t

Pós-graduação em Ciência da Computação

Woshington Valdeci de Sousa

Detecção de hate speech usando combinação de classificadores

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2020

Woshington Valdeci de Sousa

Detecção de hate speech usando combinação de classificadores

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: inteligência computacional
Orientador: George Darmiton da Cunha Cavalcanti

Recife
2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S725d Sousa, Woshington Valdeci de
Detecção de hate speech usando combinação de classificadores / Woshington Valdeci de Sousa. – 2020.
71 f.: il., fig., tab.

Orientador: George Darmiton da Cunha Cavalcanti.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências.

1. Inteligência computacional. 2. Combinação de classificadores. I. Cavalcanti, George Darmiton da Cunha (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2020 - 141

Woshington Valdeci de Sousa

Detecção de hate speech usando combinação de classificadores

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre Profissional em 02 de Junho de 2020.

Aprovado em: 02/06/2020.

BANCA EXAMINADORA

Prof. Luciano de Andrade Barbosa
Centro de Informática / UFPE

Prof. Alexandre Magno Maciel
Universidade de Pernambuco / UPE

Prof. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE
(Orientador)

AGRADECIMENTOS

O desenvolvimento deste trabalho exigiu muito esforço e dedicação deste autor. No entanto, para que essa caminhada chegasse ao fim foi necessário o apoio de diversas pessoas as quais gostaria de agradecer neste momento.

Agradeço primeiramente a Deus, pois sem a vontade dele nada seria possível.

Agradeço a minha familia e amigos que desde sempre me apoiaram e me incentivaram a sempre continuar evoluindo mesmo diante das maiores dificuldades.

Meu orientador, que demonstrou muita habilidade e paciência no direcionamento da pesquisa.

Finalmente, gostaria de agradeçer a minha esposa, Débora Rodrigues, que durante todo o percurso, começando no processo seletivo e concluindo com a finalização deste documento, foi a minha maior incentivadora sendo depois de mim, a que mais sabe o quanto foi dificil essa caminhada.

RESUMO

A popularização da *internet* e o crescimento das mídias sociais, mudaram a perspectiva geral de socialização nos últimos anos, tornando-se um meio de comunicação em massa com pouco controle sobre o conteúdo disseminado em suas plataformas. Além disso, o anonimato e a mobilidade alçaram as mídias sociais, como um importante canal disseminador de discursos de ódio. Esse tipo de discurso tornou-se um problema de escala mundial, causando preocupações em diversos países e organizações. Apesar dos grandes investimentos de empresas como Facebook, Twitter e Youtube, as atividades de controle ainda consistem em moderação manual, tornando o processo inescalável. Um grande número de pesquisas em aprendizagem de máquina, focam na detecção de discursos de ódio em redes sociais, utilizando a classificação monolítica. No entanto, é importante destacar que essa abordagem clássica, geralmente tem sua eficiência limitada no fato que um único classificador é responsável pela generalização do problema. Os sistemas de múltiplos classificadores, buscam combinar um conjunto de classificadores com comportamentos de classificação distintos para superar os resultados individuais dos mesmos. No entanto, selecionar classificadores com comportamentos diversificados, é uma tarefa complexa. Este trabalho propõe uma abordagem de seleção de classificadores baseada em uma exploração visual 2D, criada a partir da redução de dimensionalidade da matriz de dissimilaridade entre todos os classificadores, que é gerada utilizando o coeficiente *double-fault measure*. Para avaliar o método, inicialmente, foi construído um conjunto com 8 algoritmos de aprendizagem distintos, treinados com 5 métodos extratores, perfazendo um total de 40 classificadores, que tiveram seus desempenhos avaliados pelas métricas acurácia e *macro f1-score*. Em seguida, foi aplicada a proposta deste trabalho, para selecionar um grupo de classificadores que apresentavam comportamentos complementares e outro com comportamento mais semelhante, para então avaliar o desempenho destes grupos com técnicas de combinação. Este método foi aplicado em três bases de dados, duas delas (TD e ZW) foram coletadas do *Twitter* e uma (TD+ZW) construída com a junção das outras duas. Nas três bases foram realizados procedimentos de classificação com múltiplas classes, em TD e TD+ZW o objetivo era identificar o padrão como *hate*, *offensive* e *non-offensive* e em ZW classificar os padrões em *racism*, *sexism* e *none*. O método proposto obteve os melhores resultados em duas das bases avaliadas quando comparado a resultados da literatura.

Palavras-chaves: Extração de características. Combinação de classificadores. Detecção de discursos de ódio.

ABSTRACT

The popularization of the internet and the growth of social media have changed the general perspective of socialization in recent years, becoming a means of mass communication with little control over the content disseminated on their platforms. In addition, anonymity and mobility have raised social media as an important channel for disseminating hate speech. This type of discourse has become a worldwide problem, causing concern in several countries and organizations. Despite large investments by companies such as Facebook, Twitter and Youtube, control activities still consist of manual moderation, making the process inescalable. A large number of researches on machine learning focus on detecting hate speech on social networks, using the monolithic classification. However, it is important to note that this classic approach generally has limited efficiency in the fact that a single classifier is responsible for generalizing the problem. Multiple classifier systems seek to combine a set of classifiers with different classification behaviors to overcome their individual results. However, selecting classifiers with diverse behaviors is a complex task. This work proposes a classifier selection approach based on a 2D visual exploration, created from the reduction of the dimensionality of the dissimilarity matrix among all classifiers, which is generated using the double-fault measure coefficient. To evaluate the method, initially, a set was built with 8 distinct learning algorithms, trained with 5 extraction methods, making a total of 40 classifiers, whose performance was evaluated by the metrics accuracy and macro f1-score. Then, the proposal of this work was applied, to select a group of classifiers that presented complementary behaviors and another one with more similar behavior, to then evaluate the performance of these groups with combination techniques. This method was applied in three databases, two of them (TD and ZW) were collected from Twitter and one (TD + ZW) built with the combination of the other two. Classification procedures with multiple classes were carried out in the three bases, in TD and TD + ZW the objective was to identify the pattern as hate, offensive and non-offensive and in ZW to classify the patterns in racism, sexism and none. The proposed method obtained the best results in two of the bases evaluated when compared to results in the literature.

Keywords: Feature extraction. Combination of classifiers. Detecting hate speech.

LISTA DE FIGURAS

Figura 1 – Representação textual pelo modelo BoW. 23
Figura 2 – Visão da arquitetura da segunda etapa dos experimentos, adotata para selecionar e combinar conjuntos de classificadores. 31
Figura 3 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD de acordo com as extrações de características 42
Figura 4 – Quantidade de classificadores (eixo x) que acertam os padrões (eixo y) para o subgrupo *ExtraTrees* (esquerda) e para o subgrupo de CNN (direita). 43
Figura 5 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD independente do método extrator de características. 44
Figura 6 – Projeção comportamental para cada algoritmo avaliado da base ZW com as extrações de características. 49
Figura 7 – Quantidade de classificadores (eixo x) que acertam as instâncias de teste (eixo y) para a base de dados ZW. (1) - SVM, (2) - *LogisticRegression*, (3) - *RandomForest*, (4) - *Naive Bayes*, (5) - MLP, 6 - *ExtraTrees*, (7) - KNN e (8) - CNN 50
Figura 8 – Gráfico de dissimilaridade para cada algoritmo avaliado na base ZW independente do método extrator de características. 51
Figura 9 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD+ZW de acordo com as extrações de características. 55
Figura 10 – Quantidade de classificadores que acertam os padrões para os algoritmos Naive Bayes (esquerda) e MLP (direita). 56
Figura 11 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD+ZW independente do método extrator de características. 57

LISTA DE TABELAS

Tabela 1 – Matriz de ocorrência utilizando o *TF*. 23
Tabela 2 – Distribuição das Instâncias nas Bases de dados. 35
Tabela 3 – Segmentação das bases em treinamento, validação e teste. 35
Tabela 4 – Distribuição dos padrões para cada segmento das Bases de dados. . . . 35
Tabela 5 – Avaliação de hiperparâmetros para a base TD. 40
Tabela 6 – Acurácia do *baseline* de acordo com os hiperparâmetros encontrados e apresentados na Tabela 5. 41
Tabela 7 – Macro F1 do *baseline* de acordo com os hiperparâmetros encontrados e apresentados na Tabela 5. 41
Tabela 8 – *Oracle* para cada subgrupo do Grupo A para a base TD. (A) Significa que os experimentos são do Grupo A. 42
Tabela 9 – *Oracle* para os Grupos B, C e D para a base TD. 45
Tabela 10 – Resultados em termos de acurácia encontrados com as técnicas de combinações para a base TD. 45
Tabela 11 – Resultados em termos de *macro F1* encontrados com as técnicas de combinações para a base TD. 46
Tabela 12 – Avaliação de hiperparâmetros para base ZW. 47
Tabela 13 – Acurácia dos algoritmos avaliados no *baseline* da base de dados ZW de acordo com os hiperparâmetros encontrados e apresentados na Tabela 12. 48
Tabela 14 – Macro F1 dos algoritmos avaliados no *baseline* de acordo com os hiperparâmetros encontrados e apresentados na Tabela 12. 48
Tabela 15 – *Oracle* para cada subgrupo do Grupo A. (A) Significa que os experimentos são do Grupo A. 49
Tabela 16 – *Oracle* para os Grupos B, C e D da base de dados ZW. 52
Tabela 17 – Acurácia alcançada com as técnicas de combinações para os grupos A, B, C e D da base de dados ZW. 52
Tabela 18 – *Macro F1* encontrado com as técnicas de combinações para os grupos A, B, C e D da base de dados ZW. 53
Tabela 19 – Avaliação de hiperparâmetros da base de dados TD+ZW. 53
Tabela 20 – Acurácia da classificação monolítica no baseline de acordo com os parâmetros encontrados e apresentados na Tabela 19 para a base de dados TD+ZW. 54
Tabela 21 – *Macro F1* da classificação monolítica no baseline de acordo com os parâmetros encontrados e apresentados na Tabela 19 para a base de dados TD+ZW. 55

Tabela 22 – *Oracle* para cada subgrupo do Grupo A. (A) Significa que os experimentossão do Grupo A 56
Tabela 23 – *Oracle* dos grupos B, C e D para a base de dados TD+ZW. 58
Tabela 24 – Acurária encontrados com as técnicas de combinações para os grupos A e B da base de dados TD+ZW. 58
Tabela 25 – Macro F1 encontrados com as técnicas de combinações para os grupos A e B da base de dados TD+ZW. 59
Tabela 26 – Melhores resultados encontrados no baseline e na combinação de classificadores para as 3 bases de dados. 59
Tabela 27 – Melhores resultados alcançados em trabalhos anteriores e neste trabalho. 60

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BiLSTM	*Bidirectional long short-term memory*
BoW	*Bag-of-Words*
CBoW	*Continuous Bag-of-Words*
CNN	*Convolutional Neural Network*
EXTRA	*ExtraTrees*
FAST	FastText
GloVe	*Global Vector for Word Representation*
KNN	K Nearest Neighbor
LR	*Logistic Regression*
MCS	*Multiple Classifier Systems*
MLP	*Multi-layer Perceptron*
NB	*Naive Bayes*
NLP	*Natural Language Processing*
PCA	*Principal Component Analysis*
RF	*RandomForest*
RI	Recuperação de Informações
RNN	*Recurrent Neural Network*
SVM	*Support Vector Machine - Linear SVC*
t-SNE	*t-Distributed Stochastic Neighbor Embedding*
TDM	*Term Document Matrix*
TFIDF	*Term Frequency Inverse Document Frequency*
UMAP	*Uniform Manifold Approximation and Projection*
W2V	Word2Vec

LISTA DE SÍMBOLOS

Γ	Base de dados de Treinamento
Λ	Base de dados de Validação

SUMÁRIO

1 INTRODUÇÃO 14
1.1 MOTIVAÇÃO 15
1.2 OBJETIVO GERAL 17
1.3 OBJETIVOS ESPECÍFICOS 17
1.4 ESTRUTURA DO DOCUMENTO 17

2 TRABALHOS RELACIONADOS 19

3 REFERENCIAL TEÓRICO 22
3.1 CLASSIFICAÇÃO DE TEXTO 22
3.1.1 Métodos de Extração de Características 22
3.1.2 *Bag-of-Words* 22
3.1.2.1 Term Frequency - TF 23
3.1.2.2 Term Frequency - Inverse Document Frequency - TFIDF 23
3.1.3 *Word Embedding* 24
3.1.3.1 Word2Vec - W2V 24
3.1.3.2 Global Vector for Word Representation - GloVe 25
3.1.3.3 FastText 25
3.2 MULTIPLE CLASSIFIER SYSTEMS - MCS 25
3.2.1 Diversidade e Oracle 26
3.2.2 Redução de Dimensionalidade 27
3.2.3 Regras de Combinação 27
3.2.3.1 Average 28
3.2.3.2 Product 28
3.2.3.3 Maximum 28
3.2.3.4 Minimum 28
3.2.3.5 Median 28
3.2.3.6 Majority Vote 29

4 METODOLOGIA PROPOSTA 30
4.1 MATRIZ DE DISSIMILARIDADE 31
4.2 REDUÇÃO DE DIMENSIONALIDADE 32
4.3 ANÁLISE DE SENSIBILIDADE 32
4.4 COMBINAÇÃO DE CLASSIFICADORES 33

5 METODOLOGIA DOS EXPERIMENTOS 34
5.1 BASES DE DADOS 34

5.2 MÉTRICAS 36
5.3 CONFIGURAÇÃO DOS EXPERIMENTOS 36
5.3.1 Criação do baseline **36**
5.3.1.1 Pré-processamento e extração de características 37
5.3.1.2 Hiperparâmetros e treinamento dos classificadores 38
5.3.2 Aplicação do método **39**

6 RESULTADOS 40
6.1 BASE DE DADOS TD 40
6.2 BASE DE DADOS ZW 46
6.3 BASE DE DADOS TD+ZW 53
6.4 CONSIDERAÇÕES FINAIS 59

7 CONCLUSÃO 62
7.1 CONSIDERAÇÕES FINAIS 62
7.2 CONTRIBUIÇÕES 63
7.3 TRABALHOS FUTUROS 64

REFERÊNCIAS 66

1 INTRODUÇÃO

As redes sociais tornaram-se um importante canal de comunicação com crescimento exponencial ao longo dos últimos anos, sendo importante em diversos contextos, pois permitem aos usuários expressar opiniões, compartilhar conteúdo, propagar notícias, realizar campanhas publicitárias, entre outros. Além disso, causam impacto em milhões de usuários todos os dias e mudaram a perspectiva geral de socialização e personalização (Batool *et al.*, 2012).

Uma característica importante sobre as mídias socias é que oferecem um ambiente de comunicação em massa, permitindo que os usuários registrem-se rapidamente e propaguem conteúdo dos mais diversos assuntos e intenções (FAUZI; YUNIARTI, 2018).

Devido à grande quantidade de conteúdo publicado nessas plataformas, a tarefa de monitoramento torna-se árdua, dificultando o controle e proporcionando liberdade para que usuários realizem *cyberbullying*[1] ou outras formas de ataques. Segundo (ZHAO; ZHOU; MAO, 2016), as mídias sociais são lugares onde usuários estão constantemente associados ao *bullying*[2], podendo ser vítima, agressor ou apenas um usuário que presencia o fato.

Nesse cenário, torna-se muito comum a disseminação de *hate speech*, discurso de ódio, pois qualquer pessoa com uma conexão mínima de *internet* é um agente disseminador em potencial. Além disso, usuários utilizam estratégias como a criação de *fake profile* para evitar a identificação e consequentemente aumentar a confiança de que não serão punidos. Nesse contexto, destaca-se ainda o surgimento de *bots*, que são algoritmos programados para se comportarem como seres humanos, para realização de tarefas como disseminação de noticias e informações, e portanto podem atuar como agentes disseminadores de *hate speech* (HOWARD; KOLLANYI; WOOLLEY, 2016). De acordo com dados disponibilizados pelo *Twitter*, aproximadamente 8,5% dos usuários da plataforma são *bots*(Subrahmanian *et al.*, 2016). Diversas pesquisas têm sido realizadas na tentativa de caracterizar não apenas o *hate speech*, mas todo o contexto que envolve a construção desses discursos. Como exemplo desses contextos, destacam-se incidentes com grande repercussão social que provocam um aumento considerável na quantidade de discursos de ódio nas plataformas online, as eleições presidênciais americanas de 2016 e o referendo da saída do Reino Unido da União Europeia (UE) (DAFTARY; DEVEREUX; ELLIOTT, 2020), ataques terroristas ligados a grupos islâmicos (SCHARWÄCHTER; MÜLLER, 2020) e a pandemia de covid-19, que provoca discriminação contra chineses e asiáticos em geral (HU; WANG; LU, 2020), são exemplos de eventos que impulsionaram esse tipo de discurso.

Diversas definições sobre *hate speech* estão presentes na literatura, destacando-se (Watanabe; Bouazizi; Ohtsuki, 2018), que definem como uma forma particular de linguagem

[1] Ofensas repetitivas proferidas por meios eletrônicos (PATCHIN; HINDUJA, 2006)
[2] Comportamento agressivo e repetitivo (SMITH, 2016)

ofensiva, caracterizada por opiniões separatistas, racistas, com sentidos ou estereótipos extremistas. Já (SILVA *et al.*, 2016) defendem que o que caracteriza *hate speech* são ataques motivados por qualquer característica relacionada a um grupo de pessoas. Em contrapartida, (WARNER; HIRSCHBERG, 2012) definem *hate speech* como uma forma particular de linguagem ofensiva que expressa uma ideologia de ódio.

Apesar de haver diversas definições sobre *hate speech*, rotular um conteúdo como tal pode ser uma tarefa complexa, visto que esses conceitos confudem-se com linguagem apenas ofensiva e com postagens de humor, que em muitos casos são protegidos por leis de liberdade de expressão, como o Marco Civil da Internet, estabelecido no Brasil desde 2014.

Devido à essa dificuldade, a moderação humana torna-se um processo minuncioso e custoso na relação temporal, ou seja, leva-se muito tempo para analisar cada postagem individualmente. No entanto, devido a grande quantidade de conteúdo que deve ser analisado, os moderadores tem pouco tempo para caracterizar o discurso, isso pode ser evidenciado pelo relato feito por um brasileiro que trabalhava no Facebook, para a BBC Brasil[3], ele informou que cada revisor tinha que avaliar cerca 3.500 fotos, videos e textos denunciados por dia.

Nesse contexto, destaca-se o uso de técnicas de aprendizagem de máquina, que permitem que algoritmos consigam realizar previsões ou melhorar comportamentos baseados em conhecimentos adquiridos (aprendizagem) sobre um conjunto de dados (MOLNAR, 2020).

Neste trabalho, exploramos o problema de detecção automática de *hate speech*, aplicando técnicas de Processamento de Linguagem Natural (no inglês: *Natural Language Processing* (NLP)) e aprendizagem de máquina. Além disso, propomos uma análise visual para seleção de um subconjunto S dentro de um *pool* de classificadores $C = \{c_1, c_2, ..., c_n\}$, objetivando aplicar técnicas de combinação com esse subconjunto. O *pool* é composto pelos algoritmos de aprendizagem treinados com diferentes métodos extratores, que proporcionam visões distintas de representação dos dados. Para validar os classificadores selecionados pela análise visual, fizemos uso do *Oracle*, que calcula a quantidade padrões corretamente classificados por pelo menos um dos classificadores que compôem o *pool*.

1.1 MOTIVAÇÃO

De acordo com (ZHANG; LUO, 2019a), o anonimato e a mobilidade proporcionada pelas redes sociais, facilitaram a criação e disseminação de discursos de ódio e que eventualmente, esses discursos podem levar a prática do crime de ódio.

Este tipo de opinião muitas vezes protegido pela liberdade de expressão, tornou-se um problema de escala mundial, ao qual vários países e organizações têm demonstrado preocupação (Watanabe; Bouazizi; Ohtsuki, 2018).

[3] https://www.bbc.com/portuguese/geral-41912670

Criar medidas efetivas contra a proliferação desse tipo de conteúdo em mídias sociais tem como primeiro passo qualificar o discurso como sendo de ódio ou não, grandes empresas como Facebook, Twitter e Youtube têm investido milhares de milhões de euros nesta tarefa. Mas, apesar de todo o investimento, de acordo com (ZHANG; LUO, 2019a), os esforços realizados por essas companhias ainda consistem em moderação manual, tornando o processo não sustentável, sem escalabilidade e de alto custo temporal.

