesa (representa um extrator com conhecimento da língua). Devido a forma da extração de características dessas combinações foram realizados testes pré-processando a base com: limpeza dos dados (removendo links, números, repetições excessivas de caracteres e estruturas HTML) e correção ortográfica. Na lista abaixo temos os acrônimos para cada combinação de extrator de características com conjunto de treino que são utilizados nas tabelas que reportam desempenho.

- W2V 1: W2V sem conhecimento prévio
- W2V 2: W2V com conhecimento do assunto dos dados
- W2V 3: W2V com conhecimento da língua
- GloVe 1: GloVe sem conhecimento prévio
- GloVe 2: GloVe com conhecimento do assunto dos dados
- GloVe 3: GloVe com conhecimento da língua

Os resultados apresentados nas tabelas 4.9, 4.10, 4.11 e 4.12 foram encontrados utilizando o classificador florestas aleatórias. Além de ter obtido um desempenho melhor, esse

classificador possui um custo computacional mais baixo que máquinas de vetor de suporte (segundo melhor classificador dos testes).

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	85,40%	81,70%	**85,64%**	82,86%	81,44%	79,84%
Tema	52,12%	52,30%	**53,42%**	51,53%	51,79%	48,54%
Posicionamento	47,61%	48,63%	45,70%	50,92%	51,56%	**53,08%**
Racionalidade	**68,58%**	67,47%	67,00%	67,67%	67,23%	63,49%

Tabela 4.9: Tabela com os valores da porcentagem da métrica F1 dos modelos sem aprendizado profundo

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	**84,25%**	83,09%	83,08%	81,94%	80,78%	81,70%
Tema	52,95%	**56,36%**	50,80%	53,01%	47,06%	50,38%
Posicionamento	50,89%	48,97%	49,70%	50,99%	53,59%	**53,96%**
Racionalidade	66,23%	65,44%	**66,36%**	64,05%	65,68%	65,19%

Tabela 4.10: Tabela com os valores da porcentagem da métrica F1 dos modelos sem aprendizado profundo base de dados limpa

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	83,32%	82,62%	**85,87%**	82,15%	82,14%	79,14%
Tema	50,33%	**53,75%**	51,74%	53,64%	49,05%	49,64%
Posicionamento	51,31%	48,48%	46,05%	49,57%	**51,62%**	51,23%
Racionalidade	**69,49%**	64,62%	65,97%	65,24%	68,20%	64,36%

Tabela 4.11: Tabela com os valores da porcentagem da métrica F1 dos modelos sem aprendizado profundo e correção

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	81,94%	80,29%	**84,49%**	82,40%	81,48%	80,32%
Tema	52,56%	**54,79%**	50,14%	51,14%	49,39%	49,52%
Posicionamento	49,38%	50,16%	48,61%	47,98%	**54,83%**	51,20%
Racionalidade	69,11%	66,95%	66,97%	62,39%	**69,67%**	65,71%

Tabela 4.12: Tabela com os valores da porcentagem da métrica F1 dos modelos sem aprendizado profundo base de dados limpa e correção

Comparando os resultados das tabelas 4.9 e 4.11, podemos gerar a tabela 4.13 com as diferenças de cada elemento das duas tabelas. Essa tabela compara a utilização ou não do pré-processamento de correção ortográfica. Podemos ver que no contexto dessa base de dados a correção ortográfica não apresentou resultados consistentes. Em 10 dos 16 testes de modelos que tiveram treinamento do extrator de características com a base de dados (W2V 1 e 2, GloVe 1 e 2) os resultados possuem ganhos ou perdas menores que 2%. O mesmo acontecimento pode ser notado para os modelos em que o extrator de características não teve treinamento com a base de dados, isso pode ser consequência das palavras que foram corrigidas não adicionarem significado aos exemplos, colaborando para confundir o classificador.

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	-2,08%	0,92%	0,23%	-0,71%	0,70%	-0,70%
Tema	-1,79%	1,45%	-1,68%	2,11%	-2,74%	1,10%
Posicionamento	3,70%	-0,15%	0,35%	-1,35%	0,06%	-1,85%
Racionalidade	0,91%	-2,85%	-1,03%	-2,43%	0,97%	0,87%

Tabela 4.13: Tabela com os as diferenças de porcentagem da métrica F1 de cada modelo com a base de dados corrigida em relação a base de dados sem correção

Comparando os resultados das tabelas 4.9 e 4.10, podemos gerar a tabela 4.14 com as diferenças de cada elemento das duas tabelas. Essa tabela compara a utilização ou não do pré-processamento de limpeza dos dados. Podemos ver que no contexto dessa base de dados a limpeza em geral não apresentou ganhos ou perdas consistentes para as bases de "Tema" e "Tema reduzido". No entanto, para a base "Posicionamento", todos os modelos apresentaram alguma melhora de desempenho, e para a base "Racionalidade", 5 dos 6 modelos apresentaram piora no desempenho. Em modelos que não treinaram com a base de dados, a maioria, teve variações de menos de 2% e apenas um modelo teve perda maior que 3% (W2V 3 - 4,73%), porém o modelo Glove 3 teve melhoras em todas as bases. Já nos modelos que treinaram com a base de dados tiveram melhoras em 9 dos 16 casos, em especial o modelo com W2V 2, que teve melhora de 4% na base de posicionamento, e o GloVe 1, com 4,06% na base de tema.

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	-1,15%	1,39%	-2,56%	-0,92%	-0,66%	1,86%
Tema	0,83%	4,06%	-2,62%	1,48%	-4,73%	1,84%
Posicionamento	3,28%	0,34%	4,00%	0,07%	2,03%	0,88%
Racionalidade	-2,35%	-2,03%	-0,64%	-3,62%	-1,55%	1,70%

Tabela 4.14: Tabela com os as diferenças de porcentagem da métrica F1 de cada modelo com a base de dados com limpeza em relação a base de dados sem limpeza

Comparando os resultados das tabelas 4.9 e 4.12, podemos gerar a tabela 4.15 com as diferenças de cada elemento das duas tabelas. Essa tabela compara a utilização ou não da combinação dos dois pré-processamentos (limpeza dos dados e correção ortográfica). Podemos chegar a conclusão de que a combinação da correção ortográfica e com a limpeza da base não apresentou benefícios a utilização de apenas um dos pré-processamentos. Mesmo em casos onde a limpeza e a correção ortográfica tinham um aumento desempenho de pouco mais de 3% (W2V 1 - Posicionamento), a combinação dos dois resultou em um aumento de apenas 1,77%.

	W2V 1	**GloVe 1**	**W2V 2**	**GloVe 2**	**W2V 3**	**GloVe 3**
Tema reduzido	-3,46%	-1,41%	-1,15%	-0,46%	0,04%	0,48%
Tema	0,44%	2,49%	-3,28%	-0,39%	-2,40%	0,98%
Posicionamento	1,77%	1,53%	2,91%	-2,94%	3,27%	-1,88%
Racionalidade	0,53%	-0,52%	-0,03%	-5,28%	2,44%	2,22%

Tabela 4.15: Tabela com os as diferenças de porcentagem da métrica F1 de cada modelo com a base de dados corrigida e com limpeza em relação a base de dados sem correção e sem limpeza

Nesses testes, quando executados na máquina virtual disponibilizada pelo C3SL para hospedar a ferramenta, os testes que mais demoraram para executar foram os que usaram a combinação dos modelos pré-treinados e máquina de vetor de suporte, durando, em média,

2 horas, e utilizaram menos de 1 giga de memória RAM. Os testes com florestas aleatórias demoraram em média 20 minutos e também utilizaram menos de 1 giga de RAM.

