Universidade Estadual de Maringá
Centro de Tecnologia - Departamento de Informática
Bacharelado em Ciência da Computação
Trabalho de Conclusão de Curso II - 2018

ALESSANDRA HARUMI IRIGUTI

EXTRAÇÃO DE CARACTERÍSTICAS PARA A CLASSIFICAÇÃO DE ESTRUTURA RETÓRICA EM RESUMOS CIENTÍFICOS

MARINGÁ - PR
2018

ALESSANDRA HARUMI IRIGUTI

EXTRAÇÃO DE CARACTERÍSTICAS PARA A CLASSIFICAÇÃO DE ESTRUTURA RETÓRICA EM RESUMOS CIENTÍFICOS

Trabalho de Conclusão de Curso de Graduação apresentado ao Departamento de Informática da Universidade Estadual de Maringá, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Valéria Delisandra Feltrim

MARINGÁ - PR
2018

RESUMO

A classificação de estrutura retórica é uma tarefa de Processamento de Linguagem Natural na qual se busca identificar os componentes retóricos de um texto, bem como os seus relacionamentos. Os resultados de tal classificação podem ser aplicados em diferentes cenários, tais como sumarização automática, avaliação automática de redações e ferramentas de auxílio à escrita. Nesse contexto, o objetivo deste trabalho foi extrair e avaliar o impacto de diferentes conjuntos de características na implementação de classificadores retóricos para resumos científicos escritos em português. Em especial, foram avaliadas características superficiais, características profundas e características extraídas a partir de modelos de *word embeddings*. Também foi avaliado o desempenho das diferentes combinações de características. As características superficiais foram extraídas como valores TF-IDF e o teste χ^2 foi usado para redução de dimensionalidade. Como características profundas foram usadas as implementadas pelo classificador AZPort. Para a extração de características a partir de *word embeddings* foram usados modelos previamente treinados, sendo eles Word2Vec (CBOW e Skip-gram), Wang2Vec (CBOW e Skip-gram) e GloVe. Cada modelo de *embeddings* foi avaliado com as dimensões 50, 100, 300, 600 e 1000. Os diferentes conjuntos de características foram usados no treinamento de classificadores induzidos os seguintes algoritmos de aprendizado supervisionado: *Support Vector Machines* (kernels linear e RBF), *Naive Bayes* (variações Gaussian e Bernoulli), *Nearest Neighbors*, *Decision Trees* e *Conditional Random Fields* (CRF). Para treinamento e teste dos classificadores foram feitos por meio de validação cruzada de 10 partições sobre três *corpora* compostos por resumos de teses e dissertações da área de Ciência da Computação. O melhor resultado foi obtido pelo classificador CRF, como 94% de F1. Esse mesmo resultado foi alcançado com mais de uma combinação de características, sendo elas: (i) combinação das características extraídas usando o modelo Wang2Vec–Skip-gram de dimensão 100 com as características provenientes do AZPort; (ii) combinação de todas as características – TF-IDF, AZPort e *embeddings* extraídos com os modelos Word2Vec–Skip-gram e GloVe de dimensões 1000 e 300, respectivamente. A partir dos resultados, conclui-se que as características profundas foram fundamentais para o bom desempenho do CRF e que a combinação com *word embeddings* se mostrou válida.

Palavras-chaves: processamento de linguagem natural, classificação de estrutura retórica, vetores de distribuição.

ABSTRACT

Classification of rhetorical structure is a task of Natural Language Processing in which one seeks to identify the rhetorical components of a text, as well as their relationships. The results of such classification can be applied in different scenarios, such as automatic summarization, automatic evaluation of essays and writing tools. In this context, the objective of this work was to extract and evaluate the impact of different sets of features on the implementation of rhetorical classifiers for scientific abstracts written in Portuguese. In particular, we evaluated surface features, deep features and features extracted from word embeddings models. The performance of the different combinations of features was also evaluated. Surface characteristics were extracted as TF-IDF values and the χ^2 test was used for dimensionality reduction. As deep features were used those implemented by the AZPort classifier. For the extraction of characteristics from word embeddings, previously trained models were used, such as Word2Vec (CBOW and Skip-gram), Wang2Vec (CBOW and Skip-gram) and GloVe. Each template of embeddings was rated with 50, 100, 300, 600, and 1000 of dimensions. The different sets of features were used in training of classifiers induced the following supervised learning algorithms: Support Vector Machines (linear and RBF kernels), Naive Bayes (Gaussian and Bernoulli variations), Nearest Neighbors, Decision Trees, and Conditional Random Fields (CRF). For training and testing of classifiers were done through 10-fold cross-validation on three *corpora* composed by abstracts of theses and dissertations from the area of Computer Science. The best result was obtained by the CRF classifier, such as F1 like 94%. This same result was achieved with more than one combination of features: (i) combination of the features extracted using the Wang2Vec–Skip-gram model of 100 dimension with the features coming from the AZPort; (ii) combination of all features – TF-IDF, AZPort, and embedded extracted from Word2Vec–Skip-gram and GloVe models of 1000 and 300 dimensions, respectively. From the results, it was concluded that the deep features were fundamental for the good performance of the CRF and that the combination with word embeddings proved valid.

Key-words: natural language processing, rhetorical structure classification, word embeddings.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AZ	*Argumentative Zoning*
AZPort	*Argumentative Zoning Portuguese*
B-NB	Bernoulli *Naive Bayes*
CART	*Classification And Regression Trees*
CBOW	*Continuous Bag of Words*
CRF	*Conditional Random Fields*
DT	*Decision Trees*
FN	*False Negative*
FP	*False Positive*
G-NB	Gaussian *Naive Bayes*
IA	Inteligência Artificial
K-NN	K *Nearest Neighbors*
NB	*Naive Bayes*
PLN	Processamento de Linguagem Natural
RBF	*Radial-Basis Function*
SVM	*Support Vector Machines*
TF-IDF	*Term Frequency - Inverse Document Frequency*
TP	*True Positive*

SUMÁRIO

1 INTRODUÇÃO

Inteligência Artificial (IA) é um campo da Ciência da Computação em que, segundo Russell e Norvig (2016, p. 1), se tenta entender e construir entidades inteligentes. Uma das capacidades humanas é a escrita. Os estudos para que um computador seja capaz de extrair informações a partir de textos escritos são feitos na área de Processamento de Linguagem Natural (PLN). Há diversas tarefas que servem como objeto de estudo dentro de PLN, dentre elas, a classificação de textos e a análise discursiva.

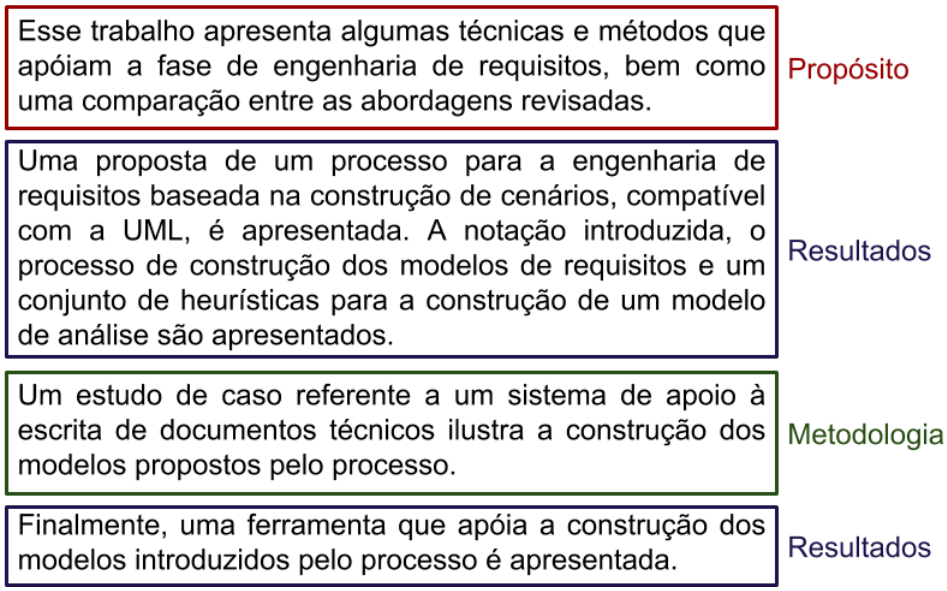
As estruturas discursivas são os padrões vistos em textos multi-sentenças (multi-clausal). Reconhecer esses padrões em termos dos elementos que os compõem é essencial para derivar e interpretar corretamente a informação no texto. Os elementos podem ser funções, cada uma realizada por uma ou mais cláusulas, em que a função servida pode ser com respeito ao discurso como um todo ou algum outro segmento do discurso (WEBBER; EGG; KORDONI, 2012). Tais elementos e suas posições são chamados de estrutura retórica. Uma estrutura retórica é formada por um conjunto de relações retóricas e são essas relações que definem como o conteúdo está relacionado e como cada parte do texto, também chamado de proposição, contribui para satisfazer as intenções do autor (ROMEIRO, 2016). A estrutura retórica pode ser representada de forma mais geral (para qualquer discurso) ou mais específica (para um gênero textual em particular), como no caso dos textos científicos.

Os linguistas Weissberg e Buker (1990), Swales e Feak (1994) investigaram estruturas retóricas específicas do gênero científico. Weissberg e Buker (1990) propuseram um modelo para a estruturação de resumos. Já Swales e Feak (1994) propuseram um modelo para a estruturação de introduções. A motivação desses pesquisadores foi investigar e auxiliar a escrita de textos científicos, uma tarefa reconhecidamente difícil, especialmente para escritores iniciantes. Outros autores que também propuseram modelos de estrutura retórica para textos científicos foram Anthony (1999) e Teufel e Moens (2002).