(BISPO, 2018) destaca que esse tipo de tarefa ainda apresenta dois grandes desafios para a ciência:

1. Exige compreensão semântica sobre o que foi escrito, podendo haver presença de sarcasmo ou outros fatores que possam mascarar o real significado da expressão;

2. Grande variedade de expressões e símbolos utilizados pelos usuários que possam esconder o discurso de ódio. Essas variações podem ser até mesmo entre regiões do mesmo país, como gírias e vicíos de linguagem.

Diversas pesquisas com aprendizagem de máquina supervisionada vêm sendo conduzidas ao longo dos últimos anos com diferentes abordagens. Nesse contexto, destaca-se a abordagem clássica de classificação monolítica, ou seja, classificadores simples são treinados e fazem as predições individualmente. Nessa abordagem, podemos citar (MINAEE *et al.*, 2020) que avalia modelos *deep learning* e (ZAMPIERI *et al.*, 2019b) que avalia *Support Vector Machine - Linear SVC* (SVM), *Logistic Regression* (LR), além de *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN) e *Bidirectional long short-term memory* (BiLSTM).

Além da abordagem tradicional, nos últimos anos, pesquisas sobre *Multiple Classifier Systems* (MCS), em tarefas de classificação automática de conteúdo, vêm atraindo a atenção de pesquisadores (SMITS, 2002), (ZHU; MOH; MOH, 2016) e (CATAL; NANGIR, 2017).

MCS consistem em treinar um conjunto de classificadores e aplicar uma função de combinação. Dado um padrão de entrada x e um conjunto de classificadores $C = \{c_1, c_2, ..., c_n\}$, a esse padrão é atribuida uma classe aplicando a função de combinação nas predições de cada classificador que compõe o grupo.

De acordo com (SCIARRONE, 2012), essa abordagem visa melhorar a precisão no processo de classificação monolítica, quando a instância avaliada apresenta alguma informação discriminatória que pode ser explorada por outro classificador.

A escolha dos classificadores é determinante para o desempenho das combinações, e essa escolha está relacionada com a diversidade comportamental entre os mesmos. Busca-se selecionar classificadores que apresentem comportamentos distintos para os mesmos padrões, independentemente de serem homogêneos (mesmo algoritmo de aprendizagem) ou heterogêneos (algoritmos de apredizagem distintos). Caso o *pool* seja composto por classificadores que acertam e erram os mesmos padrões, não haverá possibilidade de melhorar

o desempenho, visto que as respostas dos classificadores diante de uma nova instância, tendem a ser as mesmas.

Para calcular essa diversidade, diversas métricas estão presentes na literatura, tais como: *disagreement*, *Q-statistic* e *Kappa-statistic* (CAVALCANTI *et al.*, 2016). Neste trabalho adotamos a métrica *double-fault measure* que calcula a proporção de padrões classificados incorretamente por dois classificadores.

1.2 OBJETIVO GERAL

Propor um modelo de análise visual, para selecionar um subconjunto de classificadores dentro de um *pool*, com a finalidade de aplicar técnicas de combinação com este subconjunto, em dados provenientes de mídias sociais para detecção de *hate speech*.

1.3 OBJETIVOS ESPECÍFICOS

- Coletar e processar os dados relacionados.
- Criar um *baseline* com os principais classificadores apresentados na literatura. Para a construção do *baseline*, realiza-se o treinamento dos algoritmos de aprendizagem com diferentes métodos extratores de características e utiliza-se a base de dados de teste para avaliar esses algoritmos.
- Realizar a exploração visual e a análise do *Oracle*, com o objetivo de selecionar um grupo de classificadores mais promissores dentro do *pool*. A exploração visual consiste em utilizar algum método de redução de dimensionalidade nas predições dos classificadores e projetar os resultados em um plano 2D.
- Aplicar técnicas de combinação com o subconjunto selecionado.
- Analisar os resultados.

1.4 ESTRUTURA DO DOCUMENTO

O restante deste trabalho está estruturado de acordo com a descrição a seguir:

No Capítulo 2, analisamos trabalhos anteriores, que utilizaram diferentes abordagens para detecção de *hate speech* com aprendizagem de máquina. No Capítulo 3, descrevemos os principais conceitos relacionados a classificação textual, abordagens de representação de características e métodos de combinação. No Capítulo 4, apresentamos a arquitetura e as características das principais etapas dos experimentos propostos. No Capítulo 5, descrevemos a metodologia dos experimentos, com destaque para apresentação das bases de dados, métricas de avaliação e configurações dos experimentos. No Capítulo 6, apresentamos os resultados encontrados para cada uma das bases de dados adotadas. No Capítulo

7, apresentamos as principais conclusões da pesquisa, além de propor ações que podem direcionar o andamento da pesquisa com trabalhos futuros.

2 TRABALHOS RELACIONADOS

Nos últimos anos, diversas pesquisas sobre discursos de ódio e aprendizagem de máquina foram conduzidas. Esta Seção apresenta uma série de trabalhos publicados recentemente, cujo objetivo é diferenciar discursos de ódio de outras formas de discurso.

(WASEEM; HOVY, 2016) apresentam um *dataset* de 16.914 instâncias coletadas do *Twitter* e que são rotulados em *racism*, *sexism* e *none*. Este trabalho analisa o impacto de características extra-linguísticas e *n-grams* a nível de caracter na detecção de ódio. Na análise o autor utiliza o *Logistic Regression* com *10-fold cross validation* e alcança um *F1-Score* de 73,89% sem utilizar o gênero como informação adicional e 73,93% utilizando o genêro.

(PARK; FUNG, 2017), combinam 2 bases de dados (WASEEM; HOVY, 2016) e (WASEEM, 2016), o conjunto resultante é rotulado em *racism*, *sexism* e *none*. A métrica adotada foi *F1-Score* com *average weithed*. Os experimentos foram realizados sob 2 perspecivas:

1. ***One-Step*** - Neste experimento, o autor busca classificar os registros em *none*, *racism* ou *sexism*. O melhor resultado alcançado foi 82,7% com *HybridCNN*, este modelo captura características a nível de palavras e também a nível de caracteres;

2. ***Two-Step*** - Neste experimento, o autor classifica os padrões em *racism* ou *sexism*, descartando os registros *none*. O melhor resultado alcançado foi 82,4% com *Logistic Regression*.

(DAVIDSON *et al.*, 2017) construiram uma base de dados com 24.802 registros rotulados em *hate*, *offensive* e *non-offensive*, utiliza TFIDF para extrair características com variação de *1-gram* a *3-gram* e *F1-score* como métrica de avaliação, para o treinamento o autor usa a base inteira e classifica os padrões utilizando o modelo *Logistic Regression* com a técnica *One-Versus-Rest*, alcançando um *F1-score* de 90%, porém erra 40% dos padrões rotulados como *hate*.

(Watanabe; Bouazizi; Ohtsuki, 2018) utilizam uma combinação de 3 *datasets*. O conjunto resultante é rotulado em *hate*, *offensive* e *clean* e dividido em treinamento, validação e testes, nos 3 segmentos as classes são balanceadas. As métricas utilizadas foram acurácia, *recall* e *F1-score*. Neste trabalho, foram realizados experimentos com classificação binária (combinando *hate* e *offensive*) e ternária. Os melhores resultados foram alcançados com o algoritmo *j48graft* utilizando o *toolkit weka*, atingindo uma acurácia de 87,4% e 78,4% para classificação binária e ternária, respectivamente.

(GIBERT *et al.*, 2018) apresentam o primeiro *dataset* de discurso de ódio disponível publicamente sobre postagens de fóruns na internet a nível de sentença, esta base é rotulada em 4 classes: *hate*, *noHate*, *relation* e *skip*. A classe *relation* são instâncias que tem relação com discurso de ódio, porém para identificar é necessário avaliar o contexto, ou

seja, somente o padrão não apresenta *hate speech* claramente. A classe *skip* não é avaliada pois não apresenta informações para ser classificada como *hate* ou *noHate*.

A base de dados apresenta 9.916 sentenças. No entanto, para o experimento de classificação, foram utilizados apenas 2.000 sentenças divididas igualmente entre *Hate* e *noHate*, deste montante 80% foi utilizado para treinamento e 20% para teste. A métrica utilizada foi a acurácia. Para classificação avaliou-se 3 algoritmos de aprendizagem: SVM, CNN e LSTM. O melhor resultado foi LSTM atingindo 78% excluindo sentenças *relation* e 73% incluindo *relation*.

(GAYDHANI *et al.*, 2018) realizam experimentos com a combinação de 3 bases de dados, o conjunto resultante é rotulado em *hatefull*, *offensive* e *clean*. Em seguinda a base é dividida em treinamento e teste na proporção de 80% e 20%, respectivamente. No processo de extração de características o autor utiliza a abordagem *Bag Of Words* aplicando TFIDF com variação de 1 a 3 *n-grams* e normalização entre L1 e L2, esse processo é utilizado para avaliar 3 algoritmos de aprendizagem, *Logistic Regression*, *Naive Bayes* e SVM, e em seguida customizar os hiperparâmetros, utilizando o *grid search* com *10-fold cross validation*.

O modelo com melhor desempenho foi *Logistic Regression* que alcançou uma acurácia de 95,6%.

(GAMBÄCK; SIKDAR, 2017) realizam experimentos com CNN utilizando a base de dados apresentada em (WASEEM, 2016), esse *dataset* possui 6.655 registros rotulados em *racism*, *sexism*, *both* e *non-hate-speech*. Neste trabalho, o autor utiliza *Word Embedding*, construido com *word2vec*, o método de treinamento testado foi *10-fold cross-validation* e o melhor resultado alcançou 78,3% de *F1-score*.

(ZAMPIERI *et al.*, 2019a) propôe um novo *dataset* que possui 14.100 registros extraidos do *Twitter*, como métrica de resultados o autor utiliza *macro-averaged F1-score*. A base é anotada em 3 níveis hierárquicos:

1. ***Offensive Language Detection*** - Diferencia conteúdo **ofensivo** de **discurso comum**. CNN alcançou 80,0% de *F1 measure*.

2. ***Categorization of Offensive Language*** - Diferencia **discurso de ódio** de **linguagem abusiva**. CNN alcançou 69,0% de *F1 measure*.

3. ***Offensive Language Target Identification*** - Classifica os alvos dos ataques, podendo ser o **grupo**, o **indivíduo** ou **outro**. CNN e BiLSTM alcançaram *F1* de 47,0%.

(MATHEW *et al.*, 2019a) cria uma *release* do primeiro *dataset* para *counterspeech* usando comentários do *youtube*, o conjunto contém 13.924 registros e possui 2 niveis de anotações: **detecção de *counterspeech*** e **tipos de *counterspeech***.

Antes do processo de classificação, foi realizado o preprocessamento, nessa fase o autor removeu URLs, números, *stopwords* e pontuações, além disso o texto foi transformado para

minúsculo e tokenizado. Para extração de características foram utilizadas as abordagens *Bag of Words Vectors* (BoWV), *Sentence Vectors* (SV) e TF-IDF. Os melhores resultados foram alcançados com XGBoost em conjunto com SV+TF-IDF+BoWV, obtendo F1 de 71,5% para detecção de *counterspeech* e F1 de 60,1% para tipos de *counterspeech* o método de treinamento utilizado foi *10-fold cross validation*.

3 REFERENCIAL TEÓRICO

Neste Capítulo, são abordados os principais conceitos utilizados no decorrer deste trabalho. Na Seção 3.1, além de conceitos e aplicações relacionadas a classificação de texto, são descritas as principais abordagens e métodos utilizados na extração de características. Na Seção 3.2, são apresentadas definições, etapas de construção, mecanismos para seleção dos classificadores e regras de combinação para um sistema de múltiplos classificadores.

3.1 CLASSIFICAÇÃO DE TEXTO

A classificação automática de texto tem atraído a atenção de diversos pesquisadores nos últimos anos (LODHI *et al.*, 2002), (ZHANG; ZHAO; LECUN, 2015), (LAI *et al.*, 2015).

Tarefas de classificação de texto ou categorização de texto, estão sempre presentes no campo da NLP(DAMASCHK; DÖNICKE; LUX, 2019), em atividades como: organização de documentos, filtragem de notícias, detecção de *spams* e mineração de opinião(YAO; MAO; LUO, 2019). Esse tipo de trabalho, consiste na utilização de técnicas e modelos de aprendizagem de máquina para classificar ou categorizar documentos.

Abordagens tradicionais para classificação de texto, utilizam aprendizagem de máquina supervisionada ou semi-supervisionada, em ambos os casos a extração das características é uma atividade intermédiaria e essencial(JIN *et al.*, 2016).

3.1.1 Métodos de Extração de Características

A extração das características é responsável por representar o padrão original de entrada de algoritmos de aprendizagem, em um vetor númerico de características, a forma como o vetor é gerado depende da abordagem utilizada na representação dos dados. As abordagens mais utilizadas para classificação de texto, são *Bag-of-Words* e *Word Embedding*.

3.1.2 *Bag-of-Words*

O modelo *Bag-of-Words* (BoW) é uma representação simplificada usada em NLP e na Recuperação de Informações (RI), este modelo constroi um vocabulário com todas as palavras presentes no *corpus*, desconsiderando a ordem e a classe gramátical e eliminando as repetições, assim como representado na Figura 1. Neste modelo há dois aspectos importantes: o vocabulário de palavras conhecidas e a frequência de ocorrência (HASSANIEN; ELHOSENY, 2019).

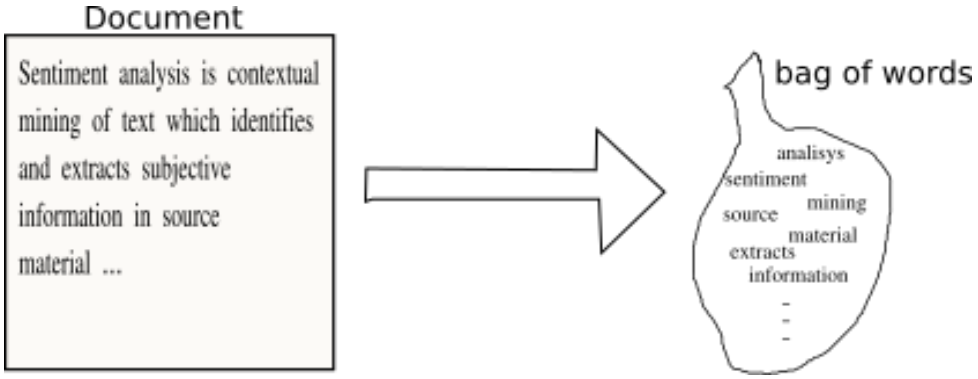

Figura 1 – Representação textual pelo modelo BoW.

Para estimar a frequência de ocorrência, pode-se utilizar o *Term Document Matrix* (TDM) ou *Term Frequency Inverse Document Frequency* (TFIDF).

3.1.2.1 Term Frequency - TF

Frequências de termos são comumente utilizadas para extração de características de documentos. Essa abordagem considera que termos frequentes são importantes para caracterizar o texto, e implementa a representação textual TDM, no qual as linhas representam os documentos e as colunas representam o termos que ocorrem no *corpus* (HASSANIEN; ELHOSENY, 2019). O TDM de um conjunto de documentos pode ser representado por *TF(i,j)*, sendo *TF* o número de ocorrências do termo i no documento j (BERRY, 2001).

Suponha que um *dataset* é formado pelas sentenças: [‘classificação de texto é legal.’, ‘análise de sentimentos’, ‘discurso de ódio’], ao utilizar o método TF nesse corpus, a matriz de ocorrência gerada seria semelhante a apresentada na Tabela 1.

Texto	**classificação**	**de**	**texto**	**é**	**legal**	**.**	**análise**	**sentimentos**	**discurso**	**ódio**
classificação de texto é legal.	1	1	1	1	1	1	0	0	0	0
análise de sentimentos	0	1	0	0	0	0	1	1	0	0
discurso de ódio	0	1	0	0	0	0	0	0	1	1

Tabela 1 – Matriz de ocorrência utilizando o *TF*.

Essa abordagem apresenta limitações, pois em geral os termos mais frequentes em um documento não representam grandes significados, como artigos e preposições. Diante disso, 2 técnicas são utilizadas para resolver este problema, a remoção de *stop words* e a utilização de *Inverse Document Frequency* (IDF).

3.1.2.2 Term Frequency - Inverse Document Frequency - TFIDF

Técnica que utiliza a abordagem BoW, porém ao contrário do modelo TF, foca na frequência inversa no documento, entendendo que palavras menos frequentes tendem a ser mais significantes para a classificação do texto.

A Equação 3.1 está representando o cálculo para vetorizar documentos utilizando o método TFIDF, nela a frequência (TF) do termo i no documento j é multiplicada pelo *log* do número total de documentos N dividido pela quantidade de documentos (DF) que contém o termo i.

$$tfidf_{(i,j)} = tf_{(i,j)} \times log(\frac{N}{df_i}) \tag{3.1}$$

O *Term Frequency* é o número de vezes que um termo aparece no documento dividido pela quantidade de termos no documento, nessa abordagem assume-se que todos os termos tem a mesma importância, enquanto que *Inverse Document Frequency* é representado por uma função logarítmica, calculada pelo número total de documentos dividido pelo número de documentos que possui o termo (HASSANIEN; ELHOSENY, 2019), nessa abordagem entende-se que termos mais raros tendem a ser mais relevantes.

3.1.3 *Word Embedding*

Enquanto *Bag-of-Words* considera apenas a frequência das palavras durante o processo de representação dos dados. O modelo *Word Embedding* captura informações sintáticas e semânticas para medir a similaridade das palavras (LIU *et al.*, 2015), codificando o significado semântico de uma palavra em um vetor númerico denso de baixa dimensão (CHEN *et al.*, 2015).

O objetivo dos algoritmos de representação de características *Word Embedding*, é gerar um vetor de palavras e seus contextos, sendo que os contextos são comumente associados a palavras localizadas ao redor de outras palavras (SHI; LIU, 2014).

Diante dessas considerações, vários algoritmos de extração de características foram desenvolvidos utilizando a abordagem *Word Embedding*. Essa seção, descreve o funcionamento dos mais populares que foram utilizados nos experimentos dessa pesquisa.

3.1.3.1 Word2Vec - W2V

Ao contrário do modelo TF e TFIDF que utilizam matriz esparsa, o Word2Vec (W2V) adota a abordagem *Word Embedding* fazendo a conversão para matriz densa e diminuindo a dimensionalidade. Este modelo foi proposto por uma equipe de pesquisadores do *Google* (MIKOLOV *et al.*, 2013), e transforma cada palavra do *corpus* em um vetor de representação semântica, utilizando o cálculo de similaridade de cossenos. Para isso, utiliza dois modelos de aprendizagem beaseados em redes neurais, o *Continuous Bag-of-Words* (CBoW) e o *Skip-gram*.

O modelo CBoW é uma rede neural com arquitetura de 3 camadas: uma de entrada, uma de projeção e uma de saída (MIKOLOV *et al.*, 2013). Neste modelo, um contexto representado por uma sentença é dado como entrada e a partir das projeções, busca-se predizer qual palavra refere-se ao contexto. O modelo *Skip-gram* apresenta funcionalidade

inversa ao CBoW, ou seja, a partir de uma palavra como entrada, as projeções buscam predizer palavras anteriores e posteriores a ela, para inferir o contexto (MIKOLOV *et al.*, 2013).

3.1.3.2 Global Vector for Word Representation - GloVe

Algoritmo de aprendizagem não supervisionada, que assim como o *Word2Vec* utiliza a abordagem *Word Embedding*, construindo vetores semânticos de palavras. No entanto, além do mapeamento semântico, *Global Vector for Word Representation* (GloVe) calcula a co-ocorrência de palavras de acordo com o contexto. Este modelo foi proposto por um grupo de cientistas da universidade de *Stanford*, e considera que as proporções de probabilidades de co-ocorrência palavra a palavra tem potencial de codificar significados (PENNINGTON; SOCHER; MANNING, 2014).

A similaridade entre duas palavras é encontrada calculando o produto escalar (função binária entre dois vetores) dos vetores dessas palavras. Para encontrar as palavras mais similares, obtem-se o produto escalar do vetor da palavra com todos os outros vetores de palavras, os maiores resultados indicam maior similaridade (Sharma *et al.*, 2017). Essas respostas, são muito precisas ao fazer analogias entre palavras, tais como: *uncle* está para *aunt* como *man* está para ?(*woman*).