4.2 APRENDIZADO DE MÁQUINA ITERATIVO

Aprendizado de máquina iterativo pode ser utilizado tanto no contexto da criação de novas bases de dados quanto da criação ou aperfeiçoamento de um sistema de aprendizado de máquina. No contexto da criação de uma nova base de dados, AMI é utilizado para auxiliar a rotulação dos exemplos.

Para avaliar o impacto do aprendizado iterativo e como se comporta a taxa de acerto de cada modelo realizamos uma simulação. Essa simulação consiste em avaliar o desempenho dos modelos com apenas uma parte de base de dados e a cada rodada adicionar uma quantidade fixa de exemplos novos anotados de forma aleatória. Isso nos permite testar a evolução da taxa de acerto dos modelos como se estivéssemos criando uma base de dados nova sem a necessidade de envolver anotadores humanos, ocorre em uma fração do tempo. A principal diferença dessa simulação para a realidade é o tempo que leva para a inclusão de mais exemplos, sendo que na realidade é maior, pois necessita de um anotador para realizar as novas anotações.

Cada simulação iniciou com 10 exemplos de cada classe anotados e ao fim do processo de treino e teste é adicionado mais 10 exemplos de cada classe até que não existam mais exemplos para serem adicionados. Foram utilizadas as bases de dados de "posicionamento com limpeza" e "tema reduzido com limpeza" da secção 4.1. Nessas simulações são utilizados os extratores de características W2V 3 e GloVe 3 pois esses foram os que tiveram o melhor desempenho um maior número de vezes nos testes da seção 4.1. Por esse mesmo motivo são utilizados os classificadores "florestas aleatórias" (RF) e "máquinas de vetor de suporte" (SVM).

Nas tabelas 4.16 e 4.17 temos o desempenho em determinadas etapas da simulação, com 10, 50, 100, 200, 400, 800 e todos os exemplos de cada classe da parte da base de dados separada para treino. Os valores apresentados nas tabelas são os valores médios de 10 execuções. Como o número de classes de cada base de dados é diferente, a quantidade total de exemplos de cada também difere, mas a quantidade de exemplos por classe é a mesma (exceto a última etapa, que utiliza todos os exemplos da base de treino). Entre parênteses temos a diferença da etapa da linha atual para a linha anterior.

# Exemplos	**W2V + SVM**	**GloVe + SVM**	**W2V + RF**	**GloVe + RF**
30	33,67%	36,27%	37,31%	37,03%
150	39,72% (6,05%)	40,47% (4,20%)	42,04% (4,73%)	41,45% (4,42%)
300	42,88% (3,15%)	44,38% (3,91%)	43,49% (1,45%)	43,62% (2,17%)
600	44,47% (1,59%)	46,25% (1,87%)	46,05% (2,57%)	45,06% (1,44%)
1200	45,79% (1,32%)	48,48% (2,22%)	46,76% (0,71%)	45,47% (0,41%)
2400	51,19% (5,41%)	50,55% (2,07%)	47,58% (0,82%)	47,12% (1,65%)
3640	53,60% (2,41%)	50,99% (0,44%)	47,03% (-0,55%)	47,45% (0,33%)

Tabela 4.16: Tabela com a taxa da métrica F1 conforme é aumentado os exemplos da base de dados de posicionamento com limpeza, entre parenteses temos a diferença da taxa de acerto para o mesmo exemplo na linha anterior

# Exemplos	W2V + SVM	GloVe + SVM	W2V + RF	GloVe + RF
20	61,13%	64,88%	66,01%	72,12%
100	68,16% (7,03%)	67,92% (3,04%)	75,63% (9,62%)	76,90% (4,79%)
200	69,40% (1,24%)	70,39% (2,47%)	77,43% (1,80%)	78,16% (1,26%)
400	71,15% (1,75%)	73,06% (2,67%)	78,23% (0,80%)	79,10% (0,94%)
800	72,22% (1,07%)	77,93% (4,87%)	79,68% (1,44%)	78,88% (-0,22%)
1600	76,39% (4,16%)	80,04% (2,11%)	80,18% (0,50%)	80,71% (1,83%)
3900	81,06% (4,68%)	81,94% (1,90%)	80,73% (0,55%)	81,25% (0,53%)

Tabela 4.17: Tabela com a taxa da métrica F1 conforme é aumentado os exemplos da base de dados de tema reduzido com limpeza, entre parenteses temos a diferença da taxa de acerto para o mesmo exemplo na linha anterior

Como podemos ver nas tabelas 4.16 e 4.17 o impacto da adição de 40 exemplos de cada classe no começo do treinamento (totalizando 50 exemplos) é, em média, maior do que a adição de 200 exemplos de cada classe a outros 200 exemplos (totalizando 400 exemplos de cada classe). A média do aumento da taxa da métrica F1 adicionando os 40 exemplos de cada classe no começo do treinamento é de 4,85% na base de posicionamento e 6,12% na base de tema reduzido ambas com limpeza. Já o aumento médio da métrica F1, adicionando 200 exemplos de cada classe a uma base de dados que já possui 200 exemplos é de 1,86% na base de posicionamento e 1,54% na base de tema reduzido.

Como podemos ver o impacto da adição de novos exemplos diminui com o tamanho da base. Isso acontece por que quando são adicionados 100 exemplos a uma base que antes possuía 10 é aumentado o tamanho da base em aproximadamente 10 vezes. Já quando é adicionado 100 exemplos a uma base que possui 1000, o tamanho da base é aumentado em 0,1 vez. Além disso quanto maior a base, maior a chance do novos exemplos serem muito parecidos com algum já existente e não contribuir para o aumento de desempenho.

A variança padrão começa alta (cerca de 5%) como é esperado para uma base que possui poucos exemplos e com treinos que irão escolher exemplos completamente diferentes. Contudo, conforme são adicionados exemplos a variança padrão vai diminuindo, com 300 exemplos de cada classe em ambos os testes, a variação fica na casa de 1% e continua baixando para menos de 1% na média de todos os modelos.