Para ilustrar a estrutura retórica de um resumo científico, o exemplo da Figura 1 mostra um resumo de dissertação com a sua estrutura destacada. Nele é possível notar que o texto segue uma estrutura prototípica do gênero, em que é dado destaque para os objetivos (propósito), métodos e resultados do trabalho sendo relatado.

A detecção automática das estruturas retóricas é geralmente feita por meio de classificadores treinados por algoritmos de aprendizado de máquina (AM). Segundo Russell e Norvig (2016, p. 456), classificação é "verificar se um objeto pertence a uma categoria". Por exemplo, identificar a classe de uma sentença baseado em suas características. Assim, técnicas de AM são utilizadas para a indução do classificador (LORENA; CARVALHO, 2007). A indução é feita

Figura 1 – Exemplo de identificação de estrutura retórica de resumo



Fonte: Resumo extraído de Junior (1998)

por meio de um indutor ou algoritmo de aprendizado. Dentre os diferentes tipos de algoritmos de aprendizado, alguns dos mais utilizados são os algoritmos supervisionados, por exemplo, *Support Vector Machine* (SVM), *Naive Bayes* (NB), *Nearest Neighbors* (K-NN), *Decision Trees* (DT) e *Conditional Random Fields* (CRF). Neles, o indutor recebe um conjunto de exemplos de treinamento para os quais o rótulo (*label*) da classe associada é conhecido (REZENDE, 2003).

Os algoritmos anteriormente enumerados, com exceção do CRF, foram avaliados em um Projeto de Iniciação Científica intitulado "Criação de Classificadores Retóricos para Resumos Científicos da PubMed". Neste projeto, três dos *corpora* utilizados estavam na língua portuguesa do Brasil. Para essas três coleções de resumos, o *f1-score* médio máximo obtido foi de 57% com um classificador SVM. Em tal processo, características extraídas superficialmente, tais como a posição da sentença no resumo, valores TF-IDFs para a sentença e para as sentenças anterior e posterior, foram usadas.

1.1 OBJETIVOS GERAL E ESPECÍFICO

O objetivo geral deste trabalho foi a implementação de classificadores retóricos sentenciais para resumos em português, coletados e manualmente anotados em trabalhos anteriores (FELTRIM; ALUÍSIO; NUNES, 2003; ANDREANI; FELTRIM, 2015), visando a avaliação de diferentes conjuntos de características, incluindo as extraídas a partir de modelos de *word*

embeddings.

Os objetivos específicos foram: (i) empregar *embeddings* já treinados para o português para extrair características e adaptar os classificadores já implementados; (ii) comparar os resultados dos novos classificadores com os obtidos em outros trabalhos; (iii) comparar os resultados obtidos neste trabalho com classificadores que utilizam atributos profundos, como os implementados pelo AZPort (FELTRIM, 2004).

1.2 JUSTIFICATIVA

Apesar de a maioria dos textos científicos serem escritos em inglês, muitos alunos brasileiros, que realizam um projeto de iniciação científica, se deparam com a escrita de projetos e relatórios. Muitas vezes, é a primeira vez que esses alunos devem escrever um texto científico. Dessa forma, ferramentas de auxílio à escrita científica podem ser úteis e facilitar o processo de escrita para esses alunos.

A motivação para o melhoramento dos classificadores retóricos é usá-los, em trabalhos futuros, como parte de ferramentas de auxílio de escrita em língua portuguesa. Procura-se assim aumentar o desempenho obtido em outros trabalhos, bem como avaliar a contribuição de diferentes conjuntos de características para a tarefa.

1.3 ORGANIZAÇÃO DO DOCUMENTO

O restante do documento está organizado da seguinte forma: nas Seções 2 e 3 são apresentadas uma fundamentação teórica sobre Inteligência Artificial e *Word Embeddings*, respectivamente; na Seção 4 são descritos os materiais e os métodos utilizados neste trabalho; na Seção 5 são mostrados e analisados os resultados obtidos; e, por fim, na Seção 6 são apresentadas as conclusões.

2 INTELIGÊNCIA ARTIFICIAL

Alan Turing propôs, em 1950, o Teste de Turing, a fim de definir as capacidades que um computador deve ter para ser inteligente (RUSSELL; NORVIG, 2016, p. 1). Entre essas capacidades estão:

- Processamento de linguagem natural;
- Representação de conhecimento;
- Raciocínio automatizado; e
- Aprendizado de máquina.

A primeira capacidade, processamento de linguagem natural, permite a comunicação em uma determinada linguagem. A segunda capacidade está relacionada a representação do conhecimento adquirido de uma forma que seja compreensível pelo computador, para seja possível utilizar a informação armazenada como subsídio nas tomadas de decisões (raciocínio). Por fim, o aprendizado de máquina capacita a máquina a se adaptar ao ambiente enquanto reconhece padrões (RUSSELL; NORVIG, 2016). Cada uma dessas capacidades estão relacionadas a diferentes subáreas da inteligência artificial (IA), nas quais pesquisadores investigam abordagens, técnicas e modelos que auxiliam na implementação de diferentes aspectos relacionados a essas capacidades.

Visto que este trabalho abrange duas dessas subáreas, as Seções 2.1 e 2.2 apresentam a fundamentação teórica sobre aprendizado de máquina e processamento de linguagem natural, respectivamente.

2.1 APRENDIZADO DE MÁQUINA

Segundo Rezende (2003), aprendizado de máquina (AM) é uma das áreas de IA na qual se tem como objetivo capacitar os computadores a adquirir conhecimento de forma automática. Para isso, tenta-se reproduzir o aprendizado indutivo, já que a indução é a forma de inferência de conhecimento mais usada pelo cérebro humano. No aprendizado indutivo, o aprendizado ocorre por meio da generalização da parte para o todo, ou seja, a conclusão, que é geral, é alcançada a partir de exemplos, que são específicos.

O aprendizado de máquina indutivo é dividido em supervisionado e não supervisionado. No aprendizado supervisionado, a classe dos exemplos é conhecida. O indutor (ou algoritmo de aprendizado) recebe exemplos – que consistem de um vetor de características (ou atributos)

e o rótulo da classe – e deve gerar um modelo que será capaz de classificar exemplos não rotulados. Esse processo é chamado de classificação, quando os rótulos a serem previstos são valores discretos, ou de regressão, quando são contínuos. *Support Vector Machine* (SVM), *K-Nearest Neighbors* (K-NN), *Naive Bayes* (NB), *Decision Trees* (DT) e *Conditional Random Fields* (CRF) são alguns exemplos de algoritmos de aprendizado supervisionado. Já no aprendizado não supervisionado, as classes não são conhecidas. No seu caso mais comum, o indutor busca encontrar agrupamentos de exemplos levando em conta a similaridade entre os vetores de atributos. O *K-means* é um exemplo de algoritmo não supervisionado.

Os algoritmos ainda podem ser divididos de acordo com os cinco paradigmas tradicionais de AM, simbólico, estatístico, baseado em exemplos, conexionista e evolutivo. Os algoritmos que foram utilizados neste trabalho se enquadram nos três primeiros paradigmas. Os algoritmos simbólicos geram modelos constituídos por símbolos a partir da análise de exemplos e contra-exemplos, como é o caso do *Decision Trees*. Já os estatísticos utilizam modelos baseados em estatística para encontrar uma aproximação do conceito induzido que seja aceitável, por exemplo, o SVM, o *Naive Bayes* e o CRF. Por fim, os baseados em exemplos mantêm os exemplos em memória e, no momento da classificação, baseiam sua resposta na distância entre o exemplo desconhecido e os exemplos conhecidos, como é o caso K-NN.

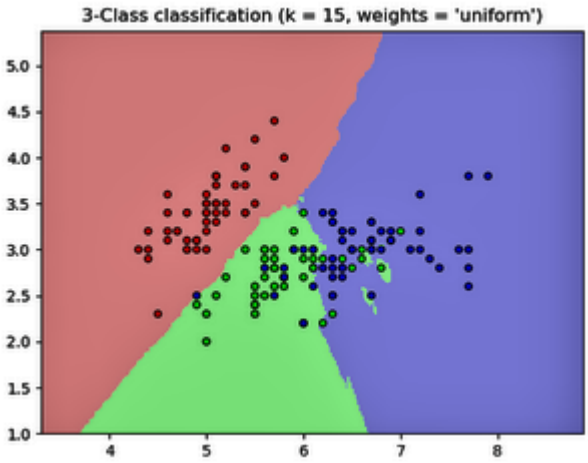
Nas Subseções a seguir, são detalhados os algoritmos de AM que foram usados neste trabalho, a saber: *K-Nearest Neighbors*, *Naive Bayes*, *Decision Trees*, *Support Vector Machines* e *Conditional Random Fields*.

2.1.1 *K-Nearest Neighbors*

O *K-Nearest Neighbors* (K-NN) é um algoritmo supervisionado baseado em exemplos. Segue uma abordagem "preguiçosa", uma vez que a fase de treino consiste apenas em armazenar os exemplos de treino. Na fase de teste, é calculada a Distância Euclidiana (Equação 2.1) entre o exemplo a ser predito com os dados da base de treino. A predição é dada usando-se a classe mais frequente observada nos k exemplos com as menores distâncias – vizinhos mais próximos.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{2.1}$$

Figura 2 – Classificação K-NN com 15 vizinhos mais próximos



Fonte: adaptado de Pedregosa *et al.* (2011)

Como exemplo, na Figura 2 é mostrada a classificação em três classes (representadas pelas cores rosa, azul e verde) utilizando $k = 15$ vizinhos mais próximos. O posicionamento de cada ponto representa sua coordenada e a sua cor representa a sua classe. Na Figura 2, o conjunto de pontos representa a base de dados classificada. Os pontos estão coloridos de acordo com sua classificação (rosa, azul ou verde). O fundo colorido mostra a maior concentração de pontos de cada classe. Para classificar um novo ponto, seria calculado sua Distância Euclidiana com todos os pontos presentes na figura. Dos 15 pontos com menores distâncias, a classe que mais se repete é a classe desse novo ponto.