3.1.3.3 FastText

FastText (FAST) é uma biblioteca *open source* desenvolvida por pesquisadores do Facebook para representação de palavras e classificação de textos, esse modelo apresenta resultados semelhantes a *deep learnings* em termos de acurácia, no entanto com velocidade superior tanto no treinamento quanto na avaliação (JOULIN *et al.*, 2017). Além disso, usa o modelo CBoW definido por (MIKOLOV *et al.*, 2013) e aplica a função *softmax* para calcular a distribuição de probabilidades sobre as classes pré-definidas. Inicialmente, o modelo transforma as palavras em um vetor de *n-grams*. Por exemplo, o vetor da palavra *apple* é separado em unidades menores e transformado no vetor ['ap', 'app', 'ple', 'le'], então o *Word Embedding Vector* da palavra *apple* é a soma de todos os *n-grams* (Mestry *et al.*, 2019).

A principal vantagem do *FastText* em relação a outros modelos *Word Embedding* é que por usar *n-grams*, o desempenho em relação a palavras raras ou não presentes no treinamento mantêm-se estável, pois o vetor de caracteres pode ser compartilhado por outras palavras (Mestry *et al.*, 2019).

3.2 MULTIPLE CLASSIFIER SYSTEMS - MCS

Abordagens tradicionais em tarefas de reconhecimento de padrões, costumam avaliar diversos algoritmos de aprendizagem, e aquele que obtiver o melhor desempenho com a

base de dados coletada, é utilizado em ambiente de produção como o melhor classificador. Essa abordagem, apesar de bastante consolidada, apresenta problemas quando lida com grandes quantidades de rótulos ou padrões ruidosos (HO; HULL; SRIHARI, 1994).

Diversas pesquisas foram realizadas e observou-se que diferentes tipos de classificadores e características podem apresentar desempenho complementar em tarefas de classificação (Cheng *et al.*, 2019) e (NWEKE *et al.*, 2019). Nesse contexto, surge o MCS, que consiste na concepção de um conjunto de classificadores bases diferentes, com a finalidade de combinar as saídas dos classificadores individualmente e prover uma única saída, o rótulo do padrão avaliado.

De acordo com (LI; ZONG; WANG, 2007), há três etapas na contrução de um MCS:

- Treinamento dos múltiplos classificadores - Consiste em treinar um conjunto de algoritmos de aprendizagem, homogêneos ou não, e constituir um *pool* com esses classificadores.

- Seleção de classificadores - Nessa fase, utiliza-se alguma medida de diversidade com o objetivo de selecionar os classificadores mais promissores com técnicas de combinação.

- Regras de combinação - Aplicar regras de combinação com as saídas de todos os classificadores selecionados na fase de seleção.

3.2.1 Diversidade e Oracle

De acordo com (Hansen; Salamon, 1990) e (Kuncheva *et al.*, 2000), não é possível melhorar o desempenho de classificações monolíticas com classificadores totalmente dependentes, ou seja, com classificadores que apresentam comportamentos idênticos. No entanto, se os classificadores errarem padrões diferentes, é possível que ao combinar suas saídas, a acurácia seja superada.

Diante dessa afirmação, percebe-se a necessidade de utilizar um método de avaliação de diversidade entre classificadores. Diversos métodos de cálculo de diversidade foram propostos e estão presentes na literatura (CAVALCANTI *et al.*, 2016). Esses cálculos servem para gerar uma matriz de dissimilaridade, que pode ser utilizada como direcionamento para seleção de um grupo promissor de classificadores para aplicar técnicas de combinação.

Para gerar a matriz de dissimilaridade, é necessário realizar as predições sobre determinada base de dados, e calcular a distância euclidiana entre os resultados de cada classificador do *pool*. A matriz é de ordem $m \times n$, na qual cada membro $d_{(i,j)}$ representa a dissimilaridade entre os classificadores C_i e C_j.

Mas além da diversidade, outras abordagens estão presentes e podem ser utilizadas. Um exemplo dessas abordagens é o *Oracle* de um grupo de classificadores qualquer garante que para cada padrão x dado como entrada será associado a classe correta, desde que

no *pool* tenha um classificador que prediz a classe correta do padrão (Kuncheva, 2000). Portanto, pode ser utilizado para avaliar a acurácia máxima que o *pool* combinado pode alcançar, e consequentemente selecionar os grupos mais promissores, além de possibilitar o descarte de outros.

3.2.2 Redução de Dimensionalidade

Modelos de classificação com aprendizagem de máquina tem como característica associar uma entrada x_q com uma saída $classe(x_q)$. Em diversos domínios e aplicações essas entradas apresentam uma alta-dimensionalidade, sendo usado frequentemente técnicas e algoritmos para redução dessa dimensionalidade, tanto para atividades de aprendizagem quanto para visualização e análises de dados.

Métodos de redução de dimensionalidade convertem um conjunto de dados de alta dimensão $X = \{x_1, x_2, ..., x_n\}$ em dados bidimensionais ou tridimensionais (MAATEN; HINTON, 2008). A redução de dimensionalidade tem como objetivo preservar as características estruturais mais significativas de conjuntos de dados de alta dimensão com um mapeamento de baixa dimensão (MAATEN; HINTON, 2008).

Nesse contexto, diversos algoritmos estão disponíveis na literatura e podem auxiliar nessa tarefa. Os mais populares são: *Principal Component Analysis* (PCA), que é uma técnica que mapeia dados de forma linear de acordo com a correspondência de matriz de covariância (GISBRECHT; SCHULZ; HAMMER, 2015), *t-Distributed Stochastic Neighbor Embedding* (t-SNE), técnica de mapeamento não linear que captura grande parte das características estruturais locais (MAATEN; HINTON, 2008) e *Uniform Manifold Approximation and Projection* (UMAP), que apresenta qualidade de visualização semelhante ao t-SNE, no entanto preserva as caracteristicas estruturais globais com maior qualidade e apresenta tempo de execução com melhor desempenho (MCINNES; HEALY; MELVILLE, 2018).

3.2.3 Regras de Combinação

De acordo com (Kuncheva, 2002), a diversidade é o principal fator determinante para um *pool* de classificadores superar a acurácia do melhor classificador monolítico. Além disso, afirma que apesar da escolha do método de combinação ter uma importância secundária, ao utilizar o método apropriado, pode-se melhorar o desempenho do MCS. Nesse contexto, pode-se citar os métodos de combinação *average*, *product*, *median*, *maximun*, *minimun* e *majority voting* que foram utilizados em (Kittler *et al.*, 1998) e (Kuncheva, 2002).

Para demonstrar o funcionamento das regras de combinações, considere um *pool* com N classificadores. Além disso, considere que $P_{j,i}(x)$ é a probabilidade a posteriori obtida pelo j-ésimo classificador para a i-ésima classe, para um padrão x dado como entrada.

3.2.3.1 Average

Obtém a média das probabilidades das classes entre os classificadores base e associa o padrão de entrada a classe que obteve a maior média (Equação 3.2). Essa função é equivalente a *sum rule* (Ponti Jr., 2011).

$$C_i(x) = argmax \left\{ \frac{1}{N} \sum_{j=1}^{N} P_{j,i}(x) \right\} \tag{3.2}$$

3.2.3.2 Product

Obtém a distribuição de probabilidade conjunta com as saídas dos classificadores base, e associa a classe que apresentou o maior valor (Equação 3.3).

$$C_i(x) = argmax \left\{ \prod_{j=1}^{N} P_{j,i}(x) \right\} \tag{3.3}$$

3.2.3.3 Maximum

Associa ao padrão de entrada a classe que apresentar o maior *score* obtido entre todos os classificadores do *pool* (Equação 3.4), ou seja, o classificador mais confiante no resultado, será o responsável por retornar a classe.

$$C_i(x) = argmax \left\{ \max_{j=1}^{N} P_{j,i}(x) \right\} \tag{3.4}$$

3.2.3.4 Minimum

Obtém a menor pontuação de cada classe entre todos os classificadores do *pool* e associa ao padrão a classe que obteve a maior pontuação (Equação 3.5).

$$C_i(x) = argmax \left\{ \min_{j=1}^{N} P_{j,i}(x) \right\} \tag{3.5}$$

3.2.3.5 Median

Considerando situações em que as probabilidades obtidas pelos classificadores de um MCS são drasticamente distintas (*outliers*), a decisão da regra *average* pode ser afetada. Esse problema pode ser resolvido ao utilizar *median*, regra que obtém a mediana da pontuação de cada classe e associa o padrão de entrada com a maior pontuação entre as medianas (Equação 3.6).

$$C_i(x) = argmax \left\{ \underset{j=1}{\overset{N}{med}} P_{j,i}(x) \right\} \tag{3.6}$$

3.2.3.6 Majority Vote

A técnica de maioria de votos também pode ser aplicada com MCS. Neste método, dado um padrão de entrada x, cada classificador "vota" uma classe para x, a saída desse método é a classe que obtiver o maior número de votos (RUTA; GABRYS, 2000).

4 METODOLOGIA PROPOSTA

No processo de combinação de classificadores, um fator importante que pode determinar se haverá ganho de desempenho é a escolha de quais classificadores serão combinados. Nesta seção, o método proposto para seleção e combinação dos classificadores é apresentado.

A Figura 2 mostra a arquitetura proposta que é formada por 6 módulos: extração de características, treinamento dos classificadores, cálculo da matriz de dissimilaridade, redução de dimensionalidade, análise de sensibilidade e combinação dos classificadores.

Cada classificador C_i do conjunto $\{C_1, C_2, ...C_n\}$, é treinado com abordagens diferentes de vetorização da base de dados de treinamento Γ. Além disso cada uma dessas abordagens é obtida com o método extrator E_i do conjunto $\{E_1, E_2, ...E_n\}$. Em seguida, realiza-se a classificação monolítica do conjunto de validação Λ. A partir das predições dessa base, calcula-se a matriz de dissimilaridade D. Essa matriz representa o comportamento de cada classificador em relação a todos os outros que compõem o *pool*. Em seguida, reduz-se a dimensionalidade da matriz D e projeta o resultado no plano 2D. Esse processo resulta em uma visão bidimensional $\tilde{D}$ representando o comportamento de cada um dos classificadores do *pool*. No módulo de análise de sensibilidade, faz-se a exploração visual do plano, com o objetivo de identificar um grupo de classificadores S com características complementares. Quanto maior for a proximidade entre os classificadores, mais similar são os resultados de classificação. Em contrapartida, classificadores mais distantes apresentam uma maior variação entre acertos e erros dos padrões. Essa informação indica que classificadores dissimilares possuem um *Oracle* maior em relação a classificadores com comportamento semelhante. No último módulo, diversas regras de combinação para o subconjunto S, resultante da exploração visual e composto por classificadores dissimilares, são avaliadas com a base de dados de teste T. A saída deste módulo é a $classe(x_q)$ para o padrão de entrada x_q.

A seguir são detalhados os seguintes módulos: matriz de dissimilaridade, espaço de projeção dos classificadores, análise de sensibilidade e combinação de classificadores.

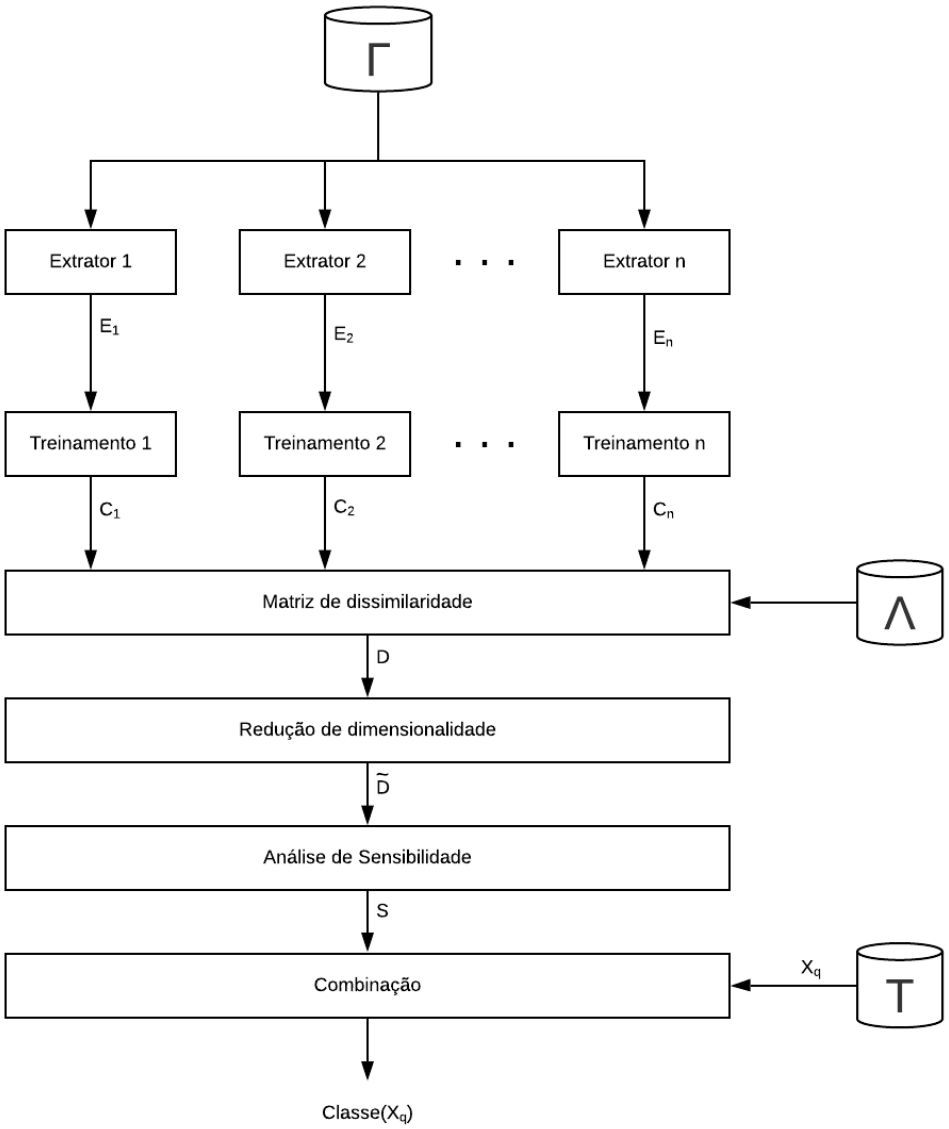

Figura 2 – Visão da arquitetura da segunda etapa dos experimentos, adotata para selecionar e combinar conjuntos de classificadores.

4.1 MATRIZ DE DISSIMILARIDADE

Como mencionado anteriormente a matriz de dissimilaridade é calculada de acordo com as predições dos classificadores em relação a determinada base de dados objetivando avaliar a diferença comportamental desses classificadores. Diversas medidas de diversidade estão presentes na literatura e são aplicadas em problemas semelhantes. Podemos citar: *Disagrement measure*, *Q-statistic* e *Double fault measure*.

O *Disagrement measure* calcula o nível de discordância entre dois classificadores base, ou seja, a proporção de padrões classificados corretamente por um e incorretamente pelo outro classificador do par. Já a métrica *Q-statistic*, calcula estatisticamente o nível de

independência entre um par de classificadores, a medida varia de -1 a 1, sendo igual a 0 quando os classificadores são totalmente independentes e maior que 0 quando a concordância de padrões corretamente classificados for maior que quando forem incorretamente classificados.

Neste trabalho, realizamos o cálculo de diversidade utilizando o coeficiente *double-fault measure* sobre a base de dados de validação. Essa métrica calcula a proporção de padrões incorretamente classificados por 2 classificadores. Esse coeficiente baseia-se no princípio que é mais importante saber quando ocorrem erros simultâneos do que acertos (KUNCHEVA, 2014). A medida é igual a 1 quando ambos os classificadores erram sempre as mesmas instâncias e 0 quando nunca erram o mesmo padrão.

Na Equação (4.1), é apresentado o cálculo do *double-fault*, sendo $N^{i,j}$ é o número de padrões corretamente (1) e erroneamente (0) classificados, para o C_i e C_j, respectivamente.

$$d_{(i,j)} = \frac{N^{00}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{4.1}$$

Esse coeficiente foi adotado devido a sua eficiência comprovada em trabalhos anteriores (LU *et al.*, 2014), (SHIPP; KUNCHEVA, 2002) e (CRUZ *et al.*, 2013). Além disso, os resultados encontrados atenderam o objetivo dessa pesquisa. Para encontrar o coeficiente foi utilizado o pacote *utils.diversity*[1] da biblioteca deslib.a

4.2 REDUÇÃO DE DIMENSIONALIDADE

Para redução de dimensionalidade utilizamos o algoritmo t-SNE (MAATEN; HINTON, 2012)(GISBRECHT; SCHULZ; HAMMER, 2015), este método é eficaz para visualizar conjuntos de dados complexos, descobrindo estruturas ocultas e convertendo as distâncias entre os pontos no espaço multidimensional em probabilidades que representam as similaridades (MAATEN, 2014). Além disso, adotamos a implementação disponível no pacote *manifold* do *scikit-learn*, com a o valor de perplexidade ajustado para 25.

Após a redução de dimensionalidade, projetamos os pontos encontrados em um plano 2D, utilizando a biblioteca *matplotlib*. Essa visualização possibilita a análise exploratória do gráfico.

4.3 ANÁLISE DE SENSIBILIDADE

Para a realização da análise de sensibilidade, o gráfico de representação comportamental de cada um dos classificadores do *pool*, obtido com a redução de dimensionalidade, é utilizado. Esse método, busca selecionar um subconjunto de classificadores visualmente dispersos, ou seja, com comportamentos complementares.

Considerando que essa visão é obtida utilizando as predições da base de dados de validação, espera-se identificar quais classificadores combinados apresentam *Oracle* maior

[1] https://deslib.readthedocs.io/en/latest/modules/util/diversity.html

nas predições da base de dados de teste, e portanto selecionar os subconjuntos mais promissores para aplicar técnicas de combinação de classificadores.

4.4 COMBINAÇÃO DE CLASSIFICADORES

Este módulo consiste em avaliar as regras de combinação de classificadores apresentadas na Seção 3.2, utilizando o subconjunto de classificadores S, resultante da análise de sensibilidade.

A base de dados de teste e as implementações disponíveis no pacote *utils.aggregation*[2] da biblioteca *deslib* foram utilizados para avaliar as combinações. Os resultados encontrados são então comparados com os valores obtidos na classificação monolítica, para determinar se há ou não ganho de desempenho.

[2] https://deslib.readthedocs.io/en/latest/modules/util/aggregation.html

5 METODOLOGIA DOS EXPERIMENTOS

Nesta seção abordamos como foram agrupados os classificadores, quais bases de dados foram utilizadas e como foram segmentadas e quais métricas de avaliação foram utilizadas.

5.1 BASES DE DADOS

Neste trabalho são avaliadas 3 bases de dados, 2 foram selecionadas de trabalhos anteriores e uma foi construida através da combinação das outras duas.

A primeira base de dados (TD), desenvolvida por (DAVIDSON *et al.*, 2017), está disponível em data.word[1] e possui 24.783 registros classificados em *hate*, *offensive* e *non-offensive*. O *dataset* foi construído buscando *tweets* que continham termos compilados pelo <*hatebase.org*>. Para capturar os registros fez-se uso da *Application Programming Interface* (API) do *Twitter*.

A segunda base de dados (ZW), desenvolvida por (WASEEM; HOVY, 2016), está disponível no github[2] e possui 16.907 registros, classificados em *racism*, *sexism* e *none*. Na construção dessa base de dados, o autor inicialmente faz uma pesquisa manual sobre insultos e termos comuns usados para relacionar minorias religiosas, sexuais, de gênero e étnicas. Com base nos termos encontrados na pesquisa, utilizou-se a API de busca do *Twitter* para coletar o *corpus*.

O *corpus* foi rotulado pelos autores e revisado por um avaliador externo com o objetivo de mitigar a possibilidade das anotações serem enviesadas por alguma das partes.

A terceira base de dados (TD+ZW) foi construída combinando as duas bases anteriores. Cada uma das bases utilizadas na combinação foi construída com uma finalidade diferente e com características de anotações distintas, portanto para combinar os registros faz-se necessário analisar quais registros podem ser combinados e unificar os rótulos.

Neste trabalho busca-se classificar os registros em **discurso de ódio**, **discurso apenas ofensivo** ou **discurso comum**, cada um desses rótulos tem um correspondente na base TD, sendo necessário modificar as anotações apenas da base ZW.

Para unificar os registros, a estratégia utilizada foi modificar as classes referentes a *racism* e *sexism* presentes na base de dados ZW, transformando-as em **discurso de ódio**. Por não haver uma descrição clara sobre qual a classificação dos registros *none*, resolveu-se descartá-los, similar ao trabalho (Watanabe; Bouazizi; Ohtsuki, 2018).

O passo seguinte foi combinar os registros, resultando em uma única base de dados com 30.131 registros. Na Tabela 2 são apresentados a distribuição das instâncias por classe para cada base apresentada.