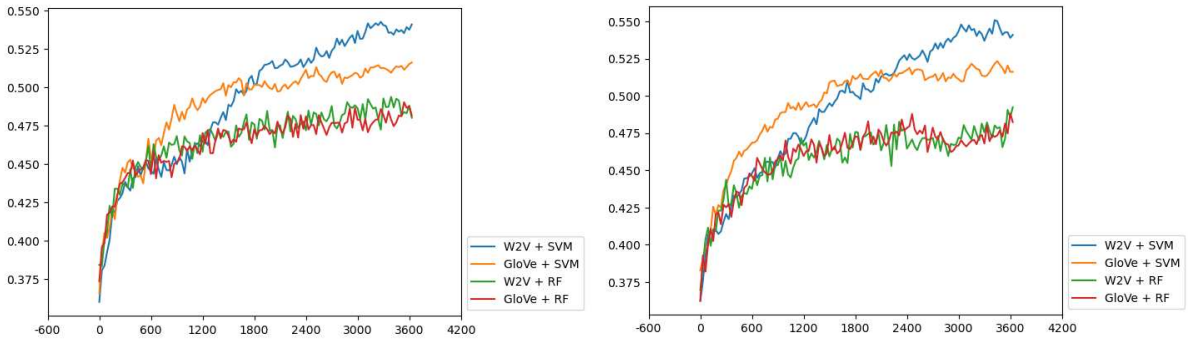
4.3 APRENDIZADO DE MÁQUINA ATIVO

Aprendizado de máquina ativo é uma ferramenta que pode ser muito útil para reduzir a quantidade de exemplos necessários para um sistema que faz uso de aprendizado de máquina atingir o desempenho desejado. No contexto de criação de uma nova base de dados, aprendizado de máquina ativo é útil para reduzir a quantidade de exemplos que precisam ser classificados de forma manual.

Para a ferramenta, descrita na capítulo 5, o aprendizado de máquina ativo tem o objetivo de que o sistema inteligente consiga dar sugestões corretas mais rápido. É importante notar que o aprendizado de máquina ativo, caso não seja anotado a base de forma completa, pode alterar a distribuição padrão dos exemplos e, consequentemente, afetar análises estatísticas no resultado das anotações.

Para avaliar o impacto do aprendizado de máquina ativo e como se comporta a taxa de acerto de cada modelo realizamos uma simulação. Essa simulação foi executada nos mesmos moldes da simulação da seção 4.2, mas os novos exemplos não são adicionados de forma aleatória a cada rodada, são adicionados de acordo com a incerteza que o classificador que está

sendo treinado tem da classe de cada elemento. Essa simulação tem as mesmas vantagens e desvantagens da simulação passada.



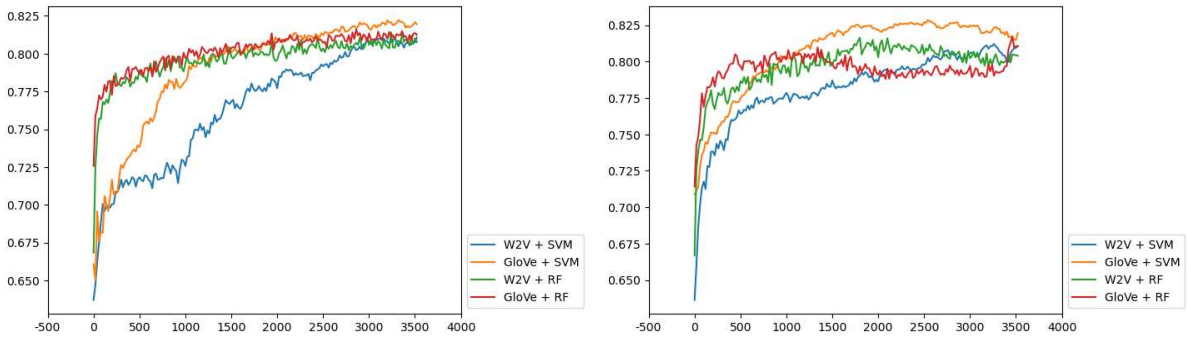
(a) Adicionando aleatoriamente

(b) Adicionando por incerteza

Figura 4.1: Gráficos do desempenho das combinações de classificadores com extratores de características conforme são adicionados novos exemplos na base "posicionamento com limpeza"

Nas figuras 4.1 e 4.2 temos os gráficos da evolução de desempenho médio de cada modelo testado. A variança padrão se comporta da mesma forma do experimento da 4.2. O eixo X é o numero total de exemplos em cada execução. O eixo Y é o desempenho de cada execução podendo variar de 0 a 1, sendo que em 0 significa que a métrica F1 é 0% dos exemplos de teste e 1, 100%.

Na figura 4.1 podemos ver que o o aprendizado de máquina ativo modificou pouco a curva de aprendizado, trazendo pouco ou nenhum ganho de desempenho no começo da execução quando comparado com o aprendizado iterativo sem a escolha dos próximos exemplos. Apesar do impacto ter sido pequeno, existiu uma pequena mudança na curva de aprendizado dos modelos que fazem uso do classificador SVM.



(a) Adicionando aleatoriamente

(b) Adicionando por incerteza

Figura 4.2: Gráficos do desempenho das combinações de classificadores com extratores de características conforme são adicionados novos exemplos na base "tema reduzido com limpeza"

Na figura 4.2 podemos ver que aprendizado de máquina ativo mudou a curva de aprendizado, aumentando a velocidade que os modelos que fazem uso do classificador SVM levaram para aprender. No modelo combinando GloVe com SVM além do modelo aprender mais no começo do treinamento, atingiu o desempenho máximo mais rápido.

4.4 CONSIDERAÇÕES

Neste capítulo foram apresentados experimentos para determinar qual o modelo de aprendizado de máquina é o ideal para a ferramenta proposta no capítulo 5 e qual seria o efeito da utilização de aprendizado ativo e iterativo.

As técnicas de aprendizagem de máquina sem a utilização de aprendizagem profunda foram a combinação dos classificadores SVM e RF com os extratores de características W2V e GloVe com diferentes conjuntos de treinos para os extratores de características, além disso foram testados 2 processos de pré-processamento do texto. As técnicas de aprendizagem de máquina com a utilização de aprendizagem profunda foram técnicas baseadas no BERT, foram testados 3 treinamentos completos com diferentes conjuntos de dados para pré-treino e o ajuste fino em 2 modelos, sendo que um deles foi treinado com bases de dados de diferentes línguas e o outro apenas com bases de dados em português.

Os melhores modelos que fazem o uso de aprendizagem profunda foram em aproximadamente 10% melhores em todas as bases de dados do que os melhores modelos que não fazem uso de aprendizagem profunda. Contudo os melhores modelos que fazem uso de aprendizagem profunda demoraram no mínimo 28 horas para finalizar o ajuste fino no ambiente que seria utilizado pela ferramenta, os tornando inviáveis. Os modelos que não fazem o uso de aprendizagem profunda terminam o treino completo em cerca de 2 horas nos modelos com SVM e 20 minutos nos modelos com RF.

As simulações de aprendizado iterativo confirmaram que grande parte do aprendizado ocorre no começo da base. A simulações de aprendizado ativo mostraram que o SVM teve melhor aproveitamento a reordenação dos exemplos do que o RF nas bases testadas.

Na proposta da ferramenta do capítulo 5 é definido como a ferramenta faz o uso das técnicas de aprendizagem definidas nesse capítulo e da aprendizagem ativa e iterativa.