2.1.2 *Naive Bayes*

O *Naive Bayes* é um modelo estatístico que considera todos os atributos de forma igualmente importante e independente (WITTEN *et al.*, 2016, p. 93). Por exemplo, considere a Tabela 1 com os dados de tempo de 14 dias e a possibilidade de jogar ou não jogar golfe. Na Tabela 2, temos esses dados de tempo com a contagem de quantas vezes o par atributo–valor ocorreu com os valores yes e no para $play$. Os valores para $play$ são: 9 ocorrências de yes (probabilidade de $9/14$) e 5 ocorrências de no (probabilidade de $5/14$).

Tabela 1 – Dados de tempo em 14 dias

Day	Outlook	Temperature	Humidity	Windy	Play
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Fonte: adaptado de Witten *et al.* (2016, p. 217)

Tabela 2 – Contagens e Probabilidades de dados do tempo

Outlook			Temperature			Humidity			Windy		
	yes	no		yes	no		yes	no		yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2
overcast	4	0	mild	4	2	normal	6	1	true	3	3
rainy	3	2	cool	3	1						
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5
rainy	3/9	2/5	cool	3/9	1/5						

Fonte: adaptado de Witten *et al.* (2016, p. 91)

Supondo o exemplo de um novo dia com os seguintes atributos–valores: $Outlook = sunny$; $Temperature = cool$; $Humidity = high$; e $Windy = true$. O valor de $play$ para esse novo dia seria calculado considerando, inicialmente, as frações dos atributos dados com a entrada yes – $2/9$, $3/9$, $3/9$ e $3/9$, para os atributos–valores do novo dia. Por fim, a fração de probabilidade de $play$ – $9/14$. Dessa forma, obtemos o $Likelihood$ para $play = yes$:

$$Likelihood(yes) = (2/9) \times (3/9) \times (3/9) \times (3/9) \times (9/14) = 0.0053$$

De forma análoga, obtemos o $Likelihood$ para $play = no$:

$$Likelihood(no) = (3/5) \times (1/5) \times (4/5) \times (3/5) \times (5/14) = 0.0206$$

Normalizando esses valores em probabilidades, temos:

$$P(yes) = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%$$

$$P(no) = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%$$

Portanto, para o novo dia com as características citadas, temos que a classificação seria $play = no$. Esse método é baseado na regra de Bayes de probabilidade condicional. Essa probabilidade, de uma hipótese H e uma evidência E, é dada pela Equação 2.2 abaixo:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{2.2}$$

Para o *Naive Bayes*, porém, o denominador da equação pode ser desconsiderado, uma vez que, depois da normalização, o valor de $P(E)$ não vai influenciar na probabilidade de $P(H|E)$ para variados H. Considerando, também, que H seja a classe y; e a(s) evidência(s) E sejam os vetores de atributos $x_1, x_2, ..., x_n$, obtemos a Equação 2.3:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.3}$$

Dentre as variações de implementação do NB, foram utilizados neste trabalho o *Gaussian Naive Bayes* (G-NB) e o *Bernoulli Naive Bayes* (B-NB). As variações, segundo Pedregosa *et al.* (2011), se diferenciam na maneira que a distribuição de $P(x_i|y)$ é considerada. Como o próprio nome diz, o G-NB considera a distribuição $P(x_i|y)$ como sendo Gaussiana.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(- \frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right) \tag{2.4}$$

Em que, na Equação 2.4, μ_y é a média e σ_y é o desvio padrão da variável aleatória x_i para a classe y.

Já no B-NB, cada valor do vetor de atributos é considerado como um valor binário (booleano). Quando a entrada não é binária, o algoritmo "binariza" o vetor de atributos. A Equação 2.5 mostra como o B-NB considera a distribuição de $P(x_i|y)$ matematicamente. Nesse algoritmo de NB, a não ocorrência do atributo i é penalizada (PEDREGOSA *et al.*, 2011).

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i) \tag{2.5}$$

2.1.3 *Decision Trees*

Outra abordagem comum para aprendizado supervisionado são as árvores de decisão ou *Decision Trees*. Segundo Witten *et al.* (2016, p. 99), para construir uma árvore de decisão, de uma forma geral, primeiramente, um atributo é selecionado para ser o nó raiz. Para este nó, são criadas ramificações para cada valor possível. Esse procedimento é repetido, recursivamente, para cada ramificação. Um outro atributo é escolhido para ser o nó raiz da subárvore e este é ramificado para cada valor possível. Esse processo de ramificação (*split*) é feito sob regras (*splitting rules*), a fim de se reduzir impurezas (heterogeneidade) do nó (MOISEN, 2008). Chega-se a um nó folha (predição) quando não é mais possível ramificar o nó ou quando um critério de parada (definido externamente) é alcançado.

Existem vários algoritmos que implementam árvores de decisão, porém o utilizado neste trabalho foi o CART (*Classification And Regression Trees*) proposto por Breiman *et al.* (1984). No CART, a árvore construída é binária, ou seja, um nó possui até, no máximo, dois filhos. Para classificação, os *splitting rules* utilizam quatro métricas, segundo Moisen (2008):

- **Erro de classificação (*Misclassification error*):** proporção de observações no nó que não pertence à maioria das classes naquele nó;
- **Índice de Gini (*Gini index*):** é a probabilidade de um nó ser classificado incorretamente se uma classe for escolhida aleatoriamente dentro deste nó. Isso é calculado pela Equação 2.6:

$$\sum_{k=1}^{K} p_m(1 - p_m) \tag{2.6}$$

  Em que K é o total de classes e p_m é a proporção da classe no nó m. O índice de Gini buscado é 0, uma vez que indica que, no nó, se encontra apenas uma classe;
- **Índice de entropia (*Entropy index*):** ou desvio de impureza, indica o quão impuro, heterogêneo está um nó e é calculado pela fórmula 2.7 abaixo:

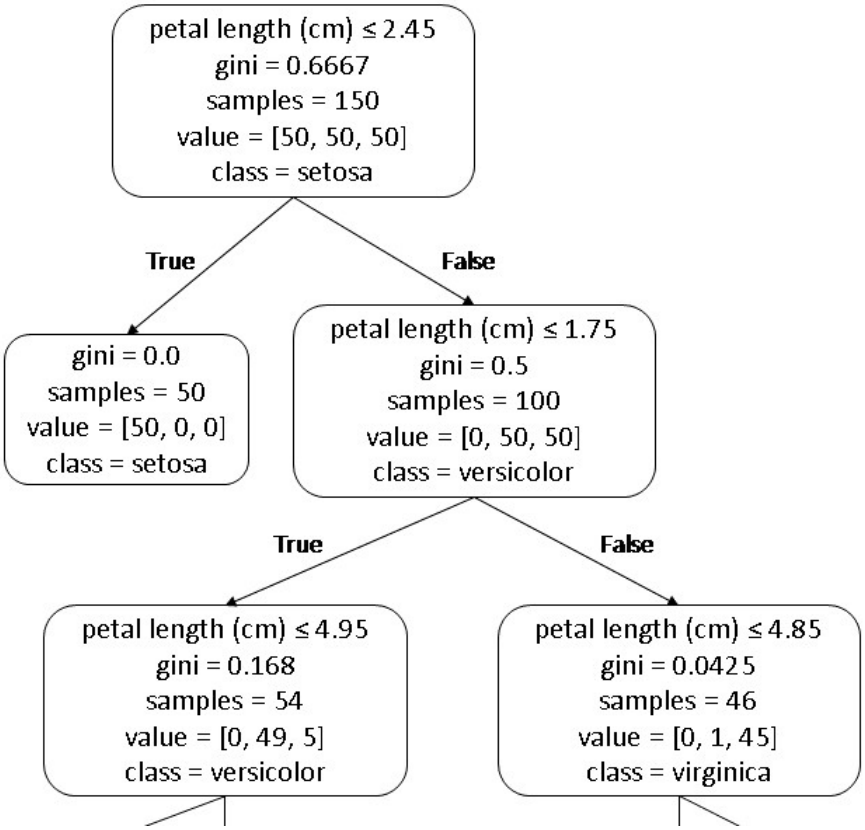
$$\sum_{k=1}^{K} p_m(\log p_m) \tag{2.7}$$

- **"Dobra" (*Twoing*):** trata cada divisão multiclasse como problema binário, podendo revelar semelhanças entre as classes.

Para exemplificar, considere o problema de classificação do *Iris flower dataset*[1], em que temos as classes *Iris Setosa*, *Iris Versicolour* e *Iris Virginica*; e quatro atributos:

[1] <https://archive.ics.uci.edu/ml/datasets/iris>

Figura 3 – Parte de uma árvore de decisão para classificar *Iris flower dataset*



Fonte: (PEDREGOSA *et al.*, 2011)

- comprimento da sépala (*sepal length*) em cm;
- largura da sépala (*sepal width*) em cm;
- comprimento da pétala (*petal length*) em cm; e
- largura da pétala (*petal width*) em cm.