[1] <https://data.world/thomasrdavidson/hate-speech-and-offensive-language>
[2] <https://github.com/ZeerakW/hatespeech>

Base de dados	# Instâncias	# Padrões por classe
[TD] (DAVIDSON *et al.*, 2017)	24.783	*hate*: 1.430 *offensive*: 19.190 *non-offensive*: 4.163
[ZW] (WASEEM; HOVY, 2016)	16.907	*racism*: 1.970 *sexism*: 3.378 *none*: 11.559
[TD+ZW]	30.131	*hate*: 6.678 *offensive*: 19.190 *non-offensive*: 4.163

Tabela 2 – Distribuição das Instâncias nas Bases de dados.

Cada uma das bases de dados foi dividida em três partes: Treinamento (67%), Validação (16,5%) e Teste (16,5%). Na Tabela 3, apresenta-se a quantidade de instâncias de cada segmento da base, enquanto que na Tabela 4 são apresentados as distribuições por classe.

Base de dados	# Treinamento	# Validação	# Teste
TD	16.604	4.090	4.089
ZW	11.327	2.790	2.790
TD+ZW	20.187	4.972	4.972

Tabela 3 – Segmentação das bases em treinamento, validação e teste.

Base de dados	# Treinamento	# Validação	# Teste
TD	*hate*: 957 *offensive*: 12.855 *non-offensive*: 2.792	*hate*: 243 *offensive*: 3.179 *non-offensive*: 668	*hate*: 230 *offensive*: 3.156 *non-offensive*: 703
ZW	*sexism*: 2.319 *racism*: 1.291 *none*: 7.717	*sexism*: 509 *racism*: 333 *none*: 1.948	*sexism*: 550 *racism*: 346 *none*: 1.894
TD+ZW	*hate*: 4.579 *offensive*: 12.811 *non-offensive*: 2.797	*hate*: 1.098 *offensive*: 3.202 *non-offensive*: 672	*hate*: 1.101 *offensive*: 3.177 *non-offensive*: 694

Tabela 4 – Distribuição dos padrões para cada segmento das Bases de dados.

5.2 MÉTRICAS

Para avaliar os modelos, duas métricas foram usadas: A **acurácia** é obtida dividindo o número de acertos pelo número total de instâncias, (Equação 5.1a), e a ***f1-score*** é a média harmônica de *precision* e *recall* (Lim *et al.*, 2007), (Equação5.1b). *Precision* é obtido pela divisão do número de padrões positivos corretamente classificados pelo número total de padrões classificados como positivos e *recall* é dado a partir da relação de verdadeiros positivos dividido pelos verdadeiros positivos somado ao falsos negativos.

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \qquad (5.1a) \qquad F1 = \frac{2 \times precision \times recall}{precision + recall} \qquad (5.1b)$$

Ao utilizar a métrica *f1-score* para problemas com múltiplas classes, pode-se configurar formas distintas de calcular a média dos dados. Neste trabalho adotamos ***macro f1-score***, este método calcula *F1-measure* para cada rótulo individualmente e em seguida computa a média dos resultados (Equação 5.2).

$$F1_{macro} = \frac{F1_0 + F1_1 + ... + F1_n}{n} \qquad (5.2)$$

5.3 CONFIGURAÇÃO DOS EXPERIMENTOS

Existe uma grande quantidade de máquinas de aprendizagem disponíveis e aplicadas em problemas diversos. Encontrar a melhor máquina de aprendizagem para resolver um problema específico pode ser uma tarefa complexa e envolve além da escolha do algoritmo, o mecanismo de extração de características e a configuração do classificador.

Buscando avaliar uma grande quantidade de combinações e avaliar o comportamento de diferentes classificadores, foram selecionados 8 algoritmos de aprendizagem e 5 métodos extratores de características, cada algoritmo foi treinado com cada um dos extratores, totalizando 40 classificadores.

O experimento foi dividido em 2 etapas: criação do *baseline* e aplicação do método.

5.3.1 Criação do baseline

A primeira fase consiste em treinar um conjunto *(pool)* de classificadores que serão utilizados no experimento de combinação. Para isso, alguns dos algoritmos mais utilizados na literatura, foram avaliados: SVM (GIBERT *et al.*, 2018), (Watanabe; Bouazizi; Ohtsuki, 2018), LR (GAYDHANI *et al.*, 2018), (DAVIDSON *et al.*, 2017), *RandomForest* (RF)(Watanabe; Bouazizi; Ohtsuki, 2018), *Naive Bayes* (NB)(GAYDHANI *et al.*, 2018), *ExtraTrees* (EXTRA) (THIN; LE; NGUYEN, 1991), K Nearest Neighbor (KNN) (DUCHARME *et al.*, 2017) e as redes neurais *Multi-layer Perceptron* (MLP) (MATHEW *et al.*, 2019b), (GRöNDAHL *et al.*, 2018) e CNN (GIBERT *et al.*, 2018), (ZHANG; LUO, 2019b).

Esse algoritmos foram treinados com diferentes visões de representação dos dados. Essas visões são proporcionadas pela aplicação de métodos extratores de características e devem ser aplicados para todos os segmentos da base de dados.

5.3.1.1 Pré-processamento e extração de características

Com o objetivo de melhorar a extração das características foram utilizadas algumas técnicas de NLP, essas atividades de processamento são comuns na literatura e estão presentes em (GAYDHANI *et al.*, 2018):

- Remoção de URLs;
- Remoção de RT (*retweets*);
- Remoção de tags iniciadas em "@";
- Remoção de números;
- Redução de sequencias de espaços;
- Remoção de *stop english words*;
- Normalização *unicode* para o padrão NFKD;
- Transformação do texto em minúsculas.

Para realizar a extração de características, 5 métodos foram utilizados. Estes métodos podem ser divididos em 2 grupos de acordo com a abordagem utilizada: *Bag-of-Words* e *Word Embedding*.

O vetor de características é uma matriz multidimensional formada pelos padrões de entrada juntamente com as características extraídas de cada padrão, sendo assim o tamanho do vetor é $L \times D$, tal qual L é a quantidade de instâncias que serão vetorizadas e D é a quantidade de dimensões utilizada pelo método extrator.

Ao utilizar a abordagem *bag-of-words*, a priori, o tamanho do vetor de características para cada instância é igual ao tamanho do vocabulário, enquanto que no modelo *Word Embedding* para os métodos *Word2Vec* e *FastText* a quantidade de características extraídas por instância foi igual a 300. O método Glove obteve um vetor de características com 25 dimensões, essa variação deve-se ao fato de não ter sido fixada as dimensões para nenhum dos métodos. Para extração do tipo BoW, foram utilizadas as implementações *TfIdfVectorizer* e *CountVectorizer* do *scikit-learn*[3], e para as implementações *Word Embedding* foi utilizada a implementação *Zeugma*[4], essa implementação forma um vetor médio de características para cada instância.

[3] https://scikit-learn.org/stable/
[4] https://zeugma.readthedocs.io/en/latest/

5.3.1.2 Hiperparâmetros e treinamento dos classificadores

Como afirmado anteriormente, o desempenho do classificador não está relacionado apenas com a escolha do algoritmo, mas também deve-se considerar os ajustes que cada modelo pode suportar e que pode melhorar sua performance. De acordo com (SNOEK; LAROCHELLE; ADAMS, 2012), os resultados alcançados por algoritmos de aprendizagem de máquina, frequentemente relacionam-se com ajustes de hiperparâmetros dos modelos.

Para facilitar a avaliação dessas combinações, fez-se uso de uma ferramenta facilitadora, o *ParameterGrid* que está disponível no *Scikit-learn*, para todos os modelos de aprendizagem adotados e cuja a implementação também esta disponível na biblioteca. A exceção é a CNN que utilizou-se a biblioteca *Keras* para implementação e para avaliar os hiperparâmetros realizou-se uma análise manual.

Para o treinamento dos classificadores, cada algoritmo de aprendizagem foi treinado com diferentes combinações de hiperparâmetros aplicada a cada um dos 5 métodos extratores, sendo que a combinação que obtém o melhor desempenho com o conjunto de validação é utilizada para o *baseline*. Para CNN além de avaliar a função de ativação e o otimizador, foi necessário construir a arquitetura, sendo que foi adotada a mesma arquitetura para todas as visões de representação dos dados. A arquitetura implementada é uma rede neural sequencial com às seguintes camadas:

1. *Embedding* - Camada de entrada da rede, sendo que os pesos são configurados com as características extraídas pelos métodos extratores.

2. *Dropout* - Camada responsável por eliminar aleatóriamente uma proporção dos neurônios com o objetivo de evitar *overfitting*, o valor foi ajustado em 20%.

3. Convolução 1D - Camada responsável pela extração de características, foi ajustada com 64 filtros, *kernel* de tamanho 3, tamanho de passo 1 e ativação *relu*.

4. Subamostragem 1D - Camada utilizada para regularizar a estrutura do mapa de características, reduzindo a dimensionalidade das saídas da camada de convolução com ativação máxima global, sendo que apenas o valor máximo presente em cada filtro da camada anterior é mantido.

5. Completamente conectada - Composta por 256 neurônios e ativação linear.

6. *Dropout* - Taxa de eliminação de neurônios também foi ajustada em 20% seguida por uma função de ativação *relu*.

7. Completamente conectada - Composta por 3 neurônios e ativação que pode variar de acordo com o método extrator de características e o *dataset*.

Além da arquitetura da CNN, foi definida uma arquitetura padrão para a MLP, essa arquitetura foi utilizada em todos os experimentos e consite em 2 camadas: uma de entrada

com 100 neurônios e uma de saída, sendo utilizada a implementação que está disponível na versão 0.23.1 do *scikit-learn*.

5.3.2 Aplicação do método

Para a realização dos experimentos, os classificadores foram agrupados de 4 formas diferentes:

1. **Grupo A** - Na classificação monolítica, foram avaliados 8 algoritmos de aprendizagem de máquina, além disso os algoritmos foram treinados com cada um dos cinco métodos extratores, totalizando 40 classificadores. O grupo A é composto por 8 subgrupos de classificadores, sendo que cada subgrupo é formado pelo mesmo algoritmo de aprendizagem aplicado com cada um dos extratores.

2. **Grupo B** - Grupo criado a partir da análise de sensibilidade em relação ao comportamento dos classificadores projetados no plano 2D. Este grupo é composto por classificadores visualmente dissimilares no plano, ou seja, distantes entre si. A quantidade de classificadores que compõem esse grupo é igual a 5, pois no módulo de combinação, avaliam-se os resultados alcançados combinando cada subgrupo do grupo A, e este grupo é composto exatamente por 5 classificadores, então para efeitos de comparação fez-se uso da mesma quantidade para compor o Grupo B.

3. **Grupo C** - Similar ao Grupo B, este grupo também é criado a partir da exploração visual do gráfico de dissimilaridade. No entanto, esse grupo tem a finalidade de confrontar os resultados obtidos pelo Grupo B selecionando classificadores visualmente próximos no plano. Como o objetivo deste grupo é contrapor os resultados obtidos no grupo B, a composição dele é formada pela mesma quantidade de classificadores, 5.

4. **Grupo D** - Grupo composto por todos os classificadores treinados, 40 classificadores. Este grupo tem a finalidade de avaliar o desempenho combinando todos os classificadores.

6 RESULTADOS

Nesta seção, serão apresentados todos os experimentos realizados para cada uma das três bases de dados: TD, ZW e TD+ZW, respectivamente.

Inicialmente, serão apresentadas as informações dos classificadores monolíticos que formam o *baseline* referente à seleção de hiperparâmetros e resultados obtidos com os mesmos. Em seguida, a exploração visual e a demonstração do *Oracle* para seleção dos grupos B e C de classificadores e, por fim, os resultados obtidos com as técnicas de combinação.

6.1 BASE DE DADOS TD

A Tabela 5 apresenta quais parâmetros foram avaliados, para cada classificador, e quais alcançaram os melhores resultados com o conjunto de validação.

Parâmetros	**TF**	**TFIDF**	**W2V**	**Glove**	**FastText**
		SVM Linear SVC			
Kernel:[Linear, Sigmoid, rbf] Gamma: [0.1, 1, 0.5]	Kernel: linear Gamma: 0.1	Kernel: linear Gamma: 0.1	Kernel: rbf Gamma: 1	Kernel: rbf Gamma: 0.5	Kernel: rbf Gamma: 1
		LogisticRegression			
penalty: [l1, l2]	penalty: l1	penalty: l1	penalty: l1	penalty: l1	penalty: l1
		RandomForest			
n_estimators: [10, 20, 50]	n_estimators: 20	n_estimators: 50	n_estimators: 50	n_estimators: 50	n_estimators: 50
		Naive Bayes			
alpha: [0.1, 0.5, 1] fit_prior: [False, True]	alpha: 1 fit_prior: False	alpha: 0.5 fit_prior: False	alpha: 0.5 fit_prior: True	alpha: 0.1 fit_prior: True	alpha: 1 fit_prior: True
		MLP			
activation: [relu, logistic] solver: [adam, lbfgs]	activation: relu solver: lbfgs	activation: logistic solver: adam	activation: relu solver: adam	activation: relu solver: adam	activation: relu solver: adam
		ExtraTrees			
n_estimators: [10, 20, 50]	n_estimators: 50	n_estimators: 50	n_estimators: 50	n_estimators: 50	n_estimators: 50
		CNN			
activation: [sigmoid, softmax] optimizer: [rmsprop, adam]	activation: sigmoid optimizer: rmsprop	activation: softmax optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop
		KNN			
algorithm: [auto, ball_tree, kd_tree] n_neighbors: [3, 5]	algorithm: auto n_neighbors: 3	algorithm: auto n_neighbors: 5	algorithm: auto n_neighbors: 5	algorithm: ball_tree n_neighbors: 5	algorithm: auto n_neighbors: 5

Tabela 5 – Avaliação de hiperparâmetros para a base TD.

Ao analisar a Tabela 5, é importante observar que entre todos os algoritmos avaliados apenas em *LogisticRegression* as opções avaliadas, ou seja, o hiperparâmetro *penalty* variando entre **L1** e **L2** e *ExtraTrees* com o hiperparâmetro *n_estimators* variando entre 10, 20 e 50 ficaram estáveis independente do método extrator de características.

Na Tabela 6, apresentam-se as informações de acurácia para a classificação monolítica da base de dados TD. De acordo com o exposto, percebe-se que os modelos que atingiram o melhor desempenho foram *LogisticRegression* com TF e *SVM Linear SVC* com TfIdf, ambos alcançaram uma acurácia de 90,6%.

Algoritmo	TF	TfIdf	W2V	GLOVE	FastText
SVM Linear SVC	0,897	**0,906**	0,847	0,829	0,851
LogisticRegression	**0,906**	0,902	0,837	0,821	0,845
RandomForest	0,893	**0,900**	0,820	0,821	0,826
Naive Bayes	**0,880**	0,854	0,749	0,799	0,772
MLP	**0,895**	0,886	0,841	0,826	0,854
ExtraTrees	0,881	**0,891**	0,817	0,822	0,811
KNN	**0,829**	0,794	0,822	0,801	0,806
CNN	**0,901**	0,894	0,876	0,870	0,837

Tabela 6 – Acurácia do *baseline* de acordo com os hiperparâmetros encontrados e apresentados na Tabela 5.

Na Tabela 7, apresentam-se as informações de *macro F1* para a classificação monolítica da base de dados TD. De acordo com o exposto, percebe-se que similarmente a acurácia, o modelo que alcançou o melhor resultado foi *LogisticRegression* com TF, atingindo 70,6%.

Algoritmo	TF	TfIdf	W2V	GLOVE	FastText
SVM Linear SVC	**0,699**	0,683	0,540	0,507	0,537
LogisticRegression	**0,706**	0,679	0,539	0,484	0,548
RandomForest	**0,699**	0,671	0,483	0,500	0,494
Naive Bayes	**0,678**	0,643	0,530	0,481	0,573
MLP	**0,692**	0,687	0,565	0,530	0,586
ExtraTrees	**0,687**	0,661	0,463	0,493	0,450
KNN	**0,618**	0,542	0,506	0,493	0,477
CNN	0,680	0,590	0,679	0,559	**0,697**

Tabela 7 – Macro F1 do *baseline* de acordo com os hiperparâmetros encontrados e apresentados na Tabela 5.

Ainda de acordo com a Tabela 7, é importante destacar que SVM não alcançou o mesmo resultado que *LogisticRegression*, diferentemente do que aconteceu com a acurácia, apresentada na Tabela 6. Esse resultado indica que os dois classificadores acertam a mesma quantidade de padrões, no entanto SVM erra mais instâncias em uma ou duas das três classes.

Outro aspecto importante pode ser observado ao analisar os resultados alcançados com *RandomForest*, haja visto que ao analisar a acurácia, percebe-se que o melhor resultado foi encontrado, aplicando o método de extração de características, TfIdf. Este classificador alcançou apenas o 4º melhor resultado com uma acurácia de 90,0%. No entanto, ao analisar o *macro F1* o modelo aplicado ao método extrator TF alcançou juntamente com SVM o segundo melhor resultado, atingindo 60,9%. Essa informação indica que estes

2 classificadores acertam quantidades distintas de padrões, porém apresentam a mesma proporção de acertos por classe.

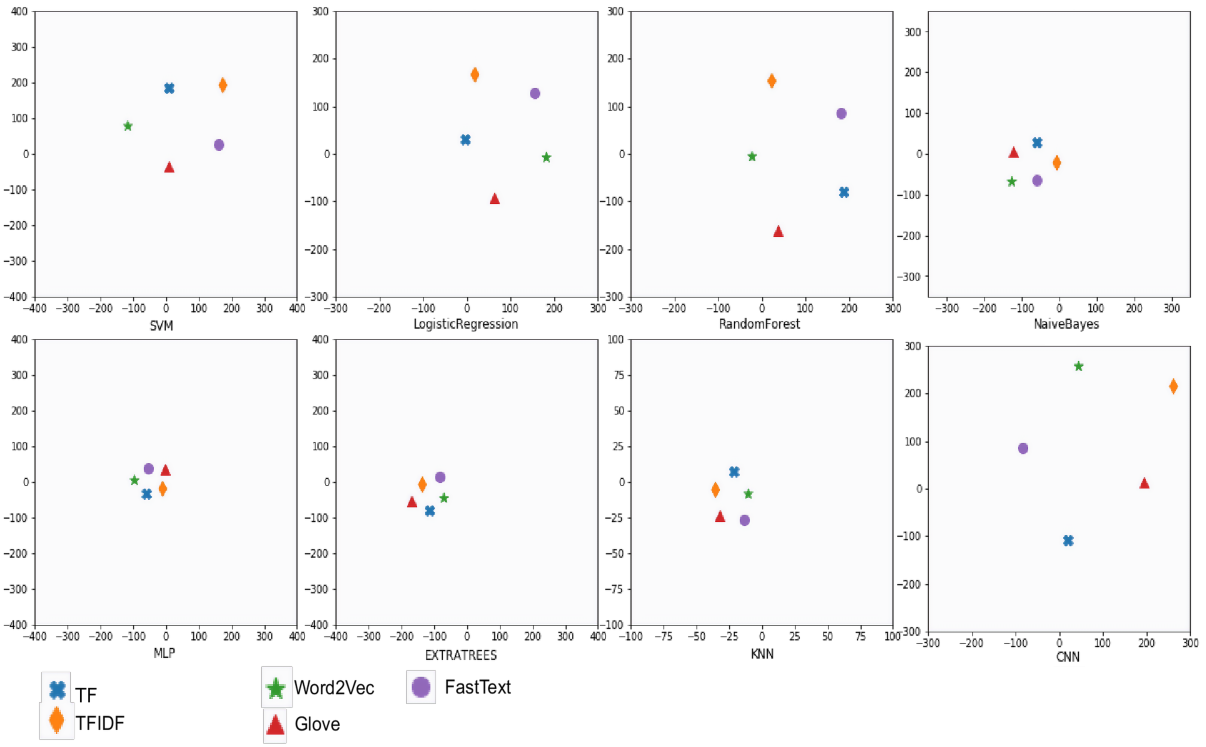

Figura 3 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD de acordo com as extrações de características .

A Figura 3 apresenta a distribuição dos algoritmos de aprendizagem testados no *baseline*, utilizando o conjunto de validação para o Grupo A, e na Tabela 8 é exibido o *Oracle* para os mesmos classificadores. Através da análise desses dados, é possível determinar que CNN, grupo de classificadores visualmente mais disperso, possui um *Oracle* maior, indicando que são classificadores com comportamento complementares.