5 FERRAMENTA DE ANOTAÇÃO

A anotação de documentos de textos é feita manualmente e geralmente sem utilizar as ferramentas adequadas. É possível elaborar uma ferramenta que facilite criar novos *corpus*, melhorando a velocidade e a qualidade das anotações (Bontcheva et al., 2010). Ainda mais com o avanço das técnicas de processamento de linguagem natural, é possível que essa ferramenta realize sugestões de anotação ou até mesmo faça anotações de forma automática depois de receber uma boa quantidade de dados (3.2.2) (Yimam et al., 2015). Mesmo que essas anotações ainda não possam ter a mesma confiança do que um anotador humano, elas poderão servir como detecção de pontos fora da curva e mostrar para os pesquisadores a influência do tamanho da nova base que será proposta na qualidade de um sistema de classificação.

Dentro desse contexto, esse capítulo propõe um protótipo de sistema inteligente de código livre que serve como ferramenta para a criação de novas bases de dados. O protótipo é uma página da internet. O protótipo permite a criação de contas, criação de bases, anotação de documentos de uma base, gerar relatórios de uma base e exportar a base para uso de outros pesquisadores. Dessa forma várias bases de dados podem ser criadas ao mesmo tempo dividindo a mesma infraestrutura.

Além disso, o protótipo tem um mecanismo de inferência capaz de aprender conforme a base de dados é anotada. Por conta de ser uma estrutura compartilhada e em nuvem o mecanismo de inferência possui custo computacional baixo o suficiente para não afetar a usabilidade do protótipo. Durante o processo de anotação o protótipo pode escolher quais os próximos exemplos a serem anotados, escolhendo os exemplos que o mecanismo de inferência tem maior incerteza para classificação, assim a quantidade de exemplos para o mecanismo de inferência conseguir fazer sugestões confiáveis diminui ((Zimmermann et al., 2015), (Vitório et al., 2019)). Conforme a base de dados anotada cresce, o mecanismo de inferência realiza sugestões de anotações mais confiáveis para o pesquisador, aumentando, assim, a velocidade de anotação sem diminuir a qualidade ((El-Assady et al., 2017), (Yimam et al., 2015)).

5.1 OBJETIVO GERAL

Facilitar a criação de bases de dados de linguagem natural utilizando um sistema inteligente que apoia os pesquisadores.

5.2 TECNOLOGIAS

Para desenvolver o protótipo escolhemos o uso da linguagem de programação Ruby e o framework Ruby on Rails (RoR). Esse conjunto de ferramentas foi escolhido pelo vasto ecossistema de componentes já desenvolvidos, chamados de gemas e pelo rápido desenvolvimento de novos componentes para fazer um protótipo funcional (Ruby et al., 2020).

Nesse framework temos o padrão de desenvolvimento MVC (modelo, visualização e controlador) (Burbeck, 1992). Escolhemos nos distanciar de modelos de front-end que fazem a utilização de api, como Angular e React por conta da curva de aprendizado para começar uma aplicação nova e não apresentar uma diferença grande na velocidade de desenvolvimento para uma aplicação pequena com apenas um desenvolvedor. Apesar disso, o código existente pode ser convertido para um API e suportar esses modelos.

5.2.1 MVC

No padrão de desenvolvimento MVC temos a separação de código por funções desempenhadas. A separação é feita em modelo, visualização e controlador. A camada de visualização é responsável pela saída gráfica ou textual da aplicação. A camada de controlador é responsável pela entrada do usuário e de comandar a visualização e o modelo para refletir as mudanças feitas pelo usuário. A camada de modelo é responsável por armazenar e recuperar os dados da aplicação.

Idealmente a camada de visualização não deve se comunicar com a camada de modelo e é fortemente conectada a um bloco da camada de controlador. Na implementação do Ruby on Rails de MVC não é permitido que um bloco da camada de visualização se comunique diretamente com um da camada de Modelo como podemos ver na figura 5.1. Os blocos da camada de modelo não precisam necessariamente possuir um bloco correspondente na camada de visualização ou controlador.

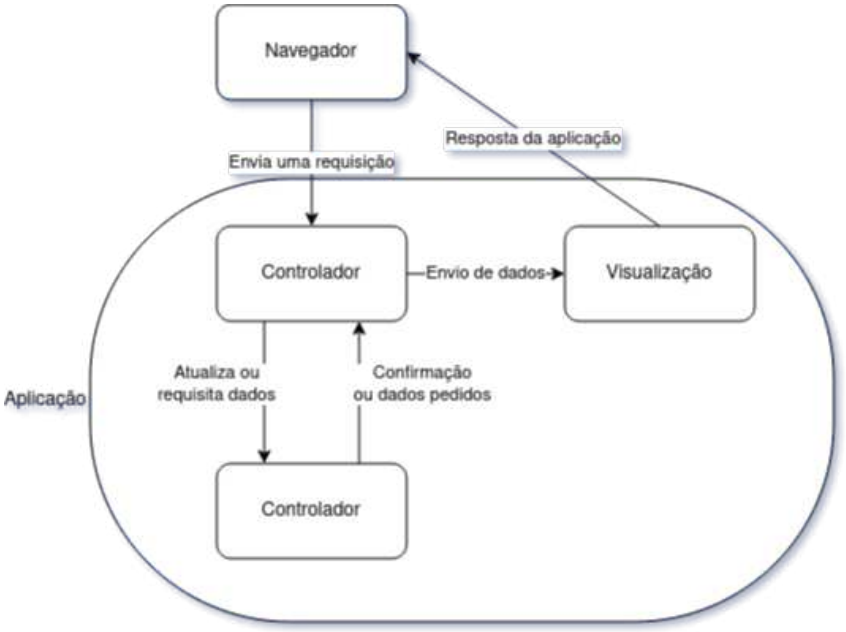


Figura 5.1: Diagrama da implementação de MVC do Ruby on Rails

5.2.2 RoR

O RoR é um framework para a linguagem de programação Ruby para desenvolvimento de aplicativos Web. Além de incluir componentes fundamentais para o desenvolvimento web, RoR conta com módulos que podem facilitar e aumentar a velocidade de desenvolvimento. Esses módulos podem ser desenvolvidos por qualquer desenvolvedor no mundo e instalados com um gerenciador de módulos chamado *bundler*. Eles são chamados de gemas, um aplicativo Web pode usar os módulos que achar necessário.

Por padrão, já é incluído na versão básica do RoR o gerenciador de rotas, compilador de ERB, *bundler* e *scaffold*. O gerenciador de rotas tem a função de interpretar o endereço da página que está sendo acessada e encaminhar os dados para o controlador correto. O compilador de ERB tem a função de permitir a criação de páginas dinâmicas que são compiladas para HTML quando a camada de visualização recebe os dados necessários. O *bundler* tem a função de gerenciar as gemas instaladas e baixar dependências de novas gemas. O *scaffold* tem a função de gerar código padrões para o desenvolvimento de novos blocos nas camadas do MVC.

Além das gemas padrões, estamos utilizando, principalmente, as gemas adicionais: *devise*, *pundit* e *pg*. A gema *devise* tem a função de adicionar mecanismos de autenticação para o aplicativo Web. A gema *pundit* tem a função de adicionar mecanismos de autorização para o aplicativo Web. A gema *pg* tem a função de adicionar os códigos necessários para que os blocos da camada de modelo possam se comunicar com um banco de dados *postgres*.