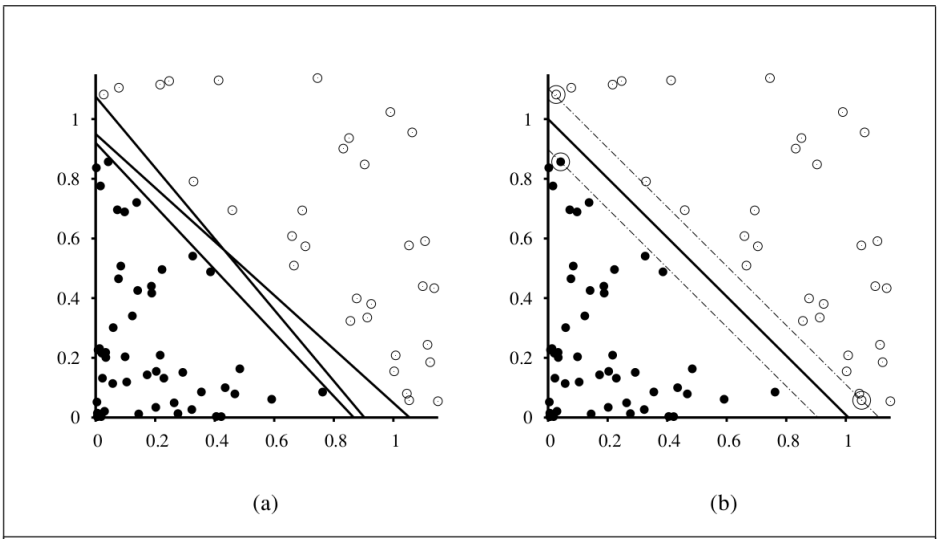
Na Figura 3, o nó raiz possui todo o conjunto de dados, com o valor $gini = 0.6667$. Na separação, $petallength(cm) \leq 2.45$, todos os exemplos que possuem comprimento de pétala menor ou igual a 2.45, vão para a ramificação $True$; e o restante para a ramificação $False$. No nó da ramificação $True$, temos $gini = 0.0$, que significa que $class = setosa$ é a classe predita para todos os dados em que $petallength(cm) \leq 2.45$. Já na ramificação $False$, o seu nó possui $gini = 0.5$. Isso significa que qualquer exemplo dentro desse nó escolhido aleatoriamente terá 50% de chance de ter $class = versicolor$. Uma vez que $gini \neq 0$, o processo de ramificação continua apenas nesse nó. Outro limiar de separação foi utilizado $petalwidth(cm) \leq 1.75$, e o processo se repete até encontrar, em todas as ramificações, nós folhas (com $gini = 0.0$).

2.1.4 *Support Vector Machines*

Support Vector Machine (SVM) é uma técnica de AM supervisionada de paradigma estatístico. É uma das técnicas mais utilizadas por apresentar resultados tão bons quanto, ou até superiores que, os resultados obtidos utilizando redes neurais artificiais (BRAGA; CARVALHO; LUDERMIR; HAYKIN, 2000, 2010 apud LORENA; CARVALHO, 2007, p. 43).

O SVM utiliza a função *kernel* para mapear os dados de acordo com seus atributos para um espaço altamente dimensional para poder classificá-los. Para isso, o SVM transforma os dados e procura uma função que separe os dados em suas categorias e possa ser desenhado como um hiperplano. Dessa forma, o SVM consegue classificar novos dados de acordo com seus atributos.

Figura 4 – Exemplo de classificação binário com SVM



Fonte: (RUSSELL; NORVIG, 2016, p. 745)

Na Figura 4, por exemplo, os dados (pertencentes às classes círculo branco ou preto) já foram transformados e mapeados. Em (a) há três candidatas a função separadora. Já em (b), observa-se a função separadora que possui a margem separadora máxima. Ou seja, a linha separa as categorias e mantém a maior distância entre ambas. Ainda em (b), observa-se os *support vectors* (os círculos destacados) que são os dados mais próximos da linha separadora.

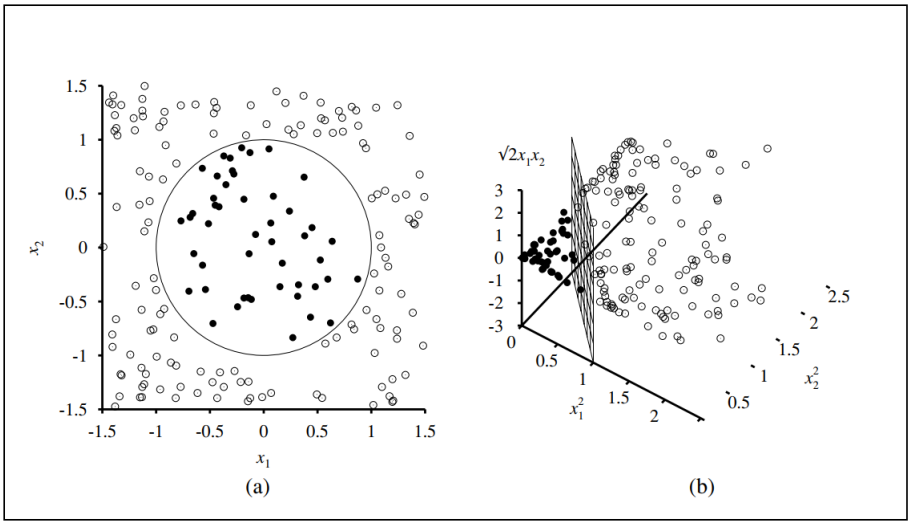
Tal SVM, que encontra um hiperplano ($w \cdot x + b = 0$) é chamado de SVM linear. De acordo com Lorena e Carvalho (2007), considere T um conjunto de treinamento com n dados $x_i \in X$ e seus rótulos $y_i \in Y = -1, +1$. T é linearmente separável se há um hiperplano que separe os dados de X nas classes -1 e $+1$. O hiperplano ótimo pode ser encontrado por meio da

seguinte equação:

$$\underset{\alpha}{\operatorname{argmax}} \sum_{j} \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (x_j \cdot x_k) \tag{2.8}$$

Em que $\alpha_j \geq 0$ são os dados dos vetores suporte; $\sum_j \alpha_j y_j = 0$ e $w = \sum_j \alpha_j x_j$. Todavia, nem sempre os dados são linearmente separáveis. Por exemplo, na Figura 5 (a), temos um conjunto de dados que foram mapeados em um espaço 2D. Nele é impossível encontrar um hiperplano que os separe.

Figura 5 – Exemplo de classificação SVM não-linear



Fonte: (RUSSELL; NORVIG, 2016, p. 747)

Mas se esses mesmos dados forem mapeados para um espaço tridimensional dados por x_i^2, x_2^2 e $\sqrt{2}x_1x_2$, como mostra a Figura 5 (b) é possível encontrar um hiperplano que separe os dados. Para encontrar esse mapeamento para um novo espaço, utiliza-se uma função chamada de função *kernel*.

A função *kernel* que realiza o mapeamento simples para o espaço bidimensional é chamado de *kernel* linear dada por $K(x_j, x_k) = (x_j, x_k)$. Este é exemplificado pela Figura 4. Já uma outra possível função *kernel* é a Gaussiana ou *Radial-Basis Function* (RBF) dada por $K(x_j, x_k) = exp(-\gamma \left\| x_j - x_k \right\|^2)$. Em que $-\gamma$ é o coeficiente de *kernel*. Exemplos de outras funções usadas como *kernel* são a polinomial e a sigmoidal.

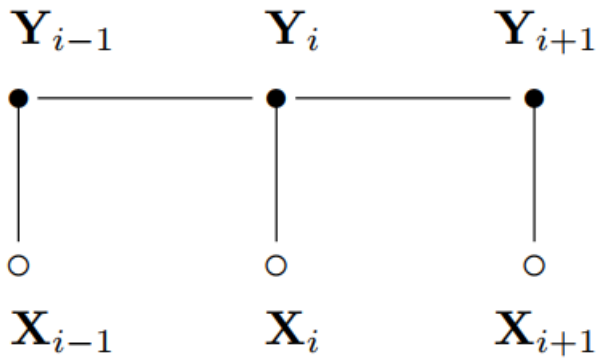
2.1.5 *Conditional Random Fields*

Conditional Random Fields (CRF) é um modelo estatístico proposto por Lafferty, McCallum e Pereira (2001) para a predição de sequências, cuja implementação para AM supervisionada tem sido bem utilizada devido ao seu desempenho. Lafferty, McCallum e Pereira (2001) definem CRF como um grafo, conforme a Definição 1.

Definição 1. *Seja $G = (V, E)$ um grafo em que $Y = (Y_v)_{v \in V}$ é indexado pelos vértices de G. Então, (X, Y) é um CRF, quando condicionado a X, a variável aleatória Y_v obedece a propriedade de Markov no grafo: $p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$, em que $w \sim v$ significa que w é vizinho de v no grafo G.*

Nesse grafo, cada vértice Y é o rótulo (a classe) e X são os seus atributos. Quando um vértice em Y é condicionado a X, a probabilidade de Y_v (rótulo seguinte) depende apenas do anterior – que é a propriedade de Markov.

Figura 6 – Representação gráfica do CRF



Fonte: (LAFFERTY; MCCALLUM; PEREIRA, 2001)

Na Figura 6, o condicionamento de Y por X é representado por uma ligação. Além disso, a propriedade de Markov é observada pela ligação de um Y_{i+1} apenas com Y_i – e não com Y_{i-1} também –, por exemplo. Resumindo, a probabilidade de Y_{i+1} é dependente apenas de Y_i, que por sua vez é dependente de X_i, e não é dependente de Y_{i-1}.