Algoritmo	**Acertos**	**Acurácia**	**Macro F1**
SVM (A)	3.882	0,949	0,781
LogisticRegression (A)	3.882	0,949	0,776
RandomForest (A)	3.858	0,943	0,759
Naive Bayes (A)	3.871	0,946	0,810
MLP (A)	3.907	0,955	0,817
ExtraTrees (A)	3.845	0,940	0,751
KNN (A)	3.834	0,937	0,772
CNN (A)	3.974	0,971	0,899

Tabela 8 – *Oracle* para cada subgrupo do Grupo A para a base TD. (A) Significa que os experimentos são do Grupo A.

Este comportamento pode ser ratificado com a Figura 4, apresentando a quantidade de classificadores que acertam o mesmo padrão para um subgrupo visualmente disperso (CNN) e para um subgrupo visualmente próximo (*ExtraTrees*). Analisando esse histograma, percebe-se que o subgrupo mais disperso, possui uma variância maior de classificadores que acertam os padrões, também percebe-se que a quantidade de instâncias que não são classificadas corretamente nenhuma vez, é maior em um subgrupo próximo.

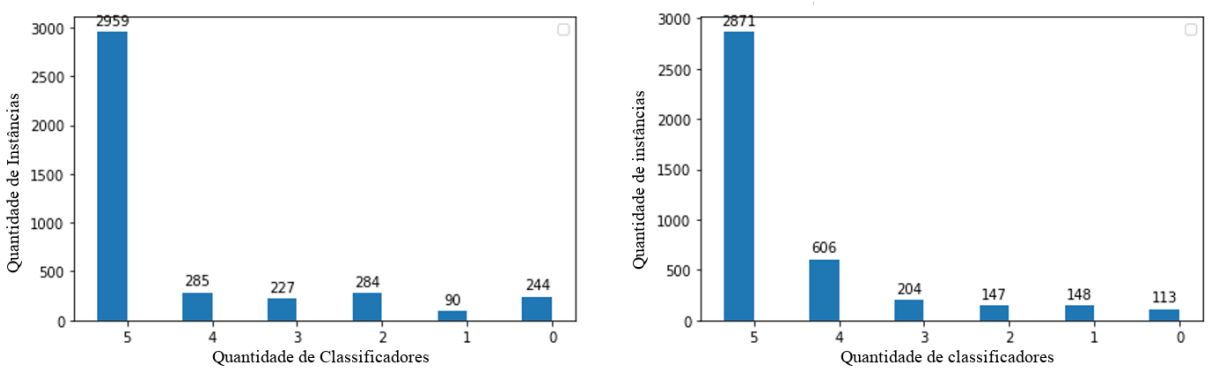

Figura 4 – Quantidade de classificadores (eixo x) que acertam os padrões (eixo y) para o subgrupo *ExtraTrees* (esquerda) e para o subgrupo de CNN (direita).

Após a análise do Grupo A, realizou-se a exploração visual de todos os classificadores. A Figura 5 apresenta o plano 2D que representa o comportamento de todos os classificadores.

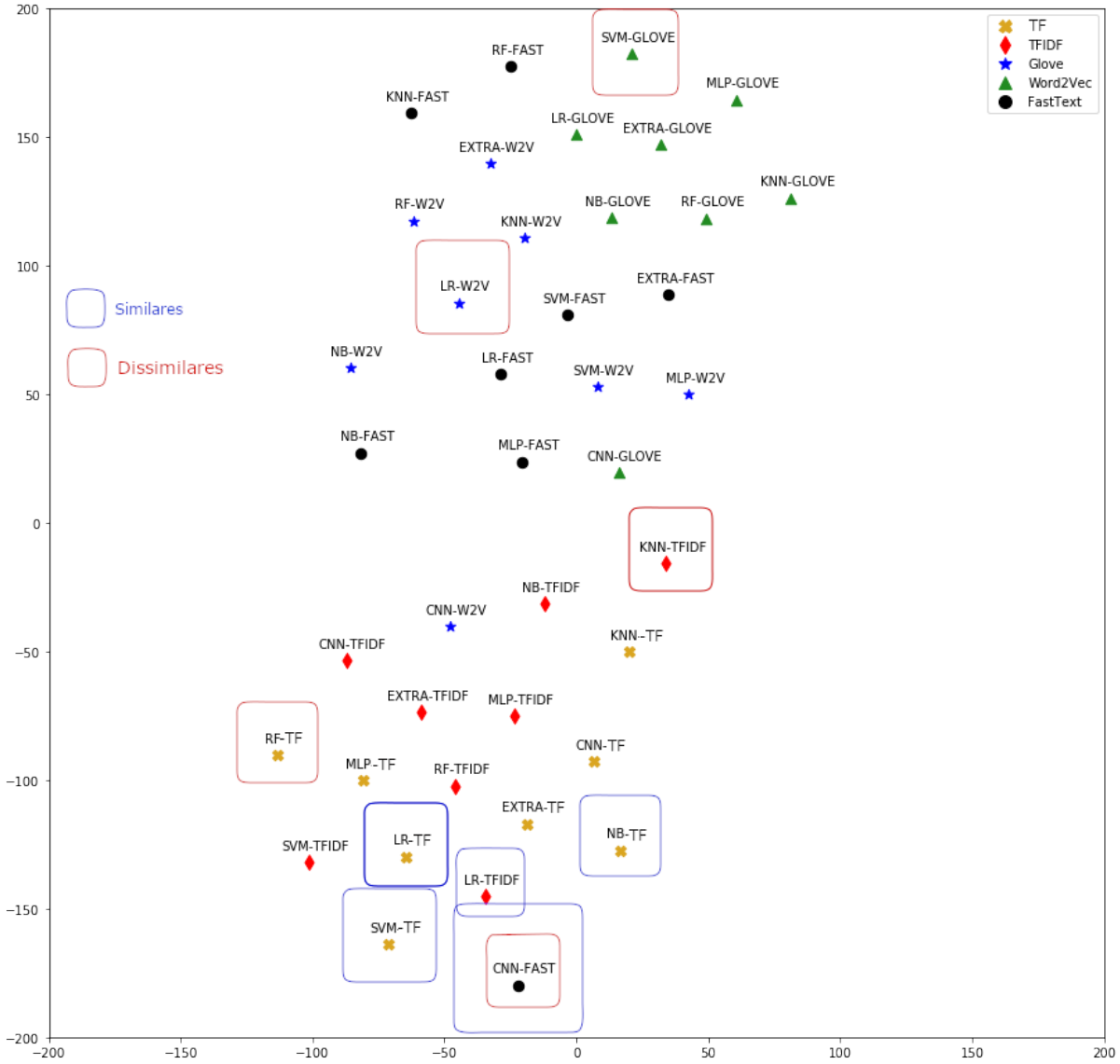

Figura 5 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD independente do método extrator de características.

Após análise da Figura 5 identificamos 2 grupos: o grupo **B** que é composto por classificadores dissimilares e o grupo **C** contendo os classificadores similares, os grupos formados foram:

B (CNN-FAST)+(RF-TF)+(KNN-TFIDF)+(LR-W2V)+(SVM-GLOVE).

C (CNN-FAST)+(SVM-TF)+(LR-TFIDF)+(LR-TF)+(NB-TF).

Na Tabela 9, percebe-se que o *Oracle* dos dissimilares é inferior somente ao de todos os classificadores juntos, representado pelo grupo D.

Algoritmo	Acertos	Acurácia	Macro F1
GRUPO B	3.978	0,972	0,900
GRUPO C	3.938	0,963	0,879
GRUPO D	4.033	0,986	0,951

Tabela 9 – *Oracle* para os Grupos B, C e D para a base TD.

A Tabela 10 mostra a acurácia encontrada utilizando as técnicas de combinações para todos os grupos utilizados na base de dados TD. De acordo com esses dados, a combinação do Grupo C superou o melhor resultado individual do *baseline*, atingindo uma acurácia de 90,8% utilizando a função de fusão *Median*. Além disso, os subgrupos KNN e CNN conseguiram superar seus resultados individuais utilizando a função *Sum/Avg*, obtendo 83,6% e 90,8%, respectivamente.

O Grupo B, apesar de ser composto por classificadores com comportamento semelhante, obteve resultados que superaram as combinações dos subgrupos de A: *Naive Bayes*, MLP, *ExtraTrees* e KNN.

Algoritmo	Voting	Sum/Avg	Max	Min	Median	Product
SVM Linear SVC (A)	0,880	0,892	**0,894**	0,893	0,879	0,894
LogisticRegression (A)	0,875	**0,887**	0,883	0,881	0,875	0,885
RandomForest (A)	0,864	0,881	**0,887**	0,881	0,862	0,882
Naive Bayes (A)	0,849	0,841	0,805	0,803	**0,850**	0,829
MLP (A)	0,882	0,890	0,892	0,890	0,882	**0,893**
ExtraTrees (A)	0,850	0,876	**0,881**	0,880	0,850	0,877
KNN (A)	0,833	**0,836**	0,831	0,819	0,832	0,825
CNN (A)	0,904	**0,908**	0,897	0,895	0,908	0,907
GRUPO B	0,879	0,880	0,877	**0,900**	0,868	0,894
GRUPO C	0,906	0,905	0,901	0,898	**0,908**	0,907
GRUPO D	0,884	**0,891**	0,855	0,840	0,886	0,841

Tabela 10 – Resultados em termos de acurácia encontrados com as técnicas de combinações para a base TD.

Na Tabela 11, apresentam-se os resultados obtido em termos de *macro F1*. Analisando esses dados, percebe-se que houve um decréscimo nos valores obtidos ao combinar os classificadores em relação ao *baseline*, essa perda de desempenho indica uma maior "confusão" na separação das classes, quando se combina os classificadores. Porém, o Grupo C superou o melhor resultado do *baseline* alcançado o resultado de 73,1% utilizando a função *Min*.

Algoritmo	**Voting**	**Sum/Avg**	**Max**	**Min**	**Median**	**Product**
SVM Linear SVC (A)	0,605	0,613	**0,642**	0,631	0,590	0,617
LogisticRegression (A)	0,596	0,601	**0,602**	0,591	0,591	0,596
RandomForest (A)	0,580	0,595	**0,630**	0,594	0,559	0,598
Naive Bayes (A)	**0,613**	0,598	0,581	0,583	0,604	0,603
MLP (A)	0,618	0,626	**0,663**	0,639	0,612	0,634
ExtraTrees (A)	0,544	0,583	**0,626**	0,608	0,531	0,596
KNN (A)	**0,531**	0,517	0,524	0,508	0,511	0,514
CNN (A)	0,671	0,640	0,630	0,696	0,664	**0,673**
GRUPO B	0,649	0,610	0,602	**0,687**	0,602	0,627
GRUPO C	0,717	0,703	0,693	**0,731**	0,706	0,717
GRUPO D	0,602	**0,615**	0,602	0,566	0,600	0,566

Tabela 11 – Resultados em termos de *macro F1* encontrados com as técnicas de combinações para a base TD.

Utilizando todos os 40 classificadores combinados, a acurácia superou as combinações de *LogisticRegression*, *RandomForest*, *Naive Bayes*, *ExtraTrees* e KNN ao alcançar 89,1% utilizando a função *Sum/Avg*. Em relação, ao *baseline* superou *Naive Bayes* e KNN. Além disso, igualou o resultado de *ExtraTrees*.

Em relação ao *macro F1*, o Grupo D superou as combinações de *LogisticRegression*, *Naive Bayes* e KNN ao atingir 61,5% também aplicando a função de combinação *Sum/Avg*, porém não superou nenhum resultado do *baseline*.

6.2 BASE DE DADOS ZW

A Tabela 12, apresenta quais hiperparâmetros foram testados e quais alcançaram o melhor desempenho para cada um dos algoritmos avaliados com cada um dos métodos extratores.

Parâmetros	TF	TFIDF	W2V	GLOVE	FastText
		SVM Linear SVC			
Kernel:[Linear, Sigmoid, rbf] Gamma: [0.1, 1, 0.5]	Kernel: rbf Gamma: 0.1	Kernel: linear Gamma: 0.1	Kernel: rbf Gamma: 1	Kernel: linear Gamma: 0.1	Kernel: linear Gamma: 0.1
		Logistic Regression			
penalty: [l1, l2]	penalty: l1	penalty: l1	penalty: l1	penalty: l1	penalty: l1
		RandomForest			
n_estimators: [10, 20, 50]	n_estimators: 20	n_estimators: 50	n_estimators: 50	n_estimators: 20	n_estimators: 50
		Naive Bayes			
alpha: [0.1, 0.5, 1] fit_prior: [False, True]	alpha: 1 fit_prior: True	alpha: 0.1 fit_prior: True	alpha: 0.1 fit_prior: True	alpha: 0.1 fit_prior: True	alpha: 1 fit_prior: True
		MLP			
activation: [relu, logistic] solver: [adam, lbfgs]	activation: relu solver: lbfgs	activation: relu solver: lbfgs	activation: relu solver: adam	activation: relu solver: adam	activation: relu solver: adam
		ExtraTrees			
n_estimators: [10, 20, 50]	n_estimators: 50	n_estimators: 20	n_estimators: 50	n_estimators: 50	n_estimators: 50
		KNN			
algorithm: [auto, ball_tree, kd_tree] n_neighbors: [3, 5]	algorithm: auto n_neighbors: 5	algorithm: auto n_neighbors: 3	algorithm: auto n_neighbors: 5	algorithm: auto n_neighbors: 5	algorithm: auto n_neighbors: 5
		CNN			
activation: [sigmoid, softmax] optimizer: [rmsprop, adam]	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop

Tabela 12 – Avaliação de hiperparâmetros para base ZW.

Ao analisar a Tabela 12, percebe-se que apenas em *LogisticRegression* e CNN não houveram mudanças quando aplicados a métodos extratores diferentes.

Para CNN, as propriedades avaliadas foram:

- *Activation* - podendo assumir os valores *sigmoid* ou *softmax*.
- *Optimizer* - podendo assumir os valores *rmsprop* ou *adam*.

Para *LogisticRegression* avaliamos apenas a propriedade *penalty*, podendo assumir os valores L1 ou L2.

Para todos os extratores, CNN alcançou os melhores resultados com *activation: sigmoid* e *optimizer: rmsprop*, já *LogisticRegression* os melhores resultados foram atingidos com *penalty: L1*.

Após a seleção de hiperparâmetros, deve-se realizar a classificação monolítica, utilizando a base de dados de teste.

A Tabela 13 mostra a acurácia alcançada por cada um dos classificadores na classificação monolítica. De acordo com estes resultados, o modelo que atingiu o melhor desempenho foi *SVM Linear SVC* aplicado ao método extrator TfIdf. Este modelo atingiu 82,4%, sendo treinado com os hiperparâmetros *Kernel: Linear* e *Gamma: 0.1*, como pode ser visto na Tabela 12.

Algoritmo	TF	TfIdf	W2V	GLOVE	FastText
SVM Linear SVC	0,813	**0,824**	0,802	0,678	0,758
LogisticRegression	**0,822**	0,810	0,774	0,698	0,760
RandomForest	0,803	**0,814**	0,752	0,707	0,753
Naive Bayes	**0,802**	0,787	0,600	0,659	0,654
MLP	**0,817**	0,793	0,790	0,733	0,791
ExtraTrees	0,792	**0,812**	0,743	0,714	0,747
KNN	0,712	0,697	**0,736**	0,673	0,696
CNN	**0,811**	0,797	0,804	0,794	0,775

Tabela 13 – Acurácia dos algoritmos avaliados no *baseline* da base de dados ZW de acordo com os hiperparâmetros encontrados e apresentados na Tabela 12.

Na Tabela 14, apresentam-se as informações de *macro F1* para a classificação monolítica da base de dados ZW. De acordo com o exposto, percebe-se que o modelo que alcançou o melhor resultado foi MLP com TF, atingindo 76,6%. Este modelo foi treinado utilizando as propriedades *activation: relu* e *solver: lbfgs*.

Algoritmo	TF	TfIdf	W2V	GLOVE	FastText
SVM Linear SVC	0,733	**0,755**	0,714	0,269	0,599
LogisticRegression	**0,754**	0,726	0,671	0,438	0,627
RandomForest	0,726	**0,746**	0,590	0,507	0,579
Naive Bayes	**0,740**	0,702	0,586	0,517	0,618
MLP	**0,766**	0,712	0,713	0,570	0,721
ExtraTrees	0,719	**0,751**	0,559	0,472	0,552
KNN	0,419	0,381	**0,670**	0,538	0,599
CNN	0,739	0,706	**0,752**	0,696	0,746

Tabela 14 – Macro F1 dos algoritmos avaliados no *baseline* de acordo com os hiperparâmetros encontrados e apresentados na Tabela 12.

Ainda de acordo com as Tabelas 13 e 14, é importante destacar que modelos diferentes alcançaram os melhores resultados. SVM com TfIdf obteve uma maior precisão, ou seja, classificou corretamente um número maior de instâncias de teste, por outro lado MLP com TF obteve um melhor desempenho ao analisar a média de acertos por classe, alcançando um *macro F1* maior entre todos os classificadores.

A Figura 6 apresenta a distribuição dos algoritmos de aprendizagem testados no *baseline*, utilizando o conjunto de validação para o Grupo A, e na Tabela 15, é exibido o *Oracle* para os mesmos classificadores.

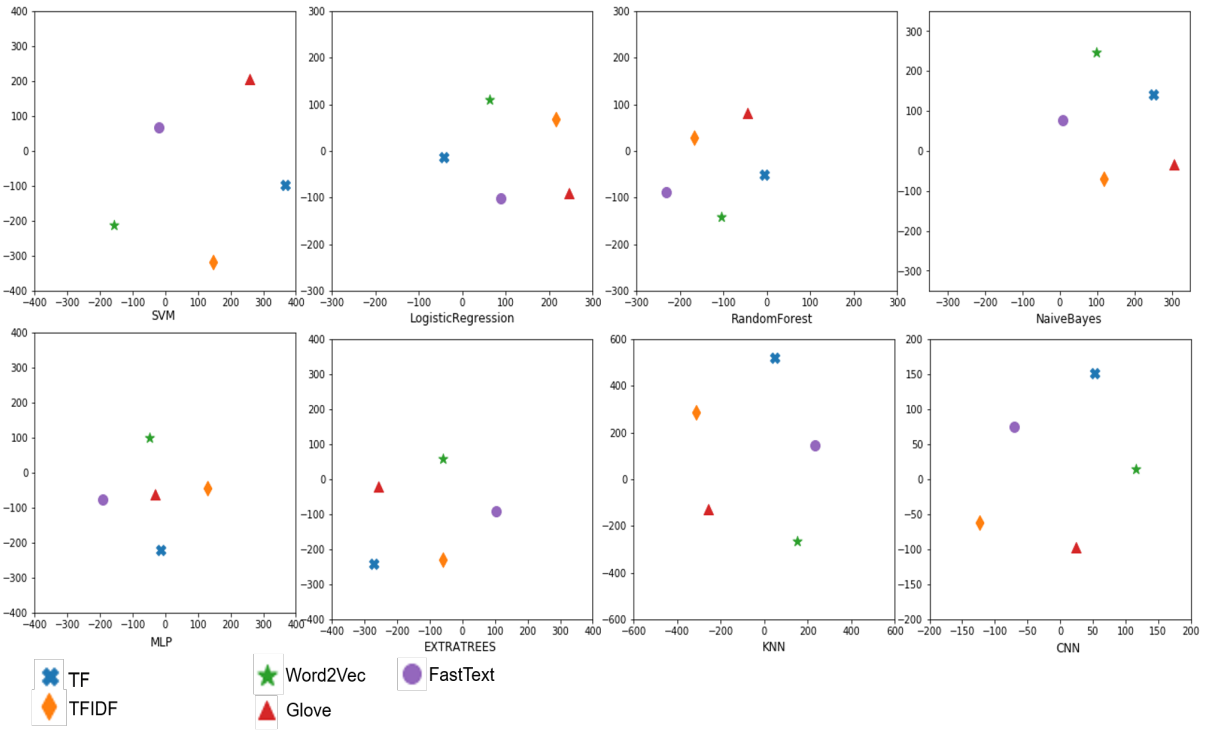

Figura 6 – Projeção comportamental para cada algoritmo avaliado da base ZW com as extrações de características.

A análise de sensibilidade da Figura 6, não possibilitou definir qual o grupo de classificadores é mais disperso, ou seja, qual das combinações são mais promissoras, isso é evidenciado pelos dados apresentados na Tabela 15, pois a maioria das combinações apresenta *Oracle* semelhante, exceto o *Naive Bayes*, conjunto de classificadores que contempla o maior número de acertos.