5.2.3 Outros

As tecnologias descritas nessa sub-sessão são menos importantes e não envolvem codificação para serem substituídas, ou seja, podem ser substituídas facilmente. Além de não afetar a performance para aplicativos Web pequenos e não afetarem o tempo de desenvolvimento. Para o servidor Web foi escolhido o *nginx*, um servidor Web de código aberto, poderia ser substituído pelo Apache ou outro servidor que tenha compatibilidade com o Ruby. Para o banco de dados foi escolhido o *postgres*, um banco de dados de código aberto, poderia ser substituído pelo *mysql* ou outro banco de dados relacional que possua compatibilidade com o Ruby.

5.3 PROTÓTIPO

O protótipo foi desenvolvido com o foco nas funcionalidades mais importantes para atingir o objetivo da ferramenta. As funcionalidades disponíveis atualmente são: criação de usuários, criação de base de dados, adição de exemplos a serem anotados, adição de pesquisadores responsáveis por anotar a base de dados, adição das classes e possíveis valores, anotar a base de dados, a geração da base de dados contendo os exemplos anotados, mecanismo de inferência para escolher os próximos exemplos a serem anotados, mecanismo de inferência para sugerir possíveis anotações. As funcionalidades que poderão ser disponibilizadas no futuro são: ativação de contas por correio eletrônico, melhorar a visualização do estado atual da quantidade de exemplos anotados, uma interface gráfica mais moderna e intuitiva e permitir a utilização de diferentes métodos de inferência.

No protótipo existem três tipos de usuários: identificado (1), anotador de base (2) e o gerente de base (3). Esse tipo de usuário é relativo a um projeto de base de dados, ou seja, um usuário pode ser anotador em projeto e gerente em outro.

1. O usuário identificado não possui permissões de visualização ou edição em uma base de dados. Ele é um usuário do sistema que não foi convidado a participar nem criou um projeto de base de dados.

2. O usuário anotador tem a responsabilidade e permissão de anotar os exemplos de uma base de dados. Ele é um usuário do sistema que foi convidado a participar de um projeto de base de dados.

3. O usuário gerente tem a responsabilidade e permissão de adicionar exemplos para serem anotados, adicionar as classes e possíveis anotações de cada classe, adicionar usuários anotadores no projeto, configurar o mecanismo de inferência, e fazer o download da base de dados anotada. Ele pode, também, atuar como usuário anotador caso exista a necessidade e é um usuário do sistema que criou um projeto de base de dados.

Na figura 5.2 podemos ver possíveis ações dos usuários. O gerente da base de dados, o usuário que criou o projeto, precisa fazer os passos G1.1, G1.2 e G1.3 antes que os anotadores possam começar o processo de anotar a base de dados. O gerente pode a qualquer momento fazer o passo G1.4 para habilitar o uso do sistema de inferência. O anotador tem somente a permissão

e função de anotar a base de dados. Caso o sistema de inferência tenha sido configurado e esteja habilitado o anotador receberá sugestões de anotação durante o processo ou o sistema de inferência poderá sugerir os exemplos conforme uma ordem específica. Um usuário identificado tem duas possíveis ações, criar um projeto de base de dados se tornando o gerente dessa ou receber e aceitar um convite para ser anotador de um projeto. Nas figuras 5.3 e 5.4, é possível ver respectivamente a página inicial de um usuário identificado e a página principal de um projeto de anotação.

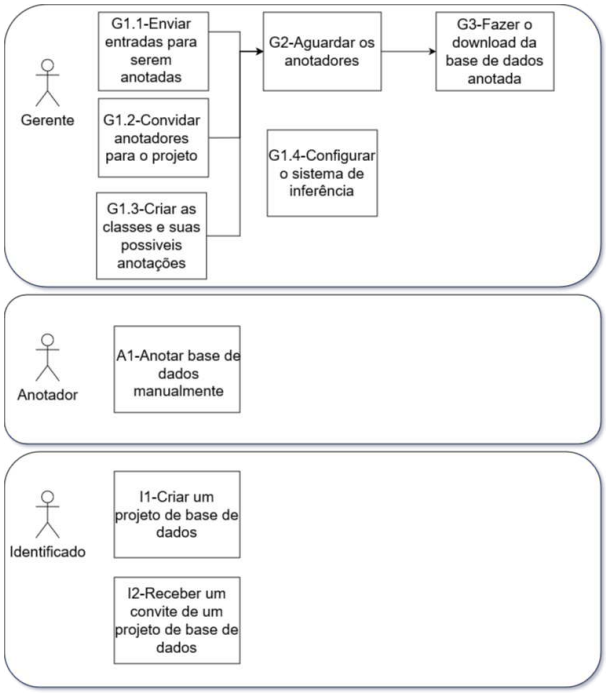


Figura 5.2: Figura com as responsabilidades de cada tipo de usuário do sistema

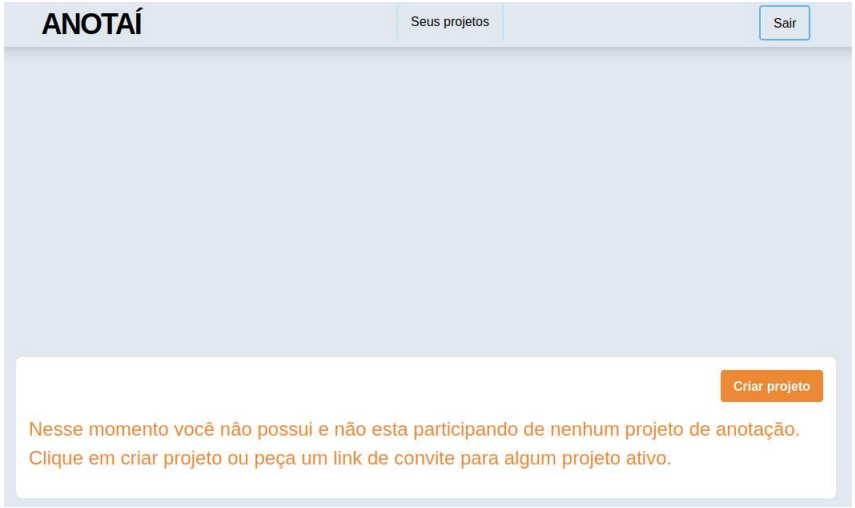


Figura 5.3: Página inicial do protótipo "Anotaí"para usuários identificados

5.3.1 Manual de ações

Um usuário identificado pode realizar as ações: criar um projeto de base de dados (I1) e entrar como anotador em uma base de dados (I2) dentro de um projeto, é possível acessar e