Segundo Andreani (2017), é preciso maximizar essas probabilidades. Isso deve ser feito, durante o treinamento, a fim de que a probabilidade de cada rótulo seja parametrizada por λ. Este "controla o *tradeoff* entre o ajuste dos pesos aos dados de treinamento e a regularização (uso de parâmetros pequenos) do modelo" (ANDREANI, 2017). Nessa etapa, o grafo CRF é construído com as transições de cada exemplo do conjunto de dados. Para maximizar tal probabilidade,

usa-se a função argmax, descrita na Equação 2.9. Nela, o valor de λ é calculado por meio de um algoritmo de estimação de parâmetros.

$$\underset{\lambda}{\mathrm{argmax}} \prod_{d=0}^{N} \frac{1}{Z(x_d)} exp\Big(\sum_{i=0}^{n} (\lambda_i f(y_{d,i}, y_{d,i-1})) \Big) \tag{2.9}$$

2.1.6 Validação Cruzada

Para treino e teste dos algoritmos descritos anteriormente é necessário empregar uma técnica de amostragem dos dados para que as estimativas de desempenho tenham validade. Uma técnica tradicionalmente usada é chamada de *hold-out*, na qual se divide os exemplos de treino e teste com base em uma taxa fixa de divisão, por exemplo, 33% do conjunto de dados para teste e o restante para treino. Essa divisão fixa nem sempre pode resultar em boas estimativas, uma vez que a porção reservada para treino pode não ser representativa (WITTEN *et al.*, 2016, p. 152–153).

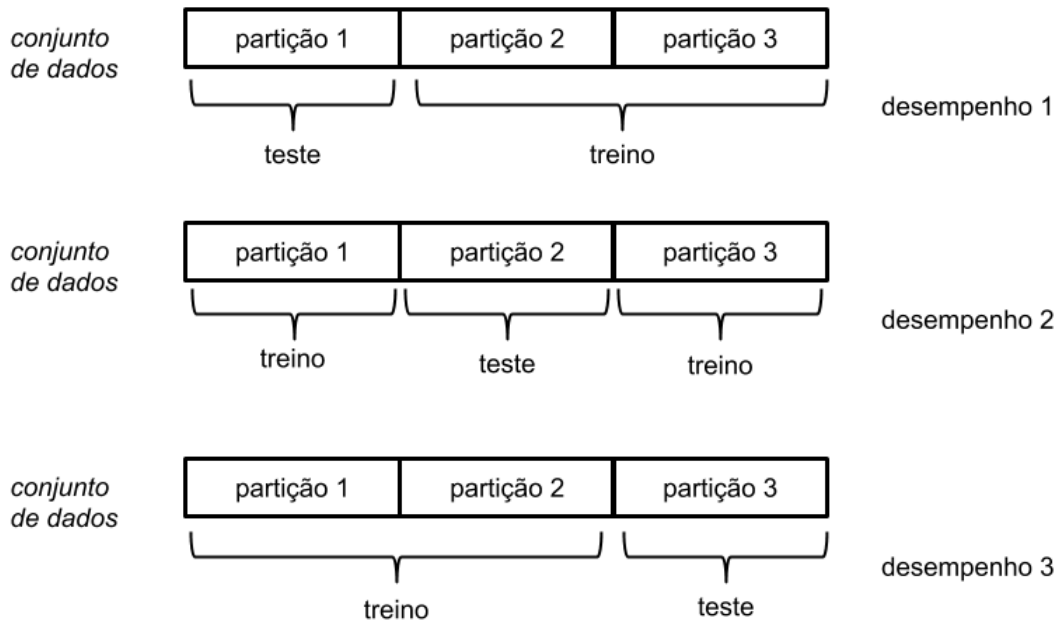
Outra técnica amplamente utilizada para estimar o desempenho de modelos de aprendizado é a validação cruzada (*cross-validation*). Essa técnica consiste em dividir o conjunto de dados em um determinado número de partições (ou *folds*). Para exemplificar o caso geral de 33% para teste, suponha uma validação cruzada de três partições (chamada de 3-*fold cross-validation*). Como mostra a Figura 7, o conjunto de dados é, então, dividido igualmente (ou aproximadamente) em três partições e três rodadas de treino e teste são realizadas. Em cada uma delas, uma das partições é utilizada para teste e o restante para treino. Ao final da validação, é feita a média dos desempenhos obtidos nas três rodadas realizadas e esse valor é usado como estimativa de desempenho do modelo.

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

O Teste de Turing foi fundamentado na linguagem humana. Um dos objetivos dos estudos para capacitar os computadores em Processamento de Linguagem Natural (PLN) é adquirir conhecimento a partir da linguagem escrita (RUSSELL; NORVIG, 2016, p. 860).

Uma linguagem é formada por um conjunto de palavras (léxico) que podem ser combinadas de acordo com as regras da gramática para formar sentenças (sintaxe). Além disso, uma sentença deve ter algum significado (semântica). As linguagens naturais são mais complexas que isso, uma vez que são ambíguas, grandes e estão em constante mudança. Por isso, são utilizados

Figura 7 – Etapas da validação cruzada de três partições



Fonte: autoria própria

modelos de linguagens naturais que são uma aproximação da linguagem natural (RUSSELL; NORVIG, 2016, p. 860–861). O modelo mais conhecido é o modelo n-grama. Tal modelo é um modelo probabilístico que permite a predição do próximo elemento (no caso, palavra) dentro de uma sequência de n palavras. Quando o $n = 1$, chamamos os elementos de unigrama, de bigrama para $n = 2$, trigrama para $n = 3$, e assim sucessivamente.

A tarefa de PLN a ser destacada neste trabalho é a de classificação (ou categorização) de sentenças de um texto. Dadas as sentenças de um texto, deve-se predizer a qual classe essas sentenças pertencem. Especificamente, se busca reconhecer a estrutura retórica do texto. As estruturas retóricas organizam os elementos de um discurso em termos das suas funções retóricas.

No contexto de AM, o texto (ou sentenças) a ser classificado pode ser classificado como um vetor de atributos (*features*). Esses atributos podem ser superficiais ou profundos. Entende-se por atributos superficiais aqueles que podem ser observados a partir da superfície do texto. Exemplos de atributos extraídos superficialmente são n-gramas e TF-IDF. Atributos profundos seriam aqueles que representam conhecimento para além da superfícies (as próprias palavras), como, por exemplo, a voz do verbo principal ou a presença de expressões formulaicas.

Quando o texto é representado usando as palavras, ou unigramas, como características, o modelo de representação é chamado de *bag of words*. Nele, o vocabulário (todas as palavras presentes no *corpus*, ou conjunto de textos) é representado em uma matriz. Essa matriz contém a

ocorrência de cada palavra do vocabulário em cada unidade de texto a ser classificada. Portanto, tal matriz possui dimensão proporcional ao tamanho do vocabulário e é esparsa. Para bigramas, a frequência da sequência de duas palavras é representada na matriz, dessa forma, o tamanho da matriz se eleva ao quadrado. De forma análoga, para trigrama, o tamanho da matriz é cúbica, e assim por diante para os demais valores de n.

Os *bag of words* são uma representação superficial e não levam em consideração a sequência das palavras (aumentar o valor de n dos n-gramas pode representar isso, mas quanto maior o tamanho da matriz gerada, mais incomputável o problema se torna) e não capturam a semântica do texto. Por isso, surgiram estudos de técnicas e de modelos de *word embeddings* que serão descritos na Seção 3.

2.2.1 TF-IDF

TF-IDF (*Term Frequency - Inverse Document Frequency*) é uma medida de frequência das palavras de um documento – no caso deste trabalho, de uma sentença –, em que as palavras mais frequentes de um documento que não aparecem em todos os documentos são tidas como caracterizantes do documento (WITTEN *et al.*, 2016, p. 329). Portanto, as palavras que são frequentes e "exclusivas" de uma sentença tendem a ter maiores valores TF-IDF.

$$TFIDF = f_{\text{ij}} \times \log \frac{numero\,de\,documentos}{numero\,de\,documentos\,que\,possuem\,a\,palavra\,i} \tag{2.10}$$

Na Equação 2.10, f_{ij} é a frequência da palavra i no documento j. Essa equação mostra como o TF-IDF de uma palavra é calculado.

Para se obter um vetor de atributos, gera-se uma matriz em que as colunas representam cada palavra do conjunto de sentenças; e, as linhas, cada sentença. Além de uma única palavra (unigrama), uma coluna pode representar também uma sequência de duas palavras (bigrama), ou de três palavras (trigrama), e assim sucessivamente. Isso gera uma matriz bem esparsa e de dimensões proporcionais ao tamanho do vocabulário (conjunto de palavras do conjunto de sentenças). Para um vocabulário de tamanho x, a matriz de TF-IDF terá x colunas para unigrama; x^2, para bigramas; x^3, para trigramas; e assim por diante. É interessante, então, diminuir o tamanho dessa matriz. Para isso, pode-se selecionar os k melhores atributos (ou palavras) utilizando, por exemplo, o teste de qui-quadrado (*chi-squared*, ou χ^2).

2.2.2 Teste qui-quadrado

O Teste qui-quadrado (χ^2) é uma estatística que mede a dependência entre variáveis estocásticas (PEDREGOSA *et al.*, 2011). Em outras palavras, o teste verifica quais são os atributos mais relevantes entre os recursos não negativos, como booleanos ou frequências (por exemplo, valores TF-IDF), relativos às classes. A intuição por trás do uso do teste para a seleção de atributos é a de que quanto mais o atributo e as classes forem dependentes, mais relevante ele é para a previsão. Assim, a partir de um ranque dos atributos pelo valor estatística χ^2, é possível selecionar os K melhores atributos.