Algoritmo	**Acertos**	**Acurácia**	**Macro F1**
SVM (A)	2.540	0,910	0,812
LogisticRegression (A)	2.547	0,912	0,821
RandomForest (A)	2.535	0,908	0,815
Naive Bayes (A)	2.640	0,946	0,900
MLP (A)	2.593	0,929	0,861
ExtraTrees (A)	2.548	0,913	0,822
KNN (A)	2.576	0,923	0,839
CNN (A)	2.585	0,926	0,863

Tabela 15 – *Oracle* para cada subgrupo do Grupo A. (A) Significa que os experimentos são do Grupo A.

Este comportamento pode ser comprovado ao analisar a Figura 7, nesse histograma apresenta-se a quantidade de classificadores que acertam cada instância de entrada do

conjunto de testes. Ainda de acordo com o histograma, é possivel identificar que similarmente ao apresentado na Figura 6, a contagem de acertos também é semelhante para todos os grupos de classificadores e que *Naive Bayes* apresenta uma variação maior entre a quantidade de classificadores que acertam cada padrão, apesar dessa informação não ter ficado evidente no gráfico de dissimilaridade.

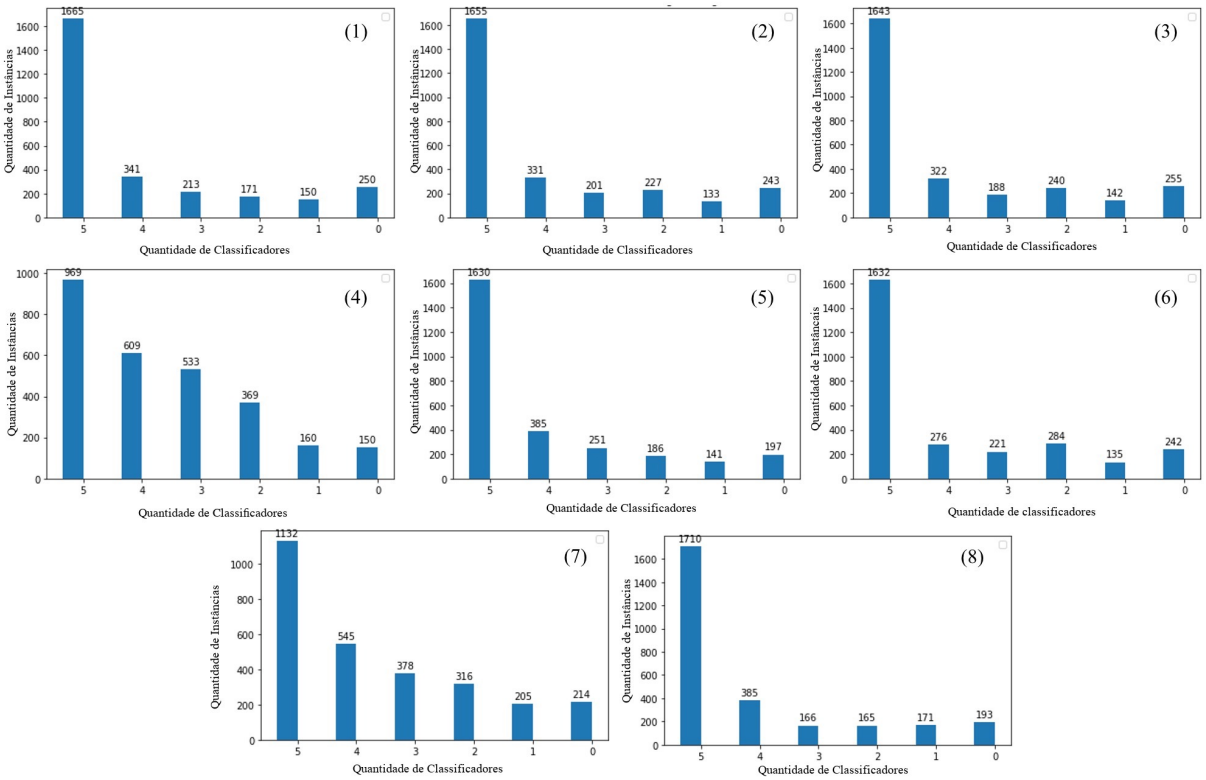

Figura 7 – Quantidade de classificadores (eixo x) que acertam as instâncias de teste (eixo y) para a base de dados ZW. (1) - SVM, (2) - *LogisticRegression*, (3) - *RandomForest*, (4) - *Naive Bayes*, (5) - MLP, 6 - *ExtraTrees*, (7) - KNN e (8) - CNN

Após a análise do Grupo A, realizou-se a exploração visual de todos os classificadores. a Figura 8 apresenta o gráfico de dissimilaridade para todos os classificadores.

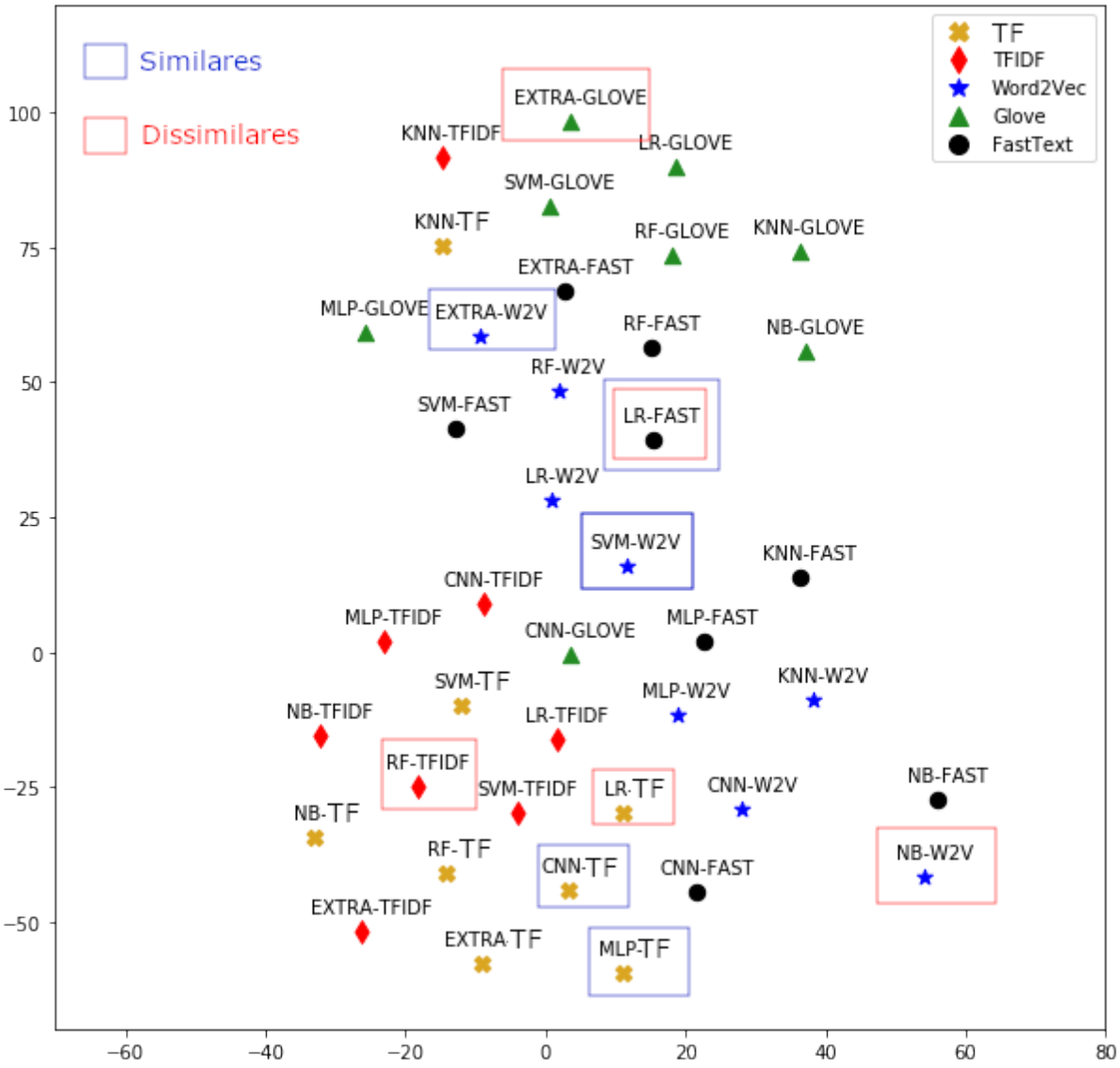

Figura 8 – Gráfico de dissimilaridade para cada algoritmo avaliado na base ZW independente do método extrator de características.

A Figura 8 que apresenta a dissimilaridade de todos os classificadores abordados para a base ZW, possibilitou a seleção dos Grupos B e C de classificadores, cuja composição ficou da seguinte forma:

B (NB-W2V)+(LR-FAST)+(LR-TF)+(RF-TFIDF)+(EXTRA-GLOVE)

C (MLP-TF)+(LR-FAST)+(SVM-W2V)+(CNN-TF)+(EXTRA-W2V)

A Tabela 16 mostra que ao combinar os acertos do Grupo B, obtêm-se um número maior de instâncias classificadas corretamente, isso em relação aos acertos do Grupo C. Além disso, ao analisar novamente a Tabela 15, percebe-se que o Grupo B supera os resultados alcançados pelo Grupo A. Nesse contexto, o Grupo B em número de instâncias

classificadas corretamente, é inferior apenas ao Grupo D. No entanto, a quantidade de classificadores que compõem o Grupo B é menor.

GRUPO	Acertos	Acurácia	Macro F1
GRUPO B	2.659	0,953	0,903
GRUPO C	2.562	0,918	0,838
GRUPO D	2.753	0,986	0,971

Tabela 16 – *Oracle* para os Grupos B, C e D da base de dados ZW.

Após a seleção dos Grupos B e C de classificadores, realizou-se as combinações para todos os grupos. A Tabela 17, mostra a acurácia encontrada por cada um dos grupos combinados.

De acordo com os dados apresentados, o melhor resultado foi encontrado utilizando a função *Max* para o Grupo C, combinação que alcançou 82,6%. Além disso, as combinações *Product* para CNN (A) e *Sum* para o Grupo B superaram SVM com TfIdf, melhor resultado do *baseline*, ambos atingindo 82,5%.

Além das combinações que superaram todos os resultados do *baseline*, é importante destacar que KNN obteve o pior resultado com classificação monolítica e também ao combinar vários classificadores, mesmo assim superou seu melhor resultado do *baseline* ao utilizar a função de combinação *Max*.

Algoritmo	Voting	Sum/Avg	Max	Min	Median	Product
SVM Linear SVC (A)	0,796	0,809	**0,810**	0,799	0,804	0,806
LogisticRegression (A)	0,785	0,804	**0,812**	0,805	0,786	0,805
RandomForest (A)	0,776	0,793	**0,801**	0,788	0,774	0,794
Naive Bayes (A)	0,770	0,771	0,708	0,713	**0,773**	0,748
MLP (A)	0,814	0,816	**0,821**	0,815	0,815	0,818
ExtraTrees (A)	0,764	0,800	**0,807**	0,789	0,765	0,791
KNN (A)	0,750	0,742	**0,760**	0,721	0,745	0,720
CNN (A)	0,818	0,820	0,815	0,824	0,822	**0,825**
GRUPO B	0,810	**0,825**	0,699	0,712	0,811	0,793
GRUPO C	0,802	0,815	**0,826**	0,806	0,805	0,814
GRUPO D	0,808	**0,813**	0,764	0,716	0,808	0,717

Tabela 17 – Acurácia alcançada com as técnicas de combinações para os grupos A, B, C e D da base de dados ZW.

De acordo com a Tabela 18, que mostra os resultados alcançados com as combinações em termos de *macro F1*, o melhor resultado foi encontrado combinando CNN através da função *Median*, atingindo 77,3%. Essa combinação superou o melhor resultado da classificação monolítica, alcançada pelo classificador MLP aplicado a TF.

Apesar da combinação de CNN ter superado o *baseline*, ao analisar as outras combinações percebe-se que há um decréscimo em relação a classificação monolítica, pois todas as combinações do Grupo A obtiveram melhores resultados no *baseline*, exceto CNN.

Algoritmo	**Voting**	**Sum/Avg**	**Max**	**Min**	**Median**	**Product**
SVM Linear SVC (A)	0,684	0,715	**0,724**	0,691	0,704	0,709
LogisticRegression (A)	0,663	0,700	**0,725**	0,698	0,666	0,702
RandomForest (A)	0,638	0,673	**0,695**	0,658	0,632	0,674
Naive Bayes (A)	0,709	**0,714**	0,668	0,671	0,707	0,699
MLP (A)	0,737	0,742	**0,755**	0,734	0,738	0,744
ExtraTrees (A)	0,604	0,686	**0,718**	0,667	0,604	0,668
KNN (A)	0,584	0,518	**0,600**	0,493	0,565	0,484
CNN (A)	0,758	0,756	0,743	0,768	**0,773**	0,770
GRUPO B	0,724	**0,758**	0,661	0,670	0,723	0,736
GRUPO C	0,708	0,734	**0,760**	0,738	0,712	0,735
GRUPO D	0,715	**0,725**	0,626	0,496	0,710	0,495

Tabela 18 – *Macro F1* encontrado com as técnicas de combinações para os grupos A, B, C e D da base de dados ZW.

6.3 BASE DE DADOS TD+ZW

A Tabela 19, apresenta quais hiperparâmetros foram avaliados e quais apresentaram o melhor comportamento para cada máquina de aprendizagem.

Parâmetros	**TF**	**TFIDF**	**W2V**	**Glove**	**FastText**
		SVM Linear SVC			
Kernel:[Linear, Sigmoid, rbf] Gamma: [0.1, 1, 0.5]	Kernel: linear Gamma: 0.1	Kernel: linear Gamma: 0.1	Kernel: rbf Gamma: 1	Kernel: rbf Gamma: 0.5	Kernel: rbf Gamma: 1
		Logistic Regression			
penalty: [l1, l2]	penalty: l1	penalty: l1	penalty: l1	penalty: l1	penalty: l1
		RandomForest			
n_estimators: [10, 20, 50]	n_estimators: 50	n_estimators: 50	n_estimators: 50	n_estimators: 50	n_estimators: 50
		Naive Bayes			
alpha: [0.1, 0.5, 1] fit_prior: [False, True]	alpha: 0.5 fit_prior: False	alpha: 0.5 fit_prior: False	alpha: 0.5 fit_prior: True	alpha: 0.1 fit_prior: True	alpha: 0.1 fit_prior: True
		MLP			
activation: [relu, logistic] solver: [adam, lbfgs]	activation: relu solver: lbfgs	activation: relu solver: adam	activation: relu solver: adam	activation: relu solver: adam	activation: relu solver: adam
		ExtraTrees			
n_estimators: [10, 20, 50]	n_estimators: 50	n_estimators: 20	n_estimators: 50	n_estimators: 50	n_estimators: 50
		CNN			
activation: [sigmoid, softmax] optimizer: [rmsprop, adam]	activation: sigmoid optimizer: rmsprop	activation: softmax optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop	activation: sigmoid optimizer: rmsprop
		KNN			
algorithm: [auto, ball_tree, kd_tree] n_neighbors: [3, 5]	algorithm: auto n_neighbors: 3	algorithm: auto n_neighbors: 3	algorithm: auto n_neighbors: 5	algorithm: ball_tree n_neighbors: 5	algorithm: auto n_neighbors: 5

Tabela 19 – Avaliação de hiperparâmetros da base de dados TD+ZW.

De acordo com a tabela 19, *RandomForest* obteve melhor desempenho no conjunto de validação, sempre com a propriedade *n_estimators: 50*.

Além disso, semelhante aos experimentos de análise de hiperparâmetros para as bases de dados **TD** e **ZW**, o algoritmo *LogisticRegression* manteve-se estável, sempre obtendo melhores resultados com a propriedade *penalty: L1*.

A Tabela 20 mostra a acurácia alcançada por cada um dos classificadores na classificação monolítica. De acordo com estes resultados, dois classificadores obtiveram o melhor desempenho, em ambos a acurácia atingiu 91,1%:

- *SVM Linear SVC* utilizando TfIdf como método extrator de características. As configurações de hiperparâmetros adotadas foram *kernel: linear* e *gamma: 0.1*.
- *LogisticRegression* utilizando TF como método extrator de características. As configurações de hiperparâmetros adotadas foram *penalty: L1*.

Algoritmo	**TF**	**TfIdf**	**W2V**	**GLOVE**	**FastText**
SVM Linear SVC	0,904	**0,911**	0,827	0,788	0,826
LogisticRegression	**0,911**	0,908	0,813	0,770	0,818
RandomForest	0,896	**0,899**	0,768	0,766	0,779
Naive Bayes	**0,888**	0,863	0,720	0,731	0,747
MLP	**0,905**	0,890	0,824	0,779	0,835
ExtraTrees	**0,895**	0,896	0,763	0,771	0,762
KNN	**0,802**	0,407	0,779	0,749	0,761
CNN	0,890	0,870	0,876	0,888	**0,892**

Tabela 20 – Acurácia da classificação monolítica no baseline de acordo com os parâmetros encontrados e apresentados na Tabela 19 para a base de dados TD+ZW.

Na Tabela 21, apresentam-se as informações de *macro F1* para a classificação monolítica da base de dados TD+ZW. De acordo com esses dados, percebe-se que os resultados são semelhantes aos apresentados em termos de acurácia, pois os dois melhores classificadores são os mesmos. No entanto, *LogisticRegression* com TF obteve um desempenho ligeiramente superior em relação a SVM com TfIdf, atingiram 88,5% e 88,1%, respectivamente. Essa informação indica que *LogisticRegression* com TF, tem desempenho mais semelhante entre as classes do que SVM com TFIDF.

Algoritmo	TF	TfIdf	W2V	GLOVE	FastText
SVM Linear SVC	0,873	**0,881**	0,756	0,674	0,750
LogisticRegression	**0,885**	0,879	0,732	0,632	0,738
RandomForest	0,859	**0,862**	0,634	0,630	0,646
Naive Bayes	**0,852**	0,814	0,638	0,607	0,671
MLP	**0,875**	0,858	0,757	0,667	0,771
ExtraTrees	**0,860**	0,861	0,611	0,631	0,587
KNN	**0,671**	0,344	0,658	0,606	0,607
CNN	0,846	0,811	0,827	0,843	**0,857**

Tabela 21 – *Macro F1* da classificação monolítica no baseline de acordo com os parâmetros encontrados e apresentados na Tabela 19 para a base de dados TD+ZW.

A Figura 9 apresenta o gráfico de dissimilaridade para os classificadores avaliados no *baseline*, a partir da análise sensitiva dessa imagem, percebe-se que o grupo de classificadores MLP apresenta uma maior dispersão entre os pontos, esse fato indica que os classificadores possuem um comportamento complementar.

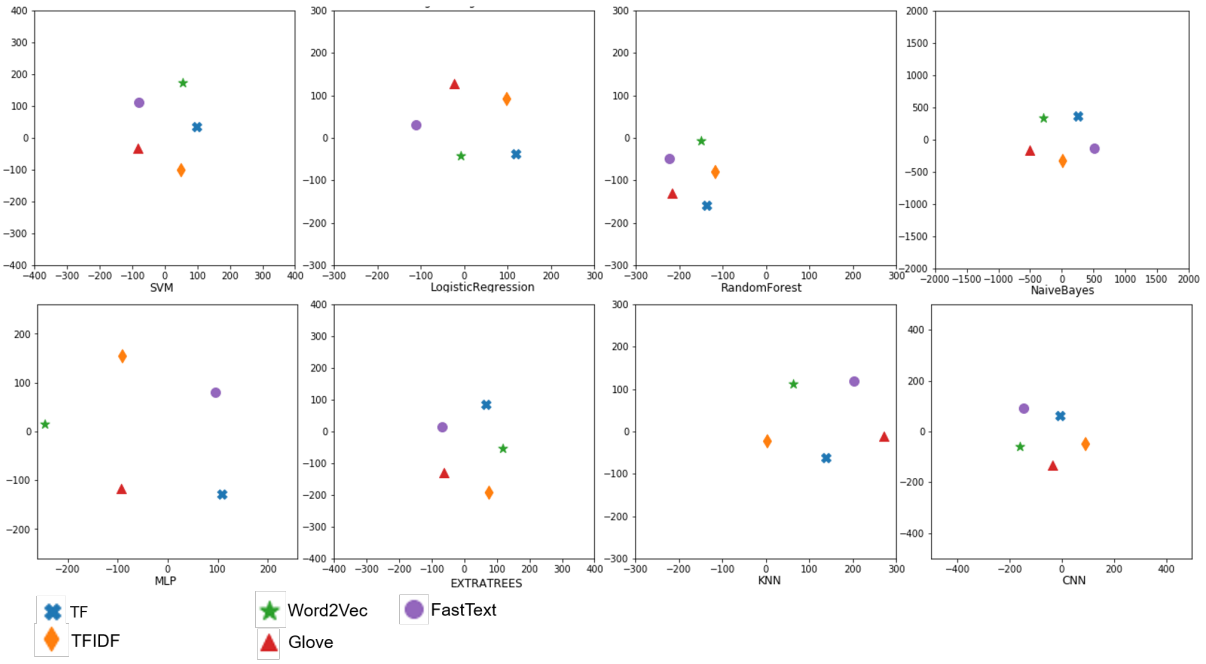

Figura 9 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD+ZW de acordo com as extrações de características.