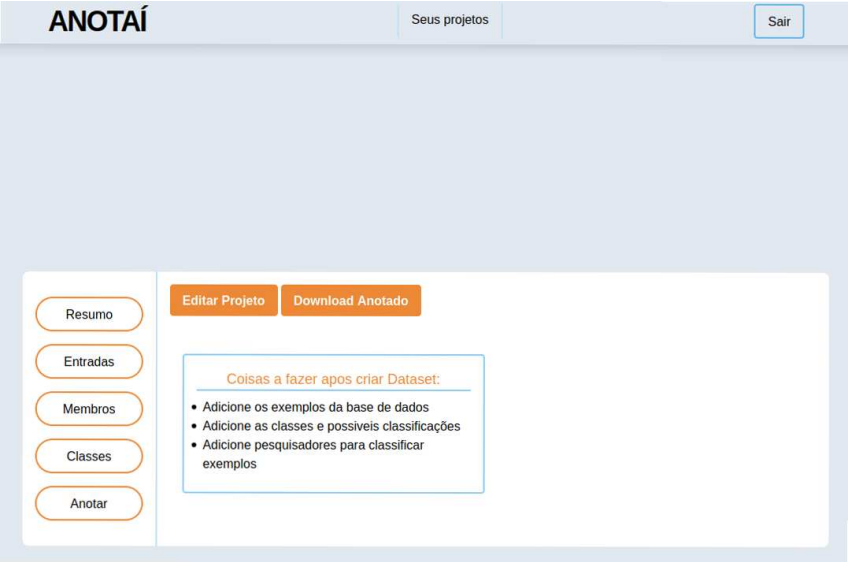


Figura 5.4: Página de um projeto do protótipo "Anotaí"

listar projetos pela página inicial do sistema. Para criar um projeto de base de dados é necessário clicar no botão “Criar projeto” na página inicial, escolher um nome e uma descrição e clicar no botão “enviar”. Para entrar em uma base de dados como anotador é necessário pedir um link de convite para um gerente de projeto.

Um usuário anotador pode apenas realizar a ação de anotar uma base de dados. Para isso, é necessário que ele acesse o projeto na página inicial do protótipo e entre no menu “Anotar” (A1). Uma vez dentro do menu de anotar o usuário deve escolher qual classe irá anotar no momento e começar a anotar.

Um usuário gerente pode realizar as ações G1.1, G1.2, G1.3, G1.4 e G3 dentro de um projeto que pode ser acessado pela página inicial.

Para enviar entradas para serem anotadas (G1.1), o usuário precisa acessar o menu “Entradas”, clicar em “Enviar csv”, selecionar um arquivo CSV (planilha separada por virgulas) contendo um exemplo da base de dados por linha, sem cabeçalhos e clicar no botão “enviar”. Ainda na página de “Entradas” o usuário pode deletar a base inteira clicando em “Deletar Base” (essa ação é permanente) ou gerenciar cada entrada da base, editando ou os deletando. Um arquivo CSV pode ser gerado através de programas comuns de planilhas, como Microsoft Excel e Google Sheets.

Para convidar anotadores para o projeto (G1.2), o usuário precisa habilitar o link de convite dentro do projeto dentro do menu “Membros”. Esse link poderá ser utilizado por uma quantidade ilimitadas de pessoas, após todos terem entrado é possível desabilitar o link ou gerar um link novo, invalidando o antigo. Ainda na página “Membros”, é possível gerenciar os anotadores, sendo possível visualizar e expulsar os usuários do projeto. Caso um usuário seja expulso, todas as anotações dele são perdidas sem possibilidade de recuperar.

Para criar as classes e possíveis classificações (G1.3), o usuário precisa acessar o menu “Classes”, onde é possível gerenciar as classes do projeto. Para criar uma nova classe é necessário clicar no botão “Nova classe”, nessa página é esperado o nome da classe que está sendo criada e é possível adicionar quantas possíveis classificações para essa classe for necessário clicando no botão “Adicionar”, quando o usuário estiver satisfeito com a classe pode concretiza-la no botão “Salvar”. Ainda na página “Classes”, é possível editar (tendo acesso a editar ou remover possíveis classificações) e remover classes, no caso de editar uma classe ou possível classificação todos os

exemplos que já foram classificados são editados e ao remover classes ou possíveis classificações todas as classificações envolvidas são removidas.

Para configurar o sistema de inferência, o usuário precisa acessar o menu “Inferência”. O sistema de inferência vem desabilitado por padrão. O sistema tem duas possíveis funções: sugerir possíveis classificações (aprendizado iterativo) e ordenar os exemplos (aprendizado ativo). Para a sugestão de possíveis classificações, é necessário a taxa de acerto miníma e a quantidade de exemplos. Para ordenar os exemplos, é necessário apenas habilitar. Para ambas as funções é necessário configurar a quantidade miníma de exemplos anotados de cada possível classificação e o desempenho mínimo. Além disso, é possível habilitar o *k-fold cross validation* para cada classe e que possuam menos de uma quantidade de exemplos anotados determinada pelo usuário. Caso o sistema de ordenação esteja desabilitado a ordem padrão dos exemplos segue a ordem em que os exemplos foram inseridos no sistema.

Para fazer o download da base de dados anotada (G3), o usuário precisa acessar o menu “Download Anotado”, na página de “Resumo”, onde é possível configurar a seleção dos atributos e como é considerado que um atributo anotado (quantos anotadores precisam concordar na anotação). Após selecionar de qual classe o gerente deseja fazer o download, é necessário escolher a versão reduzida ou completa e definir o mínimo de anotadores que concordaram em uma classe. Na versão reduzida apenas consta o exemplo e qual a classificação que mais anotadores escolheram como correta que é maior que o mínimo definido. Na versão completa consta o exemplo, qual a classificação que cada anotador escolheu e uma coluna com a classificação que foi escolhida pelo maior número de anotadores que é maior que o mínimo definido, em caso de empate e em caso de nenhuma possível classificação tenha o mínimo definido de anotadores, na coluna de classificação vai constar um X.

5.3.2 Sistema de Inferência

O sistema de inferência, caso habilitado, realiza o treinamento durante a madrugada e traz resultados após isso. Esse sistema possui duas etapas: aprendizado não supervisionado e aprendizado supervisionado. O aprendizado não supervisionado é para treinar o extrator de características, esse treino é executado apenas uma vez ou quando novas entradas são adicionadas ao conjunto de dados e é treinado com o conjunto de dados completo. O aprendizado supervisionado é para treinar o classificador e respeita a separação da base de dados para treino e teste.

O sistema de inferência foi implementado como agente externo, ou seja, o protótipo não depende dele para funcionar e é capaz de funcionar com qualquer implementação que tenha uma interface para realizar chamadas de API. Isso possibilita, com as devidas modificações, que cada projeto de base de dados possua um agente externo ao servidor do protótipo que realiza o treino e teste sendo possível utilizar métodos de aprendizagem com custo computacional mais elevado como aprendizado profunda. A documentação para as modificações necessárias está junto com o código fonte.

O sistema tenta separar 90% de cada classe para treino e 10% para teste dentre os exemplos que estão anotados, caso alguma classe não possua o mínimo de exemplos definido pelo gerente da base para o teste e treino o processo de treino para essa classe não é executado, consequentemente o aprendizado ativo e o iterativo não funcionam. Para testes que possuem poucos exemplos (definido pelo usuário) pode ser executado o *k-fold cross validation* para tentar garantir que a base de dados possua um desempenho mais estável, esse tamanho deve ser definido com cuidado, levando em consideração a quantidade de classes existentes e que o treino é essencialmente repetido “k” vezes.