A estatística χ^2 é calculada por meio da Equação 2.11, na qual e_i é a frequência esperada da classe i ocorrer e o_i é a frequência observada da classe i.

$$\chi^2 = \sum_i \frac{(o_i - e_i)^2}{e_i} \tag{2.11}$$

Considere, por exemplo, o conjunto de dados apresentado na Tabela 3. Nela, temos três exemplos classificados em A ou B, de acordo com três atributos booleanos.

Tabela 3 – Exemplo de conjunto de dados

Exemplo	Classe	Atributo 1	Atributo 2	Atributo 3
1	A	Falso	Verdadeiro	Verdadeiro
2	B	Verdadeiro	Falso	Verdadeiro
3	A	Verdadeiro	Verdadeiro	Falso

Fonte: autoria própria

A partir dos dados da Tabela 3, temos as frequências observadas para os valores do Atributo 1 e as classes A e B, conforme mostrado na Tabela 4. As frequências observadas para o Atributo 3 e as classes A e B são idênticas as do Atributo 1, por isso elas também são representadas pela Tabela 4.

Tabela 4 – Frequência observada dos atributos 1 e 3 para cada classe

OBS	Falso	Verdadeiro	Total
A	1	1	2
B	0	1	1
Total	1	2	3

Fonte: autoria própria

Ainda a partir da Tabela 3, temos a frequência observada de cada valor possível do Atributo 2 para as classes A e B, conforme mostrado na Tabela 5.

Tabela 5 – Frequência observada do atributo 2 para cada classe

OBS	Falso	Verdadeiro	Total
A	0	2	2
B	1	0	1
Total	1	2	3

Fonte: autoria própria

A partir das tabelas de frequências observadas, podemos calcular as frequências esperadas. Dado que N é o número total de observações (neste caso, $N = 3$), a frequência esperada e_{ij} para uma classe i e um valor de atributo j é dada por $e_{ij} = (Total_i * Total_j)/N$. As frequências esperadas para as classes A e B e os valores possíveis de atributos são mostradas na Tabela 6.

Tabela 6 – Frequência esperada das classes dados os três exemplos

ESP	Falso	Verdadeiro	Total
A	0,667	1,334	2
B	0,333	0,666	1
Total	1	2	3

Fonte: autoria própria

Aplicando os valores das Tabelas 4, 5 e 6 na Equação 2.11, obtemos os seguintes valores de χ^2 para os atributos 1, 2 e 3, respectivamente: $0,75$, $3,0$ e $0,75$. Como o valor da estatística é maior para o Atributo 2, podemos inferir que ele é mais informativo a respeito das classes do que os atributos 1 e 3.

2.2.3 AZPort

O AZPort (*Argumentative Zone for Portuguese*) é um sistema de detecção automática de estrutura retórica de resumos acadêmicos em português (FELTRIM, 2004). Tal sistema utiliza a técnica de *Argumentative Zoning* (AZ) proposta por Teufel e Moens (2002), mas aplicada a textos na língua portuguesa. Essa técnica segmenta um artigo científico em zonas argumentativas ("categorias") de forma automática.

Assim como o AZ, o AZPort é um classificador bayesiano, de modo que foi necessário utilizar um conjunto de exemplos para o treinamento do modelo. O *corpus* utilizado para esse fim

foi chamado de CorpusDT (ou Corpus 366, conforme descrito na Seção 4.1). O classificador é baseado em um conjunto de oito atributos derivados dos atributos propostos por Teufel e Moens (2002), conforme descritos na Tabela 7.

Tabela 7 – Resumo do conjunto de atributos

Atributo	Valores possíveis
Tamanho	curto, média ou longa
Localização	primeira, segunda, mediana, penúltima ou última
Citação	sim ou não
Expressão	B, C, G, M, P, R ou *no expr*
Tempo	IMP, PRES, PAST, FUT, PRES-CPO, PAST-CPO, FUT-CPO, PRES-CT, PAST-CT, FUT-CT, PRES-CPO-CT, PAST-CPO-CT, FUT-CPO-CT ou *no verb*
Voz	passiva, ativa ou *no verb*
Modal	sim, não ou *no verb*
Histórico	_, B, C, G, M, O, P ou R

Fonte: adaptado de (FELTRIM, 2004)

O atributo Tamanho utiliza os delimitadores 20 e 40 palavras para determinar se uma sentença é curta, média ou longa. O atributo Expressão identifica a presença ou não de expressões formulaicas[2] na sentença. Se houver, o valor é a classe da expressão encontrada. Já os atributos sintáticos Tempo, Voz e Modal são determinados com base no primeiro verbo ou primeira locução verbal da sentença (estando no modo indicativo ou imperativo). O atributo Tempo indica a flexão verbal, a Voz indica a voz do verbo (ativa ou passiva) e o Modal indica se há um auxiliar modal no complexo verbal – como, por exemplo, ter, que, de, $dever$, $precisar$ e $poder$. Caso não haja verbos na sentença, o valor *no verb* é atribuído. Por fim, o atributo Histórico indica qual a classe da sentença anterior. Caso seja a primeira, o valor "_" é atribuído.

De acordo com Feltrim (2004), o resultado (Medida F1, definido na Seção 4.2) obtido com o AZPort sobre o CorpusDT foi de 72%. Esse valor é bem melhor do que os obtidos neste trabalho – como é possível observar na Seção 5 e/ou no Apêndice A –, uma vez que o algoritmo bayesiano implementado no AZPort se aproxima mais do *Naive Bayes* tradicional, que considera atributos categóricos sem binarização e estima as probabilidades a priori diretamente do *corpus* de treinamento (sem o uso de uma função de densidade de probabilidade). Este, por sua vez, se diferencia das implementações *Gaussian Naive Bayes* e *Bernoulli Naive Bayes* descritas na Seção 2.1.2.

[2] Expressões mais ou menos padronizadas que caracterizam uma classe de sentenças, por exemplo: "Conclui-se que..." é uma expressão formulaica para a classe Conclusão.

3 *WORD EMBEDDINGS*

Word Embeddings (WE) é uma representação distribuída de língua, em que uma palavra é um vetor (ou *embedding*) criado de acordo com seu valor de distribuição em um *corpus*. Em geral, esses vetores possuem dimensões menores que aqueles usados pelo modelo de *bag of words*, por exemplo. Essa forma de representação melhora o desempenho nas tarefas de PLN, uma vez que, diferentemente de *bag of words* – que é, simplesmente, a contagem da quantidade de vezes que uma palavra aparece –, os WE tratam as palavras de acordo com sua similaridade e dissimilaridade (MIKOLOV *et al.*; LEVY; GOLDBERG, 2013a, 2014 apud SOUSA, 2016, p. 21).

Para isso, tal método aprende uma representação para cada palavra (*embeddings*) e as mapeia para vetores em um espaço semântico de modo que palavras semanticamente relacionadas apresentam uma relação espacial consistente, permitindo inferências sobre os seus significados. Observe o exemplo a seguir retirado de Mikolov *et al.* (2013b):

$$vec(\text{“}Madrid\text{”}) - vec(\text{“}Spain\text{”}) + vec(\text{“}France\text{”}) = vec(\text{“}Paris\text{”})$$

Na equação acima, foram calculadas representações para os países e suas capitais. Como Madri (“Madrid”) está para Espanha (“Spain”), então possuem *embeddings* semelhantes. Dessa forma, ao realizar o lado esquerdo da equação e sobrar o valor de França (“France”), encontra-se Paris. Em outras palavras, essa equação mostra que Madri está para Espanha, assim como Paris está para França.

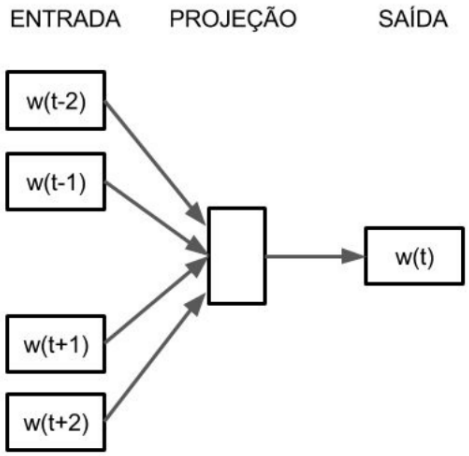
WE podem ser aprendidos a partir de um *corpus* de forma não supervisionada usando redes neurais. Duas arquiteturas conhecidas para o treinamento de WEs são *Continuous bag of words* e *Skip-gram*.

3.1 *CONTINUOUS BAG OF WORDS*

Continuous bag of words (CBOW) é um modelo semelhante ao *bag of words*. A diferença é que o CBOW usa uma representação contínua e distribuída do contexto (MIKOLOV *et al.*, 2013a, p. 4). Tal modelo busca prever uma próxima palavra a partir de um contexto. Como mostra a Figura 8, as camadas são semelhantes às camadas de redes neurais. A predição de uma palavra de entrada $w(t)$ dada por essa arquitetura é feita a partir de uma quantidade de palavras ao redor de $w(t)$, ou seja, a partir de seu contexto $w(t-2)$, $w(t-1)$, $w(t+1)$ e $w(t+2)$. Para isso, na

camada de projeção, é feita a soma dos *embeddings* das palavras na janela de contexto – camada de saída.