O comportamento complementar entre os classificadores MLP, pode ser comprovado após análise do *Oracle* apresentado na Tabela 22. De acordo com esses dados, MLP possui o maior potencial combinatório, atingindo um total de 4.806 acertos, esses valores representam uma acurácia de 96,6% e um *macro F1* de 95,9%.

Algoritmo	Acertos	Acurácia	Macro F1
SVM (A)	4.768	0,958	0,947
LogisticRegression (A)	4.772	0,959	0,946
RandomForest (A)	4.724	0,950	0,928
Naive Bayes (A)	4.708	0,946	0,931
MLP (A)	4.806	0,966	0,959
ExtraTrees (A)	4.721	0,949	0,927
KNN (A)	4.639	0,933	0,867
CNN (A)	4.779	0,961	0,955

Tabela 22 – *Oracle* para cada subgrupo do Grupo A. (A) Significa que os experimentossão do Grupo A

Além do *Oracle*, que demonstra a quantidade total de instâncias corretamente classificadas por um grupo de classificadores, é importante analisar a quantidade de classificadores que acertam cada instância, esse dado pode indicar qual classificador obtêm o melhor resultado com voto majoritário.

A Figura 10 apresenta o histograma de contagem de classificadores para o grupo *Naive Bayes*(similar) e MLP (dissimilar), nele é possível verificar que MLP além de acertar mais instâncias com 3 classificadores ou mais, apresenta uma variação menor entre a quantidade de acertos, fato que justifica a dispersão apresentada no gráfico de dissimilaridade.

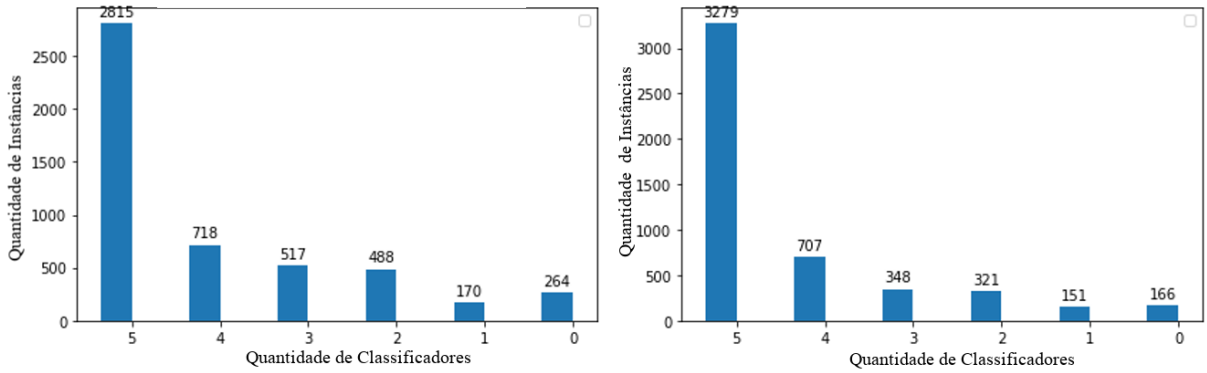

Figura 10 – Quantidade de classificadores que acertam os padrões para os algoritmos Naive Bayes (esquerda) e MLP (direita).

Após a análise do grupo A, realizou-se a exploração visual de todos os classificadores. A Figura 11 apresenta a dissimilaridade dos classificadores no plano 2D para a base de dados TD+ZW. Esse gráfico apresenta o comportamento dos modelos de aprendizagem avaliados no *baseline* considerando-se as predições do conjunto de validação.

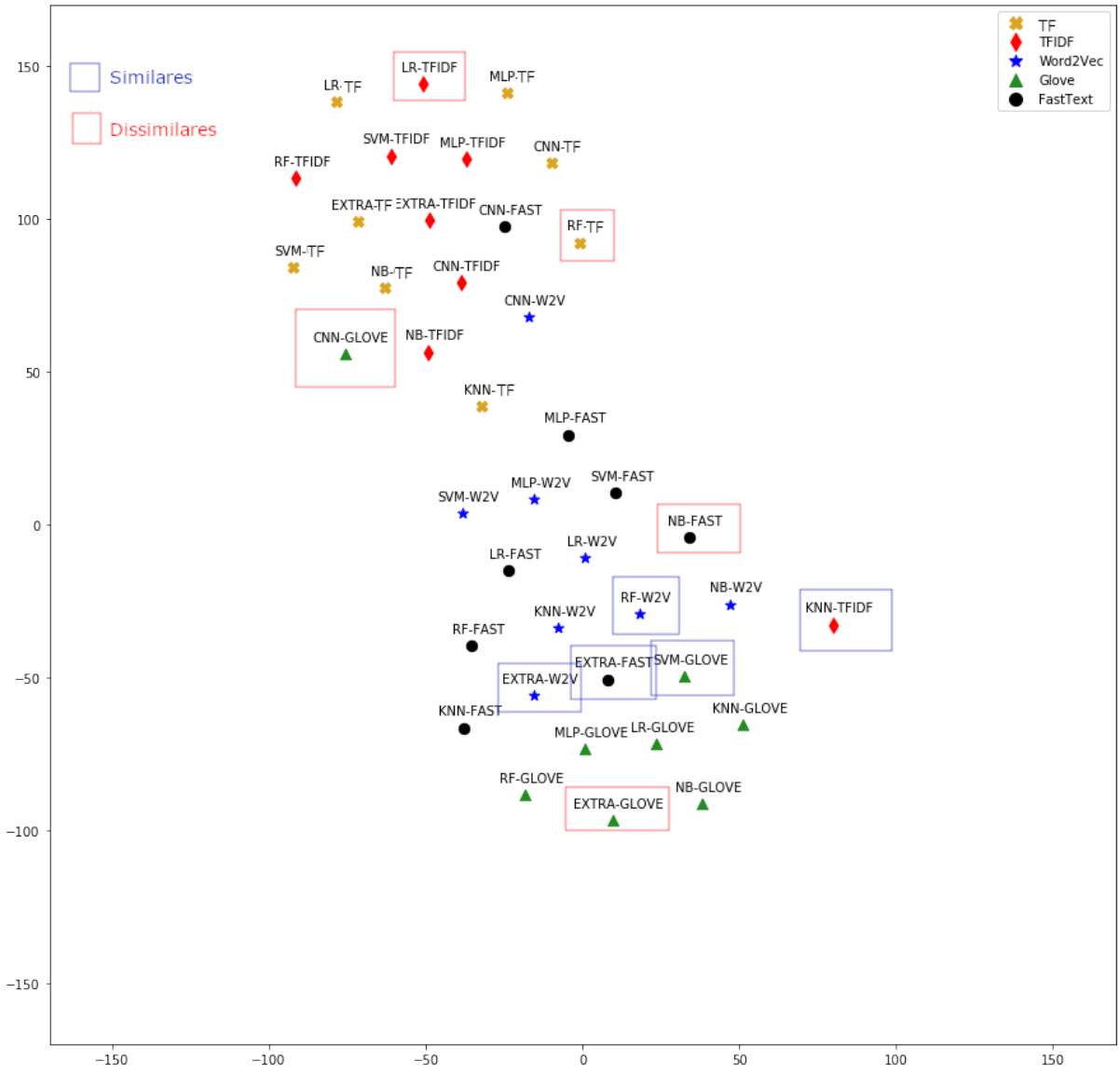

Figura 11 – Gráfico de dissimilaridade para cada algoritmo avaliado na base TD+ZW independente do método extrator de características.

A análise do gráfico possibilita a seleção dos grupos dissimilares e similares de classificadores, B e C, respectivamente. Os grupos formados foram:

B (EXTRA-GLOVE)+(NB-FAST)+(CNN-GLOVE)+(RF-TF)+(LR-TFIDF)

C (SVM-GLOVE)+(EXTRA-W2V)+(EXTRA-FAST)+(RF-W2V)+(KNN-TFIDF)

A Tabela 23 mostra que ao combinar os acertos do Grupo B, obtêm-se um número maior de acertos tanto em relação ao Grupo A, quanto em relação ao Grupo C, sendo superado apenas pelo Grupo D, no entanto o grupo D é necessário combinar 40 classificadores.

GRUPO	Acertos	Acurácia	Macro F1
GRUPO B	4.809	0,967	0,960
GRUPO C	4.606	0,926	0,855
GRUPO D	4.949	0,995	0,993

Tabela 23 – *Oracle* dos grupos B, C e D para a base de dados TD+ZW.

Após a seleção dos grupos B e C, realizou-se a combinação para cada um dos 4 grupos. A Tabela 24, apresenta a acurácia encontrada para todas as combinações avaliadas.

Os dados apresentados mostram que o melhor resultado foi alcançado utilizando a função *Sum/Avg* com o grupo CNN, atingindo 90,9%. No entanto, esse valor ainda é inferior a classificação monolítica de *LogisticRegression* com *TF* e SVM com TfIdf.

Apesar de nenhuma combinação superar o melhor resultado do *baseline*, as combinações MLP, KNN e CNN superaram os melhores resultados dos respectivos algoritmos na classificação monolítica. Além disso, o Grupo B superou todas as combinações do Grupo A, exceto MLP e CNN, enquanto que o Grupo C obteve resultado inferior em todas as combinações em relação ao Grupo B.

Algoritmo	Voting	Sum/Avg	Max	Min	Median	Product
SVM Linear SVC	0,875	0,896	**0,900**	0,899	0,877	0,900
LogisticRegression	0,868	**0,902**	0,897	0,891	0,872	0,901
RandomForest	0,837	0,880	**0,884**	0,872	0,842	0,877
Naive Bayes	0,829	0,825	0,806	0,793	**0,830**	0,806
MLP	0,879	0,900	0,900	0,902	0,882	**0,907**
ExtraTrees	0,825	0,877	**0,885**	0,873	0,832	0,874
KNN	0,791	**0,811**	0,502	0,493	0,802	0,493
CNN	0,904	**0,909**	0,900	0,898	0,909	0,907
GRUPO B	0,899	0,897	0,832	0,841	**0,902**	0,874
GRUPO C	0,776	**0,787**	0,495	0,499	0,785	0,500
GRUPO D	0,889	**0,897**	0,510	0,492	0,888	0,493

Tabela 24 – Acurária encontrados com as técnicas de combinações para os grupos A e B da base de dados TD+ZW.

A Tabela 25 mostra o *macro F1* encontrado para todas as combinações. Esses dados mostram que, similarmente a acúrácia, com *macro F1* CNN utilizando a função *Sum/Avg* obteve o melhor desempenho, alcançando 88,0%, mesmo assim o resultado foi inferior a SVM com TfIdf, melhor resultado da classificação monolítica.

Ao comparar as combinações do grupo A com a classificação monolítica do *baseline*, percebe-se que MLP e CNN superaram as classificações individuais, alcançando 87,7% utilizando a função *product* e 88,0% utilizando a função *sum/avg*, respectivamente.

Algoritmo	Voting	Sum/Avg	Max	Min	Median	Product
SVM Linear SVC	0,825	0,860	**0,865**	0,862	0,831	0,865
LogisticRegression	0,811	**0,867**	0,859	0,849	0,818	0,865
RandomForest	0,747	0,829	**0,833**	0,811	0,762	0,824
Naive Bayes	0,758	0,756	0,740	0,733	**0,764**	0,741
MLP	0,836	0,866	0,867	0,871	0,841	**0,877**
ExtraTrees	0,720	0,824	**0,840**	0,815	0,737	0,819
KNN	0,620	**0,666**	0,414	0,408	0,659	0,408
CNN	0,872	**0,880**	0,861	0,862	0,879	0,877
GRUPO B	0,859	0,858	0,768	0,787	**0,867**	0,828
GRUPO C	0,613	0,629	0,413	0,424	**0,641**	0,424
GRUPO D	0,845	**0,857**	0,451	0,411	0,844	0,411

Tabela 25 – Macro F1 encontrados com as técnicas de combinações para os grupos A e B da base de dados TD+ZW.

6.4 CONSIDERAÇÕES FINAIS

Nesta seção, são apresentadas as conclusões sobre os resultados encontrados com os experimentos das 3 bases de dados.

A Tabela 26 apresenta os melhores resultados do *baseline* e das combinações para cada base de dados.

Base de Dados	Melhor Classificador Baseline	Resultado Acc %	Resultado F1 %	Melhor Combinação	Resultado Acc %	Resultado F1 %
TD	LR+TF	90,6	70,6	Grupo C	90,8	73,1
ZW	SVM+TFIDF	82,4	75,5	Grupo C	82,6	76,0
TD+ZW	LR+ TF	91,1	88,5	CNN	90,9	88,0

Tabela 26 – Melhores resultados encontrados no baseline e na combinação de classificadores para as 3 bases de dados.

De acordo com a Tabela 26, pode-se concluir que:

- Com a base TD a acurácia atingiu um resultado interessante, mas o F1 ficou muito abaixo, isso indica que as máquinas estão com dificuldade de separar duas ou mais classes. De acordo com a literatura discurso de ódio e linguagem ofensiva são padrões que se confundem e portanto explicaria esse baixo desempenho relacionado a *F1-score*;

- Com a base ZW, a acurácia não atingiu resultados expressivos, porém *F1-score* fica próximo, evidenciando um certo padrão nos acertos por classe, talvez a dificuldade

aqui seja a classe *none*, que possui discursos variados que venha a confundir a máquina;

- Utilizando a base ZW+TD houve uma pequena melhoria na acurácia, mas a questão mais relevante é a significativa melhora de F1. Acreditamos que deve-se ao fato de ter adicionado mais registros do tipo *hate*, proporcionando mais informações para a aprendizagem.

- Utilizamos a exploração visual para selecionar o melhor subconjunto de classificadores para avaliar as combinações, no entanto em 2 das 3 bases utilizadas o grupo mais próximo apresentou resultados melhores. Embora o *Oracle* tenho sido maior para o grupo dissimilar.

Após análise dos resultados encontrados nestes experimentos, realizou-se uma análise comparativa com outros trabalhos encontrados na literatura.

De acordo com a Tabela 27, percebe-se que os trabalhos analisados sempre utilizam as métricas *F1-Score* ou acurácia para avaliar os experimentos. Além disso, a maioria dos experimentos são realizados com uma das bases de dados que adotamos ou com uma combinação envolvendo as bases ZW e TD.

Autor	**Base de dados**	**Resultado**	
		Acc (%)	**F1 (%)**
(Watanabe; Bouazizi; Ohtsuki, 2018)	Combine 3	78,4	-
(DAVIDSON *et al.*, 2017)	TD	-	90,0
(GIBERT *et al.*, 2018)	LS[1]	78,0	-
(GAYDHANI *et al.*, 2018)	Combine 3	95,6	-
(GAMBÄCK; SIKDAR, 2017)	Zeerak	-	78,3
(WASEEM; HOVY, 2016)	ZW	-	73,9
Proposto	TD	90,8	73,1
Proposto	ZW	82,6	76,0
Proposto	TD+ZW	91,1	88,0

Tabela 27 – Melhores resultados alcançados em trabalhos anteriores e neste trabalho.

Avaliando os trabalhos que adotaram apenas a base de dados TD, temos:

- (DAVIDSON *et al.*, 2017) que alcançou 90,0% de *F1-Score*. A extração de características foi realizada com o método TfIdf com variação de *1-gram* a *3-gram*. A base de dados é rotulada em *Hate*, *Offensive* e *Non-offensive*, além disso 100% dos dados são utilizados no treinamento.

- O método proposto que atingiu 73,1% com o Grupo C.

[1] Disponível em: <https://github.com/aitor-garcia-p/hate-speech-dataset>

De acordo com esses resultados, percebe-se um desempenho inferior do método proposto em relação ao autor mencionado, entretanto há 2 aspectos a considerar-se:

1. Em (DAVIDSON *et al.*, 2017), não há descrição em relação a como foi calculado o *F1-Score*, se foi *macro*, *micro* ou outra forma. Porém, ao analisar as informações da matriz de confusão, que está disponível nos resultados do autor, percebe-se que não foi utilizada a configuração *macro*.

2. Segundo (HEMKER, 2018), os resultados encontrados por (DAVIDSON *et al.*, 2017) foram alcançados com a acurácia. Nesse contexto o método que está sendo proposto teria superado, pois alcançou 90,8%.

Em relação a base de dados ZW, 2 trabalhos foram analisados:

- (WASEEM; HOVY, 2016) analisa o impacto de caracteristicas extra-linguisticas e *n-grams* a nível de caracter. O autor utiliza *LogisticRegression* com *10-fold cross validation* para classificar as instâncias em *racism*, *sexism* e *none* e alcança um *F1-Score* de 73,89 % sem utilizar o gênero como informação adicional e 73,93% utilizando o gênero.

- O método proposto superou (WASEEM; HOVY, 2016), alcançando 76,0% com o Grupo C.

Ainda de acordo com a Tabela 27, 3 trabalhos combinaram bases de dados:

- (Watanabe; Bouazizi; Ohtsuki, 2018) utiliza uma combinação de 3 datasets, o conjunto resultante é rotulado em *Hate*, *Offensive* e *Clean* e dividido em treinamento, validação e testes, as três partes foram balanceadas. Os experimentos foram realizados utilizando o algoritmo *j48graft* através do *toolkit weka*, esse algoritmo alcançou 78,4% de acurácia para classificação ternária.

- (GAYDHANI *et al.*, 2018) realizou experimentos com a combinação de 3 bases de dados, o conjunto resultante é rotulado em *Hatefull*, *Offensive* e *Clean*. A base de dados foi dividida em treinamento e teste na proporção de 80% e 20%, respectivamente. Além disso, utiliza TfIdf para extração de características, variando de *1-gram* a *3 n-gram* e normalização de L1 e L2. Para avaliar os resultados, *10-fold cross validation* e *LogisticRegression* foram utilizados, alcançando uma acurácia de 95,6% e obtendo o melhor resultado entre os trabalhos analisados.

- O método proposto, alcançou 91,1% de acurácia com *LogisticRegression*, superando (Watanabe; Bouazizi; Ohtsuki, 2018), porém com desempenho inferior a (GAYDHANI *et al.*, 2018).

7 CONCLUSÃO

Nesta dissertação de mestrado, apresentamos um modelo de seleção de classificadores para aplicar técnicas de combinação, utilizando a exploração visual e a análise do *Oracle* na deteção automática de *hate speech*. A Seção 7.1 descreve as últimas considerações sobre esta pesquisa. A Seção 7.2 mostra as principais contribuições dessa pesquisa. Em seguida, a Seção 7.3 mostra atividades que podem ser utilizadas para dar continuidade a este trabalho.

7.1 CONSIDERAÇÕES FINAIS

A tarefa de automatizar detecção de *hate speech* tem se mostrado necessária, porém árdua, poís até mesmo a definição sobre o que é considerado *hate speech*, ainda é muito discutida. Essa dificuldade ganha escala no contexto das redes sociais, que são ferramentas que proporcionam um grande volume de informações diariamente e que além disso, tornaram-se um canal de comunicação, que possibilita qualquer pessoa expressar opiniões de qualquer parte do mundo e sobre qualquer tema ou pessoa, com pouco ou nenhum controle sobre o sentido dessas opiniões.

Diversos trabalhos já foram desenvolvidos utilizando diferentes métodos e modelos de aprendizagem de máquina para detecção de *hate speech*, tanto para classificação binária quanto para múltiplas classes. Tradicionalmente, utiliza-se classificação monolítica para esse tipo de tarefa. No entanto, sabe-se que sistemas de múltiplos classificadores têm se mostrado eficiente em comparação com o mecanismo de classificação monolítica, porém para que este método seja eficiente é necessário que o grupo de classificadores que serão combinados, possuam características complementares, ou seja, apresentem comportamentos distintos utilizando o mesmo conjunto de dados. Este trabalho, apresentou uma proposta que visa selecionar um subgrupo de classificadores com essas características dentro de um *pool*, fazendo uma análise visual de um plano 2D e verificando a conformidade desse subgrupo com análise do *Oracle*.