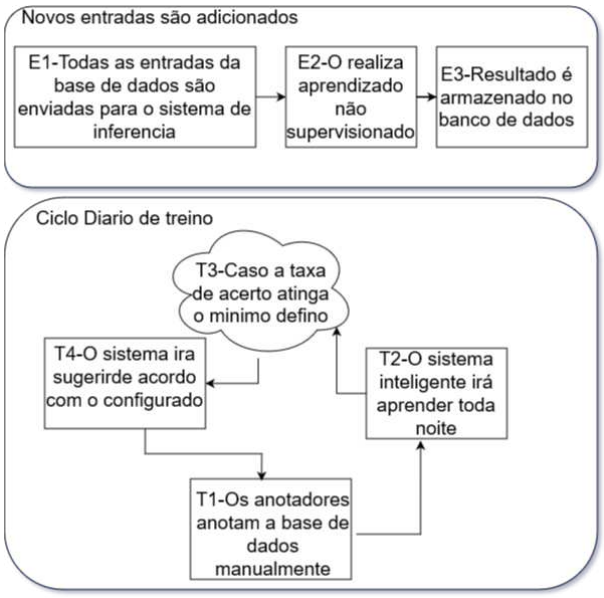


Figura 5.5: Os dois ciclos do sistema de inferência: o ciclo de treinamento e a adição de novos exemplos

Na figura 5.5 temos o ciclo completo do sistema de inferência. Quando são adicionados novos exemplos ao projeto de base de dados o sistema irá retreinar (E1, E2), utilizando todos os exemplos existentes e armazenar o resultado da extração de características no banco de dados (E3). Conforme os anotadores vão anotando novos exemplos, o protótipo vai armazenando as anotações na base de dados (S1). Quando o horário programado para começar um ciclo de aprendizado chega, o sistema busca por todos os exemplos que possuem mais de uma anotação e assume a anotação correta como a que possui mais votos, caso exista discordância entre os anotadores (S2). Para o sistema de inferência cada classe que está sendo anotada é uma base de dados diferente. Utilizando cada base de dados, o sistema treina com a parte de treino da base de dados e com as características armazenadas previamente no banco e é avaliado. Caso ele atinja a performance miníma definida, ele é treinado novamente com a base completa (S3).

Caso apenas o aprendizado iterativo esteja habilitado, a parte da base que não está anotada é enviada para o sistema sugerir anotações e essas anotações são guardadas no banco de dados para serem sugeridas aos anotadores (S4). Caso apenas o aprendizado ativo esteja habilitado, a certeza de cada classificação de cada exemplo também é guardada na base de dados para ser utilizada na escolha do próximo exemplo a ser classificado (S4). Se nenhum dos aprendizados estiverem habilitados, o sistema de inferência não executa nenhum treinamento.

É possível acompanhar a performance do sistema de classificação na página inicial do projeto e na página de classes. Também é possível ver quando foi a ultima vez que o sistema de inferência conseguiu aprender com sucesso.

O uso dessas técnicas de aprendizado pode mudar o resultado da base de dados. O uso do aprendizado iterativo pode enviesar a base de dados, fazendo com que a possível classe que um anotador escolheria para um exemplo mude. O uso de aprendizado ativo pode mudar a distribuição padrão das possíveis classes caso a base de dados não seja anotada por completo, dificultando análises estatísticas.

5.4 CONSIDERAÇÕES

Neste capítulo foi apresentado protótipo de ferramenta para facilitar a criação de novas bases de dados de processamento de linguagem natural, diminuindo a chance de erros em

processos de gerência da base e das anotações de cada pesquisador. Além disso, o protótipo conta com um sistema de inferência que é capaz de sugerir possíveis anotações com base nos exemplos já anotados e também escolher uma ordem de anotação que agregue mais conhecimento para o sistema de inferência.

O protótipo desenvolvido é de código aberto e é uma plataforma WEB que utiliza um modelo de desenvolvimento e um framework atual de programação web. Com isso, permite que qualquer desenvolvedor possa realizar melhoramentos e correções de problemas.

O sistema de inferência é um sistema separado do protótipo, ou seja, não é necessário estar habilitado ou até mesmo estar funcionando para o protótipo ser utilizado. Esse sistema pode ser substituído por outro sistema ou desenvolvido em uma linguagem diferente sem necessidade de alterar o protótipo. A utilização do sistema de inferência pode alterar o resultado final da base, sendo que é possível atrapalhar avaliações estatísticas.

6 CONCLUSÃO

Neste trabalho foi apresentada uma avaliação das principais técnicas de aprendizado de máquina para processamento de linguagem natural, como essas técnicas podem auxiliar no processo de criação de banco de dados e quais técnicas possuem custo computacional compatível com um servidor de nuvem tradicional. Além disso, foi proposta uma ferramenta para criação de bases de dados que pode utilizar as técnicas avaliadas.

As técnicas de aprendizado de máquina para processamento de linguagem natural mais recentes fazem uso do aprendizado profundo. Essas técnicas possuem custo computacional muito elevado para servidores de nuvem. Portanto, técnicas mais antigas e com desempenho inferiores como a combinação de florestas aleatórias com Word2Vec foram escolhidas para acompanhar a ferramenta e serem utilizadas pelo sistema inteligente da ferramenta.

O sistema inteligente da ferramenta utiliza duas técnicas de aprendizado de máquina: aprendizado ativo e aprendizado iterativo. O aprendizado ativo é utilizado para propor uma ordem de anotação dos exemplos com o objetivo de que uma quantidade menor de exemplos anotada seja necessária para que os algoritmos de aprendizado de máquina tenham um melhor desempenho, ou seja, utilizando menos exemplos é obtido um melhor resultado quando comparado com a não utilização do aprendizado ativo. O aprendizado iterativo é utilizado para que o sistema inteligente possa sugerir possíveis classificações durante o período de anotação da base de dados, aumentando, assim, a velocidade de anotação.

Quando comparada com maneiras manuais de criação de bases de dados, além do sistema inteligente, a ferramenta possui outras vantagens como: diminuir a chance de erro humano durante o processo de anotação e gerenciar as anotações feitas, facilitar o processo de adicionar novos exemplos e facilitar para que todos os anotadores trabalhem no mesmo projeto em diferentes plataformas. O sistema utiliza métodos e ferramentas modernas de desenvolvimento para a internet. Possui código aberto e está disponível para visualização e modificações em `gitlab.c3sl.ufpr.br/mestr/anotate-tool`.

6.1 TRABALHOS FUTUROS

Alguns possíveis trabalhos futuros envolvem melhorias na ferramenta e estudos do impacto da ferramenta como:

- Melhorar a interface com o usuário, focando a experiência de usuário.
- Permitir a utilização de um agente externo ao servidor da nuvem para realizar as inferências.
- Incluir técnicas de limpeza da base de dados.
- Estudar o impacto do sistema de aprendizado ativo no resultado da base de dados.
- Estudar o impacto do sistema de aprendizado iterativo no resultado da base de dados.
- Realizar testes com os usuários e com bases de dados maiores.
- Fazer o uso de técnicas de Inteligência Artificial que mostram os motivos que levaram a uma classificação.