Figura 8 – Modelo CBOW



Fonte: adaptado de (MIKOLOV *et al.*, 2013a)

Na Equação 3.1, temos a probabilidade da palavra $w(t)$ (saída) ocorrer no contexto h (SOUSA, 2016, p. 29):

$$P(w(t)|h) = \frac{e^{score(w(t),h)}}{\sum_{w(i)\in W} e^{score(w(i),h)}} \tag{3.1}$$

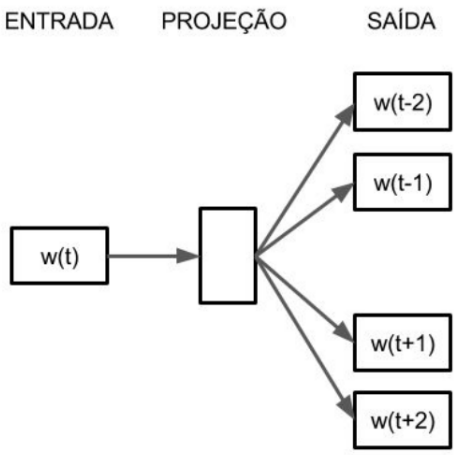
Em que $w(t)$ e h são, respectivamente, o *embedding* da palavra t e os *embeddings* do contexto h. Se o tamanho do contexto é maior que um, h é a soma dos *embeddings* do contexto de $w(t)$. Além disso, W é o conjunto de palavras pertencentes ao vocabulário e $score$ verifica a semelhança entre dois vetores (no caso, entre a os *embeddings* da palavra $w(t)$ ou $w(i)$ e seu contexto).

3.2 *SKIP-GRAM*

O modelo *Skip-gram* realiza o processo inverso do modelo CBOW, ou seja, prediz o contexto a partir de uma palavra. Segundo Mikolov *et al.* (2013a, p. 4), essa arquitetura tenta maximizar a predição de uma palavra baseada em outra que se encontra na mesma sentença. Esse modelo pode ser observado na Figura 9.

O *skip-gram* se utiliza da mesma equação que o CBOW (Equação 3.1). A diferença está na forma que as probabilidades são usadas. A probabilidade $P(w(t)|h)$ é usada para encontrar as

Figura 9 – Modelo *Skip-gram*



Fonte: adaptado de (MIKOLOV *et al.*, 2013a)

probabilidades das palavras dentro do contexto de $w(t)$.

3.3 WORD2VEC VS. WANG2VEC

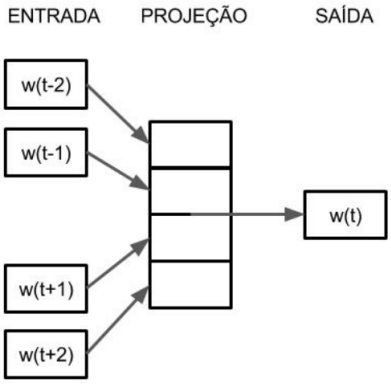
Word2Vec é uma ferramenta proposta por Mikolov *et al.* (2013a) para a geração de *embeddings* utilizando os modelos CBOW e *skip-gram*. Esse modelo captura apenas a semântica das palavras de um texto, já que desconsidera a ordem das palavras (LING *et al.*, 2015, p. 1). Como poder ser visto nas Figuras 8 e 9, não importa a ordem das palavras na janela de contexto que entra ou que sai da camada de projeção.

Por sua vez, o Wang2Vec é uma ferramenta, um Word2Vec incrementado, proposta por Ling *et al.* (2015). No Wang2Vec, os modelos CBOW e *skip-gram* sofrem alterações a fim de gerar *embeddings* somados com informações sintáticas. Para isso, foram feitas duas alterações, uma no modelo CBOW e outra no modelo *skip-gram*.

A primeira alteração, o modelo *CWindow* proposto por Ling *et al.* (2015, p. 3), é o modelo CBOW com a camada de projeção modificada. Ao invés de realizar, simplesmente, a soma dos *embeddings* das palavras do contexto, realiza-se a concatenação dos *embeddings* das palavras, guardando assim a ordem das palavras, e é feita a soma de seus *embeddings* relativos às suas posições. O esquema desse modelo pode ser observado na Figura 10.

A segunda alteração, o modelo *skip-gram* estruturado, mantém o esquema básico apresentado na Figura 9. A diferença está em seu processamento, em que a predição da janela de contexto é feita para a posição relativa à palavra corrente $w(t)$ (LING *et al.*, 2015, p. 2–3).

Figura 10 – Modelo *CWindow*



Fonte: adaptado de (LING *et al.*, 2015)

3.4 GLOVE

Global Vectors (GloVe) é um outro modelo para representação de palavras proposto por Pennington, Socher e Manning (2014). Nele, os *embeddings* são gerados a partir da matriz de co-ocorrência. Tal matriz X, possui como elementos a contagem de uma palavra em relação a outra. Ou seja, o elemento $x(i, j)$ é o número de vezes que a palavra j aparece no contexto de i. Dessa forma, a probabilidade da palavra j aparecer no contexto de i é dada pela Equação 3.2.

$$P(j|i) = \frac{x(i,j)}{x(i)} \tag{3.2}$$

Na Equação 3.2, $x(i)$ é a ocorrência de qualquer palavra no contexto de i e é calculada como mostra a Equação 3.3:

$$x(i) = \sum_{k} x(i,k) \tag{3.3}$$

4 MATERIAIS E MÉTODOS

Os classificadores construídos neste trabalho utilizam algoritmos conhecidos de aprendizado supervisionado, a saber: SVM, K-NN, *Naive Bayes*, *Decision Trees* e CRF. Os atributos utilizados variaram entre superficiais e semânticos. Dentre eles estão: *n-gramas*, TF-IDF, posição da palavra, *word embeddings* e os atributos extraídos do AZPort. Além disso, toda a etapa de treinamento e teste dos algoritmos foi feita usando validação cruzada de 10 partições.

4.1 *CORPORA*

Os três *corpora* utilizados se encontram na língua portuguesa do Brasil e são constituídos de resumos de teses e dissertações, da área de Ciência da Computação. Os resumos foram coletados e anotados como parte dos trabalhos de Feltrim (2004) e Andreani (2017). A Tabela 8 apresenta informações referentes aos três *corpora* usados.

Tabela 8 – Quantidade de sentenças de cada *corpus*

Rótulos	**Corpus 366**	**Corpus 466**	**Corpus 832**
B	77	179	256
C	20	20	40
G	36	36	72
M	45	59	104
O	6	1	7
P	65	68	133
R	117	103	220
Total sentenças	**366**	**466**	**832**

Fonte: adaptado de (ANDREANI, 2017)

Todos os *corpora* estão anotados com as seguintes classes: B, C, G, M, O, P e R, que representam as categorias retóricas a saber: contexto (B), conclusão (C), lacuna (G), metodologia (M), estrutura (O), propósito (P) e resultado (R). O *corpus* 366 (também chamado de CorpusDT) é composto por 52 resumos, totalizando 366 sentenças, e corresponde ao *corpus* utilizado no trabalho de Feltrim (2004). O *corpus* 466 corresponde ao *corpus* usado no trabalho de Andreani (2017), que também possui 52 resumos que totalizam 466 sentenças. Por fim, o *corpus* 832 corresponde a junção dos *corpora* 366 e 466. Consequentemente, possui 104 resumos, totalizando 832 sentenças. Observando a Tabela 8, é possível notar que, embora as frequências das categorias sejam distintas nos três *corpora*, os *corpora* têm uma distribuição similar de categorias em termos

proporcionais.

4.2 MÉTRICAS

Para medir o desempenho dos classificadores, foram utilizadas as medidas precisão ($precision$), revocação ($recall$) e medida F1 (*f1-score* ou *f-measure*). Tais medidas são estimadas usando-se os conceitos de verdadeiros positivos (TP – *True Positives*), falsos positivos (FP – *False Positives*) e falsos negativos (FN – *False Negatives*).

Tomando como referência uma classe X, verdadeiros positivos (TP) são as instâncias que foram selecionadas e que fazem parte dos elementos relevantes, em outras palavras, quando uma instância é predita como pertencente a uma classe X e ela de fato é da classe X. Falsos negativos (FN) são as instâncias que não foram selecionadas, mas que fazem parte dos elementos relevantes, ou seja, quando uma instância da classe X é predita como sendo de uma classe diferente de X. Falsos positivos (FP) são as instâncias que não pertencem aos elementos relevantes, mas que foram selecionados, em outras palavras, FP são as instâncias que foram classificadas como X, mas que não pertencem a X.

Tendo em conta as definições citadas, as medidas de precisão, revocação e F1 podem ser enunciadas de acordo com as seguintes equações:

- **Precisão (*precision*):** é a proporção de acertos dentro do total de predições (Equação 4.1).

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

- **Revocação (*recall*):** é a proporção de acertos dentro da quantidade de dados relevantes (Equação 4.2).

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

- **Medida F1 (*f1-score*):** é uma medida de desempenho que considera tanto a precisão ($precision$) quanto revocação ($recall$) na sua estimativa, realizando a média harmônica entre essas duas medidas (Equação 4.3).

$$F1 = \frac{precision \times recall}{precision + recall} \tag{4.3}$$

4.3 MODELOS DE *WORD EMBEDDINGS*

Os modelos de *word embeddings* utilizados foram os modelos CBOW e *skip-gram* – ambos treinados com as ferramentas Word2Vec e Wang2Vec – e modelo GloVe. Todos os modelos

empregados foram treinados por Hartmann *et al.* (2017) e estão disponíveis no repositório NILC–*Embeddings*[1]. Nesse repositório, os modelos disponibilizados se constituem de vetores gerados a partir de um *corpus* em português brasileiro e europeu, coletado de fontes e gêneros variados. Cada modelo foi disponibilizado no repositório com vetores de dimensões iguais a 50, 100, 300, 600 e 1000.