Para treinar esses classificadores, adotamos 5 métodos extratores de características diferentes: TF e TfIdf utilizam a abordagem *Bag-of-Words*, enquanto que *Word2Vec*, *Glove* e *FastText* adotam a abordagem *Word Embedding*. Na abordagem *Bag-of-Words* a extração de características desconsidera a ordem e o contexto das palavras dentro do documento, por outro lado, a abordagem *Word Embedding* transforma o corpus em um vetor de características de similaridade sintática e semântica.

Para verificar se classificadores apresentam comportamentos de classificação heterogêneos, projetamos cada classificador em um plano 2D, sendo que cada ponto deste plano representa um classificador do *pool*. Para posicionar os pontos no plano, calculamos as dis-

tâncias euclidianas entre todos os classificadores. Para calcular essa distância utilizamos o coeficiente *double-fault measure* para gerar a matriz de dissimilaridade.

A matriz de dissimilaridade possui muitas dimensões, variando de acordo com a quantidade de classificadores do *pool*, tornando a representação gráfica desses dados complexa. Devido a isso, fizemos a redução de dimensionalidade utilizando t-SNE, uma ferramenta para visualizar dados de alta-dimensão em dimensão reduzida, preservando ao máximo as características comportamentais de cada classificador.

Nossa proposta foi aplicada em três bases de dados diferentes. Duas delas, foram retiradas de trabalhos anteriores e uma foi desenvolvida na fase de experimentos desta pesquisa. Nas três bases, foram realizados experimentos com múltiplas classes, e o método proposto se mostrou eficiente em duas das bases de dados avaliadas, superando os resultados encontrados na classificação monolítica. No entanto, os subgrupos que alcançaram os melhores resultados, foram os que continham os classificadores mais próximos de acordo a projeção dos mesmos no plano 2D, apesar do grupo contabilizar um *Oracle* menor que o subgrupo mais distante.

7.2 CONTRIBUIÇÕES

A principal contribuição deste trabalho é a proposta da metodologia, que dado um *pool* de classificadores, apresenta um mecanismo para selecionar um subconjunto, em tese, com características complementares que possam obter resultados satisfatórios com técnicas de combinação.

Além disso, podemos destacar que, ao combinar classificadores é possível melhorar os resultados da classificação monolítica, isso pode ser evidenciado com o cálculo do *Oracle*, tendo em vista que para todas as combinações avaliadas o *Oracle*, superou os acertos individuais. No entanto, é importante observar que compor o grupo que será combinado é determinante para conseguir ganhos de desempenho.

Outro aspecto importante que deve ser considerado, diz respeito aos resultados encontrados combinando os grupos de classificadores selecionados com diferentes técnicas de combinação. De acordo com esses dados, pode-se concluir que:

- KNN sempre consegue atingir resultados melhores combinando os classificadores em relação ao *baseline*. Isso evidencia que a utilização de técnicas de combinação tendem a conseguir melhores resultados com a junção de classificadores fracos;

- Para as 3 bases, houve ao menos um grupo combinado que superou seu correspondente, como resultado individual. Esse dado evidência além da eficiência das combinações, que essa eficiência depende do conjunto selecionado.

Além disso, a utilização de diferentes bases de dados com algoritmos distintos obtendo os melhores resultados, e utilizando o mesmo protocolo de experimentos, permitiu concluir que o desempenho da atividade de classificação também é influenciado por:

- **Dados para treinamento** - Sabe-se que o número de instâncias para treinamento é um fator importante no processo de aprendizado. No entanto, a qualidade dos dados e a proporção por classe, interferem diretamente no desempenho dos classificadores utilizados.

- **Ajustes de hiperparâmetros** - Modelos de aprendizagem se comportam diferente tanto em relação aos dados quanto aos ajustes de hiperparâmetros, isso é evidenciado ao verificar que, para a maioria dos algoritmos avaliados, os hiperparâmetros que atingiram o melhor desempenho foram variados, inclusive dentro do mesmo conjunto de dados, modificando apenas o método extrator de características.

- **Extração de características** - A forma utilizada para extrair as características do *corpus* tem influência direta no desempenho das classificações. Na classificação monolítica, os melhores resultados para cada base de dados, foram obtidos com a abordagem *Bag-of-Words*.

- **Métrica de Avaliação** - O desempenho do classificador é influenciado diretamente pela forma como avaliamos seus resultados, isso é evidenciado, quando ao utilizar a base de dados ZW, o classificador *SVM Linear SVC* com TfIdf alcançou a maior acurácia. No entanto, MLP com TF obteve o maior *macro f1-score*. Este caso, indica que o algoritmo que acerta mais padrões de teste, pode apresentar desempenho ótimo na classe mais predominante da base de dados, aumentando a acurácia. No entanto, esse mesmo algoritmo pode errar muitos padrões das classes restantes, gerando um desempenho inferior em relação a *macro f1-score*.

7.3 TRABALHOS FUTUROS

Nossa proposta consiste em selecionar um subconjunto de classificadores "distantes" entre si e avaliar técnicas de combinação com esse subconjunto. Para calcular essa distância utilizamos o coeficiente *double-fault measure*.

Para dar continuidade a este trabalho propomos avaliar o coeficiente *Kappa*, cuja implementação calcula o nível de concordância entre classificadores (BARBOSA; FENG, 2010). Essa atividade vai reposicionar os classificadores no plano 2D e a análise de sensibilidade possibilitará selecionar conjuntos de classificadores diferentes dos selecionados neste trabalho.

Outra sugestão para prosseguir com os experimentos, é balancear as bases de dados, pois como mencionado anteriormente a proporção dos dados por classe no *corpus*, pode

influenciar no processo de generalização de aprendizagem, dificultando a predição de instâncias rotuladas com a classe minoritária. Outro aspecto que pode ser considerado com o avanço da pesquisa, está relacionado ao ajuste de hiperparâmetros. Nas máquinas de aprendizagem *ExtraTrees* e *RandomForest*, percebemos que o melhor resultado sempre foi alcançado com o maior valor avaliado para *n_estimators*, então sugerimos testar valores maiores que 50.

Avaliar redes complexas no pré-processamento dos dados, é outra sugestão que pode dar bons resultados. As redes complexas utilizam o conceito de grafos e no contexto do processamento de linguagem natural, pode ser utilizado para calcular relações entre palavras dentro do texto e o grau de importância no significado, de acordo com alguma métrica de centralidade.

Como última sugestão para trabalhos futuros, deve-se considerar a utilização de métodos de seleção dinâmica de classificadores. Neste método, a combinação é realizada com base na região de competência, sendo que a classe final do padrão é dada por um único classificador, aquele que apresenta os melhores resultados na região, ou por um *ensemble* que contenha o melhor classificador, dentro do *pool* (CRUZ *et al.*, 2020).

REFERÊNCIAS

BARBOSA, L.; FENG, J. Robust sentiment detection on twitter from biased and noisy data. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. USA: Association for Computational Linguistics, 2010. (COLING '10), p. 36–44.

Batool, R.; Khan, W. A.; Hussain, M.; Maqbool, J.; Afzal, M.; Lee, S. Towards personalized health profiling in social network. In: *2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012)*. [S.l.: s.n.], 2012. p. 760–765.

BERRY, M. *Computational Information Retrieval*. [S.l.]: Society for Industrial and Applied Mathematics, 2001. (Proceedings in Applied Mathematics).

BISPO, T. D. *Arquitetura LSTM para classificação de discursos de ódio cross-lingual Inglês-PtBR*. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DE SERGIPE, 2018.

CATAL, C.; NANGIR, M. A sentiment classification model based on multiple classifiers. *Applied Soft Computing*, v. 50, p. 135 – 141, 2017.

CAVALCANTI, G. D.; OLIVEIRA, L. S.; MOURA, T. J.; CARVALHO, G. V. Combining diversity measures for ensemble pruning. *Pattern Recogn. Lett.*, Elsevier Science Inc., USA, v. 74, n. C, p. 38–45, abr. 2016.

CHEN, X.; XU, L.; LIU, Z.; SUN, M.; LUAN, H. Joint learning of character and word embeddings. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2015. (IJCAI'15), p. 1236–1242. ISBN 9781577357384.

Cheng, K.; Song, Z.; Yue, Y.; Shan, F.; Guo, X. Classifier selection method based on multiple diversity measures. In: *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. [S.l.: s.n.], 2019. p. 47–53.

CRUZ, R. M.; CAVALCANTI, G. D.; TSANG, I. R.; SABOURIN, R. Feature representation selection based on classifier projection space and oracle analysis. *Expert Systems with Applications*, v. 40, n. 9, p. 3813 – 3827, 2013.

CRUZ, R. M. O.; HAFEMANN, L. G.; SABOURIN, R.; CAVALCANTI, G. D. C. Deslib: A dynamic ensemble selection library in python. *Journal of Machine Learning Research*, v. 21, n. 8, p. 1–5, 2020.

DAFTARY, A.-M.; DEVEREUX, P.; ELLIOTT, M. Discrimination, depression, and anxiety among college women in the trump era. *Journal of Gender Studies*, Routledge, v. 0, n. 0, p. 1–14, 2020.

DAMASCHK, M.; DÖNICKE, T.; LUX, F. Multiclass text classification on unbalanced, sparse and noisy data. In: *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*. Turku, Finland: Linköping University Electronic Press, 2019. p. 58–65.

DAVIDSON, T.; WARMSLEY, D.; MACY, M.; WEBER, I. Automated hate speech detection and the problem of offensive language. *International AAAI Conference on Web and Social Media*, 2017.

DUCHARME, D.; COSTA, L.; DIPIPPO, L.; HAMEL, L. Svm constraint discovery using knn applied to the identification of cyberbullying. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER .... *Proceedings of the International Conference on Data Mining (DMIN)*. [S.l.], 2017. p. 111–117.

FAUZI, M. A.; YUNIARTI, A. Ensemble method for indonesian twitter hate speech detection. *Indonesian Journal of Electrical Engineering and Computer Science*, v. 11, n. 1, p. 294–299, 2018.

GAMBÄCK, B.; SIKDAR, U. K. Using convolutional neural networks to classify hate-speech. In: *Proceedings of the First Workshop on Abusive Language Online*. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 85–90.

GAYDHANI, A.; DOMA, V.; KENDRE, S.; BHAGWAT, L. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and TFIDF based approach. *CoRR*, abs/1809.08651, 2018.

GIBERT, O. de; PEREZ, N.; GARCÍA-PABLOS, A.; CUADROS, M. Hate speech dataset from a white supremacy forum. In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 11–20.

GISBRECHT, A.; SCHULZ, A.; HAMMER, B. Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, v. 147, p. 71 – 82, 2015.

GRöNDAHL, T.; PAJOLA, L.; JUUTI, M.; CONTI, M.; ASOKAN, N. All you need is "love": Evading hate speech detection. In: *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. New York, NY, USA: Association for Computing Machinery, 2018. (AISec '18), p. 2–12.

Hansen, L. K.; Salamon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 12, n. 10, p. 993–1001, 1990.

HASSANIEN, A.; ELHOSENY, M. *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*. [S.l.]: Springer International Publishing, 2019. (Advanced Sciences and Technologies for Security Applications).

HEMKER, K. *Data Augmentation and Deep Learning for Hate Speech Detection*. Dissertação (Mestrado) — IMPERIAL COLLEGE LONDON, 2018.

HO, T.; HULL, J.; SRIHARI, S. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 16, n. 1, p. 66–75, 1994.

HOWARD, P. N.; KOLLANYI, B.; WOOLLEY, S. Bots and automation over twitter during the us election. *Computational Propaganda Project: Working Paper Series*, 2016.

HU, J.-R.; WANG, M.; LU, F. Covid-19 and asian american pacific islanders. *Journal of General Internal Medicine*, Springer, p. 1–2, 2020.

JIN, P.; ZHANG, Y.; CHEN, X.; XIA, Y. Bag-of-embeddings for text classification. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2016. (IJCAI'16), p. 2824–2830.

JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; MIKOLOV, T. Bag of tricks for efficient text classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, 2017. p. 427–431.

Kittler, J.; Hatef, M.; Duin, R. P. W.; Matas, J. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 3, p. 226–239, 1998.

Kuncheva, L. I. Clustering-and-selection model for classifier combination. In: *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*. [S.l.: s.n.], 2000. v. 1, p. 185–188 vol.1.

Kuncheva, L. I. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 2, p. 281–286, 2002.

KUNCHEVA, L. I. *Combining Pattern Classifiers: Methods and Algorithms*. 2nd. ed. [S.l.]: Wiley Publishing, 2014.

Kuncheva, L. I.; Whitaker, C. J.; Shipp, C. A.; Duin, R. P. W. Is independence good for combining classifiers? In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. [S.l.: s.n.], 2000. v. 2, p. 168–171 vol.2.

LAI, S.; XU, L.; LIU, K.; ZHAO, J. Recurrent convolutional neural networks for text classification. In: *Twenty-ninth AAAI conference on artificial intelligence*. [S.l.: s.n.], 2015.

LI, S.; ZONG, C.; WANG, X. Sentiment classification through combining classifiers with multiple feature sets. In: . [S.l.: s.n.], 2007. p. 135–140.

Lim, J.; Jang, H.; Lim, J.; Park, S. Normalization of gene/protein names in biological literatures using vector-space model. In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.: s.n.], 2007. p. 390–393.

LIU, Y.; LIU, Z.; CHUA, T.-S.; SUN, M. Topical word embeddings. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. [S.l.]: AAAI Press, 2015. (AAAI'15), p. 2418–2424.

LODHI, H.; SAUNDERS, C.; SHAWE-TAYLOR, J.; CRISTIANINI, N.; WATKINS, C. Text classification using string kernels. *Journal of Machine Learning Research*, v. 2, n. Feb, p. 419–444, 2002.

LU, H. juan; AN, C. lin; ZHENG, E. hui; LU, Y. Dissimilarity based ensemble of extreme learning machine for gene expression data classification. *Neurocomputing*, v. 128, p. 22 – 30, 2014.

MAATEN, L. van der. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, v. 15, n. 93, p. 3221–3245, 2014.

MAATEN, L. van der; HINTON, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, v. 9, p. 2579–2605, 2008.

MAATEN, L. Van der; HINTON, G. Visualizing non-metric similarities in multiple maps. *Machine learning*, Springer, v. 87, n. 1, p. 33–55, 2012.

MATHEW, B.; SAHA, P.; THARAD, H.; RAJGARIA, S.; SINGHANIA, P.; MAITY, S. K.; GOYAL, P.; MUKHERJEE, A. Thou shalt not hate: Countering online hate speech. In: *Proceedings of the International AAAI Conference on Web and Social Media*. [S.l.: s.n.], 2019. v. 13, n. 01, p. 369–380.

MATHEW, B.; SAHA, P.; THARAD, H.; RAJGARIA, S.; SINGHANIA, P.; MAITY, S. K.; GOYAL, P.; MUKHERJE, A. Thou shalt not hate: Countering online hate speech. In: *Thirteenth International AAAI Conference on Web and Social Media*. [S.l.: s.n.], 2019.

MCINNES, L.; HEALY, J.; MELVILLE, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Mestry, S.; Singh, H.; Chauhan, R.; Bisht, V.; Tiwari, K. Automation in social networking comments with the help of robust fasttext and cnn. In: *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*. [S.l.: s.n.], 2019. p. 1–4.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. [S.l.: s.n.], 2013.

MINAEE, S.; KALCHBRENNER, N.; CAMBRIA, E.; NIKZAD, N.; CHENAGHLU, M.; GAO, J. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*, 2020.

MOLNAR, C. *Interpretable Machine Learning*. Lulu.com, 2020. ISBN 9780244768522. Disponível em: <https://books.google.com.br/books?id=jBm3DwAAQBAJ>.

NWEKE, H. F.; TEH, Y. W.; MUJTABA, G.; AL-GARADI, M. A. Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions. *Information Fusion*, v. 46, p. 147 – 170, 2019. ISSN 1566-2535.

PARK, J. H.; FUNG, P. One-step and two-step classification for abusive language detection on twitter. In: *Proceedings of the First Workshop on Abusive Language Online*. Vancouver, BC, Canada: Association for Computational Linguistics, 2017. p. 41–45. Disponível em: <https://www.aclweb.org/anthology/W17-3006>.

PATCHIN, J. W.; HINDUJA, S. Bullies move beyond the schoolyard: A preliminary look at cyberbullying. *Youth Violence and Juvenile Justice*, v. 4, n. 2, p. 148–169, 2006.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543.

Ponti Jr., M. P. Combining classifiers: From the creation of ensembles to the decision fusion. In: *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*. [S.l.: s.n.], 2011. p. 1–10.

RUTA, D.; GABRYS, B. An overview of classifier fusion methods. *Computing and Information systems*, v. 7, n. 1, p. 1–10, 2000.

SCHARWÄCHTER, E.; MÜLLER, E. Does terrorism trigger online hate speech? on the association of events and time series. *arXiv preprint arXiv:2004.14733*, 2020.

SCIARRONE, F. An extension of the q diversity metric from single-label to multi-label and multi-ranking multiple classifier systems for pattern classification. In: . [S.l.: s.n.], 2012. v. 1, p. 6–10.

Sharma, Y.; Agrawal, G.; Jain, P.; Kumar, T. Vector representation of words for sentiment analysis using glove. In: *2017 International Conference on Intelligent Communication and Computational Techniques (ICCT)*. [S.l.: s.n.], 2017. p. 279–284.

SHI, T.; LIU, Z. *Linking GloVe with word2vec*. 2014.

SHIPP, C. A.; KUNCHEVA, L. I. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, v. 3, n. 2, p. 135 – 148, 2002.

SILVA, L.; MONDAL, M.; CORREA, D.; BENEVENUTO, F.; WEBER, I. Analyzing the targets of hate in online social media. 2016.

SMITH, P. K. Bullying: Definition, types, causes, consequences and intervention. *Social and Personality Psychology Compass*, Wiley Online Library, v. 10, n. 9, p. 519–532, 2016.

SMITS, P. C. Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection. *IEEE Transactions on Geoscience and Remote Sensing*, v. 40, n. 4, p. 801–813, 2002.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems 25*. [S.l.]: Curran Associates, Inc., 2012. p. 2951–2959.

Subrahmanian, V. S.; Azaria, A.; Durst, S.; Kagan, V.; Galstyan, A.; Lerman, K.; Zhu, L.; Ferrara, E.; Flammini, A.; Menczer, F. The darpa twitter bot challenge. *Computer*, v. 49, n. 6, p. 38–46, 2016.

THIN, D. V.; LE, L. S.; NGUYEN, N. L.-T. Nlp@ uit: Exploring feature engineer and ensemble model for hate speech detection at vlsp 2019. *TRAINING*, v. 5, p. 3–51, 1991.

WARNER, W.; HIRSCHBERG, J. Detecting hate speech on the world wide web. In: *Proceedings of the Second Workshop on Language in Social Media*. Montréal, Canada: Association for Computational Linguistics, 2012. p. 19–26.

WASEEM, Z. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In: *Proceedings of the First Workshop on NLP and Computational Social Science*. Austin, Texas: Association for Computational Linguistics, 2016.

WASEEM, Z.; HOVY, D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: *Proceedings of the NAACL Student Research Workshop*. San Diego, California: Association for Computational Linguistics, 2016. p. 88–93.

Watanabe, H.; Bouazizi, M.; Ohtsuki, T. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*, v. 6, p. 13825–13835, 2018.

YAO, L.; MAO, C.; LUO, Y. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, p. 7370–7377, 07 2019.

ZAMPIERI, M.; MALMASI, S.; NAKOV, P.; ROSENTHAL, S.; FARRA, N.; KUMAR, R. Predicting the type and target of offensive posts in social media. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 1415–1420.

ZAMPIERI, M.; MALMASI, S.; NAKOV, P.; ROSENTHAL, S.; FARRA, N.; KUMAR, R. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019. p. 75–86.

ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level convolutional networks for text classification. In: CORTES, C.; LAWRENCE, N. D.; LEE, D. D.; SUGIYAMA, M.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems 28*. [S.l.]: Curran Associates, Inc., 2015. p. 649–657.

ZHANG, Z.; LUO, L. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *ArXiv*, abs/1803.03662, 2019.

ZHANG, Z.; LUO, L. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, IOS Press, v. 10, n. 5, p. 925–945, 2019.

ZHAO, R.; ZHOU, A.; MAO, K. Automatic detection of cyberbullying on social networks based on bullying features. In: *Proceedings of the 17th International Conference on Distributed Computing and Networking*. New York, NY, USA: Association for Computing Machinery, 2016. (ICDCN '16).

ZHU, Y.; MOH, M.; MOH, T. Multi-layer text classification with voting for consumer reviews. In: *2016 IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2016. p. 1991–1999.