REFERÊNCIAS

Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, volume Fourth Edition. The MIT Press, Cambridge, MA 02142-1209.

Amershi, S., Cakmak, M., Knox, W. B. e Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120.

Barbosa, R. (2018). Whatsapp tem 29 milhões de mensagens enviadas a cada minuto, indica pesquisa. `https://www.tudocelular.com/curiosidade/noticias/n133210/whatsapp-29-milhoes-mensagens-cada-minuto.html`. Acessado: 05-09-2019.

Blum, A., Kalai, A. e Langford, J. (1999). Beating the hold-out: Bounds for k-fold and progressive cross-validation. Em *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, página 203–208, New York, NY, USA. Association for Computing Machinery.

Bontcheva, K., Cunningham, H., Roberts, I. e Tablan, V. (2010). Web-based collaborative corpus annotation: Requirements and a framework implementation. *New Challenges for NLP Frameworks*, páginas 20–27.

Boser, B. E., Guyon, I. M. e Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Em *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, páginas 144–152, New York, NY, USA. ACM.

Bottou, L. (1991). Stochastic gradient learning in neural networks. Em *Proceedings of Neuro-Nımes*, Nimes,France. EC2.

Burbeck, S. (1992). Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc). *Softsmarts Inc.*

Córdova, Y. (2019). Como o youtube se tornou um celeiro da nova direita radical. `https://theintercept.com/2019/01/09/youtube-direita/`. Acessado: 05-09-2019.

D'Agostini, G. (1995). A multidimensional unfolding method based on bayes' theorem. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 362(2):487 – 498.

Devlin, J., Chang, M.-W., Lee, K. e Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Em *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, páginas 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dudley, J. J. e Kristensson, P. O. (2018). A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–37.

El-Assady, M., Sevastjanova, R., Gipp, B., Keim, D. e Collins, C. (2017). Nerex: Named-entity relationship exploration in multi-party conversations. *Computer Graphics Forum*, 36(3):213–225.

Han, J., Pei, J. e Kamber, M. (2012). *Data mining: concepts and techniques*. Morgan Kaufmann, Boston, third edition.

Hartmann, N. S., Fonseca, E. R., Shulby, C. D., Treviso, M. V., Rodrigues, J. S. e Aluısio, S. M. (2017). Portuguese word embeddings: Evaluating on word analogies and natural language tasks. Em *XI Brazilian Symposium in Information and Human Language Technology and Collocated Events*, páginas 122–131, Uberlândia, Brazil. SBC, Sociedade Brasileira de Computação.

Hastie, T., Tibshirani, R. e Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Hinton, G. E., Krizhevsky, A. e Wang, S. D. (2011). Transforming auto-encoders. Em *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, página 44–51, Berlin, Heidelberg. Springer-Verlag.

Hinton, G. E., Sejnowski, T. J. et al. (1999). *Unsupervised learning: foundations of neural computation*. Computational Neuroscience Series. MIT press, Cambridge, MA 02142-1209, first edition.

Hochreiter, S. e Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

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

Kaddari, Z., Mellah, Y., Berrich, J., Belkasmi, M. G. e Bouchentouf, T. (2021). Natural language processing: Challenges and future directions. Em Masrour, T., El Hassani, I. e Cherrafi, A., editores, *Artificial Intelligence and Industrial Applications*, páginas 236–246, Cham. Springer International Publishing.

Kang, H., Yoo, S. J. e Han, D. (2012). Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*, 39(5):6000 – 6010.

Kim, B., Glassman, E., Johnson, B. e Shah, J. (2015). ibcm: Interactive bayesian case model empowering humans via intuitive interaction. Relatório técnico, MIT-CSAIL, Cambridge, MA 02142-1209.

Kranjc, J., Smailović, J., Podpečan, V., Grčar, M., Žnidaršič, M. e Lavrač, N. (2015). Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the clowdflows platform. *Information Processing & Management*, 51(2):187–203.

LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

Liddy, E. (2001). *In Encyclopedia of Library and Information Science*. NY. Marcel Decker, Inc., 2nd edition.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. e Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Em *Advances in neural information processing systems*, páginas 3111–3119.

Mishra, S., Diesner, J., Byrne, J. e Surbeck, E. (2015). Sentiment analysis with incremental human-in-the-loop learning and lexical resource customization. Em *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, páginas 323–325.

Mohri, M., Rostamizadeh, A. e Talwalkar, A. (2018). *Foundations of machine learning*. MIT press, Cambridge, MA, USA.

Pennington, J., Socher, R. e Manning, C. (2014). GloVe: Global vectors for word representation. Em *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. e Zettlemoyer, L. (2018). Deep contextualized word representations. Em *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, páginas 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T. e Sutskever, I. (2018). Improving language understanding by generative pre-training. Relatório técnico, OpenAI.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. e Sutskever, I. (2019). Language models are unsupervised multitask learners. Relatório técnico, OpenAI.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Ruby, S., Copeland, D. B. e Thomas, D. (2020). *Agile Web Development with Rails 6*. Pragmatic bookshelf, Raleigh, NC, 2008.

Rui, H., Liu, Y. e Whinston, A. (2013). Whose and what chatter matters? the effect of tweets on movie sales. *Decision Support Systems*, 55(4):863 – 870. 1. Social Media Research and Applications 2. Theory and Applications of Social Networks.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1988). *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, MA, USA.

Russell, S. e Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Settles, B. (2009). Active learning literature survey. Relatório técnico, University of Wisconsin-Madison Department of Computer Sciences.

Souza, F., Nogueira, R. e Lotufo, R. (2020). Bertimbau: Pretrained bert models for brazilian portuguese. Em *Brazilian Conference on Intelligent Systems*, páginas 403–417. Springer.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. e Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations*.

Vitório, D., Souza, E. e Oliveira, A. L. I. (2019). Evaluating active learning sampling strategies for opinion mining in brazilian politics corpora. Em Moura Oliveira, P., Novais, P. e Reis, L. P., editores, *Progress in Artificial Intelligence*, páginas 695–707, Cham. Springer International Publishing.

Walker, M. A., Anand, P., Abbott, R., Tree, J. E. F., Martell, C. e King, J. (2012). That is your evidence?: Classifying stance in online political debate. *Decision Support Systems*, 53(4):719 – 729. 1) Computational Approaches to Subjectivity and Sentiment Analysis 2) Service Science in Information Systems Research : Special Issue on PACIS 2010.

Yimam, S. M., Biemann, C., Majnaric, L., Šabanović, Š. e Holzinger, A. (2015). Interactive and iterative annotation for biomedical entity recognition. Em Guo, Y., Friston, K., Aldo, F., Hill, S. e Peng, H., editores, *Brain Informatics and Health*, páginas 347–357, Cham. Springer International Publishing.

Zimmermann, M., Ntoutsi, E. e Spiliopoulou, M. (2015). Incremental active opinion learning over a stream of opinionated documents. *WISDOM 2015 (KDD'15)*.