A partir do modelos de *word embeddings* é possível consultar o vetor (*embedding*) para uma palavra, porém neste trabalho era necessário utilizar o *embedding* de uma sentença. Para isso, foram utilizadas duas estratégias de combinação de *embeddings* – soma e média. Assim, para obter o *embedding* de uma sentença foi feita a soma ou a média dos *word embeddings* das palavras que formavam a sentença.

4.4 COMBINAÇÕES DOS ATRIBUTOS

Além dos atributos advindos dos modelos de *word embeddings*, também foram avaliados atributos gerados por meio de TF-IDF e os extraídos pelo classificador AZPort. A extração dos valores de TF-IDF foi feita baseada em unigramas ($ngrama = 1$) e, posteriormente, para redução de dimensionalidade, foi feita a seleção dos $k = 100$ melhores atributos utilizando qui-quadrado. Os atributos do AZPort foram extraídos pelo mesmo *pipeline* de extração implementado por Feltrim (2004), de forma que foi necessária apenas a leitura dos arquivos de saída para sua utilização. A extração de características usando modelos de *word embeddings* explorou os cinco modelos descritos na Seção 4.3 com variadas dimensões.

Além desses conjuntos de atributos serem avaliados individualmente, foram feitas combinações entre eles. As combinações foram feitas através da concatenação dos vetores extraídos para cada classe de atributos. Foram avaliadas as seguintes combinações:

- TF-IDF + AZPort-*features*;
- *Word embeddings* + TF-IDF;
- *Word embeddings* + AZPort-*features*;
- *Word embeddings* + TF-IDF + AZPort-*features*.

[1] <http://www.nilc.icmc.usp.br/embeddings>

4.5 CLASSIFICADORES

Como já foi mencionado, os classificadores utilizados foram: K-NN, *Naive Bayes* com as variações Gaussiana e Bernoulli, *Decision Trees* (DT) com a implementação do algoritmo CART, SVM de *kernels* linear e RBF e CRF. Tais algoritmos foram importados da biblioteca Python para aprendizado de máquina *scikit-learn*[2].

Para o classificador K-NN, com exceção do parâmetro $n\_neighbors = 3$ (que determina o número de vizinhos a ser considerado), os parâmetros utilizados foram *default*. Tais parâmetros são: $weights = `uniform'$, $algorithm = `auto'$, $p = 2$, $metric = `minkowski'$, $metric\_params = None$ e $n\_jobs = 1$. O parâmetro $weights$ determina a função-peso (*weight function*) a ser utilizada. No caso, todos os pontos da vizinhança possuem os mesmos pesos. O segundo parâmetro, $algorithm$, determina o algoritmo a ser utilizado para calcular os vizinhos mais próximos. Com $algorithm = `auto'$, o classificador decidirá qual o melhor algoritmo a ser utilizado, baseado no conjunto de treino. Como as matrizes de atributos são esparsas, o algoritmo escolhido é o de força bruta (*brute force*). Os parâmetros p e $metric$ são considerados em conjunto. Com seus valores *default*, determinam o cálculo da distância Euclidiana. Por fim, n_jobs determina a quantidade de processos paralelos para calcular a distância dos vizinhos. No caso, apenas um processo foi utilizado.

Para os classificadores *Naive Bayes*, também foram utilizados parâmetros *default*. O *Gaussian NB* (G-NB) possui apenas um parâmetro $prior$. Esse parâmetro determina as probabilidades anteriores das classes. Com $prior = None$, essas probabilidades são ajustadas de acordo com os dados de treino. Já o *Bernoulli NB* (B-NB) possui os parâmetros $alpha = 1.0$, $binarize = 0.0$, $fit\_prior = True$ e $class\_prior = None$. O primeiro parâmetro determina que o classificador realize suavização aditiva. O parâmetro $binarize$ determina o valor de limiarização para valores binários dos atributos. Seguindo a mesma linha do parâmetro $prior$ do G-NB, o parâmetro $fit\_prior = True$ determina que o classificador utilize as probabilidades anteriores das classes. Por fim, o parâmetro $class\_prior$ realiza exatamente a mesma função que o $prior$ do G-NB.

Para o DT-CART, foram usados o parâmetro $random\_state = 0$ e os parâmetros restantes com valores *default*. São eles:

- $criterion = `gini'$;
- $splitter = `best'$;
- $max\_depth = None$;

[2] <http://scikit-learn.org>

- $min\_samples\_split = 2$;
- $min\_samples\_leaf = 1$;
- $min\_weight\_fraction\_leaf = 0.0$;
- $max\_features = None$;
- $max\_leaf\_nodes = None$;
- $min\_impurity\_decrease = 0.0$;
- $min\_impurity\_split = None$;
- $class\_weight = None$; e
- $presort = False$.

O primeiro parâmetro define a semente que será usada para gerar números aleatórios. Seguindo para os parâmetros com valores *default*, o primeiro define a função que irá medir a qualidade da fragmentação do nó. Como descrito na Seção 2.1.3, será usado o Índice de Gini (Seção 2.1.3). O parâmetro $splitter$ define a estratégia para a fragmentação do nó ('$best$' ao invés de '$random$'). Em seguida, o parâmetro max_depth define a profundidade máxima da árvore. Com valor $None$, a árvore se expande até que todas as folhas sejam puras ou até que todas as folhas tenham menos exemplos do que a quantidade determinada no parâmetro $min\_samples\_split$. O outro parâmetro, $min\_samples\_leaf$ determina a quantidade mínima de exemplos que um nó deve conter. Ou seja, um nó deve ter no mínimo um exemplo. O parâmetro seguinte define a fração ponderada mínima da soma total de pesos necessária para estar em um nó folha. O parâmetro $max\_features$ determina o número de atributos a considerar enquanto a melhor fragmentação é procurada. Quando igual a $None$, esse número é a quantidade de atributos fornecida ao classificador. O parâmetro $max\_leaf\_nodes = None$ define que a árvore cresça sem limitação da quantidade de nós folha. O parâmetro seguinte define se um nó será fragmentado. Isso ocorrerá se a fragmentação induzir a uma diminuição de impureza (Índice Gini) maior ou igual ao seu valor. Continuando, $min\_impurity\_split = None$ define um limiar para interrupção precoce do crescimento da árvore. No caso, não é utilizado. $class\_weight = None$ define que todas as classes tenham peso igual a um. Por fim, o parâmetro $presort = False$ define que os dados não sejam pré-ordenados no treinamento (a fim de acelerar a busca pela melhor fragmentação de um nó).

Para o classificador SVM, os valores do parâmetro $kernel =$ '$linear$' e $kernel =$ 'rbf' define o *kernel* do SVM como linear ou RBF, repectivamente. Outros dois parâmetros com valores não-*default* são: C, que define o parâmetro de penalidade do termo de erro, e $gamma$,

que define o coeficiente de *kernel* no caso de SVM-RBF. No SVM-linear, para os *corpora* 366 e 832, foi usado $C = 100$; e, para o *corpus* 466, $C = 1000$. Já no SVM-RBF, para todos os *corpora*, foram usados $C = 1000$ e $gamma = 0.001$. Os valores desses parâmetros foram escolhidos por meio de *grid search* para maximizar o desempenho com os atributos de TF-IDF. Os valores dos parâmetros restantes são *default*. São eles: $shrinking = True$, $probability = False$, $tol = 0.001$, $cache\_size = 200$, $class\_weight = None$, $verbose = False$, $max\_iter = -1$, $decision\_function\_shape =$ 'ovr' e $random\_state = None$. O primeiro define que a heurística de encolhimento seja usada. O segundo define a tolerância ao critério de parada. Em seguida, $cache\_size$ define o tamanho da *cache* do *kernel*. Da mesma forma que o DT, $class\_weight = None$ define que todas as classes tenham peso igual a um. A saída verbosa é desativada com $verbose = False$. O parâmetro $max\_iter = -1$ define que o número de iterações seja ilimitado. O parâmetro seguinte determina que uma função *one-vs-rest* ('ovr') de decisão na forma $(n\_sentences, n\_classes)$ seja retornada. Por fim, o último parâmetro define a semente do pseudo gerador de números aleatórios. Como $random\_state = None$, o gerador utilizado é o da biblioteca NumPy[3].

Por fim, o CRF utilizado foi o *sklearn-crfsuite*[4], uma versão do *CRFsuite*[5] que é compatível com os estimadores do *scikit-learn*. A implementação de um classificador CRF funciona diferente dos demais, uma vez que os atributos devem ser armazenados em um $dict$, além de ter marcações de início e fim de uma sequência, no caso deste trabalho, um resumo. Os atributos derivados do TF-IDF são vetores de dimensão 100 (seleção com o teste χ^2) e os derivados dos *word embeddings* são vetores que variam entre 50, 100, 300, 600 ou 1000.

Foram feitas duas implementações de classificadores CRF – CRF e CRF2. No CRF, há apenas uma posição do vetor para armazenar TF-IDF e outro para *word embeddings*. Como uma posição do $dict$ aceita apenas um valor, foi feita a soma do TF-IDF de uma sentença para armazenar seu TF-IDF. Isso vale para *word embeddings* também. Já no CRF2, cada posição do vetor de TF-IDF e de *word embeddings* tem uma posição no $dict$. Isso possibilita armazenar cada valor sem somá-los. Em ambas as implementações, é armazenada também a posição da sentença. Por fim, cada um dos oito atributos do AZPort é armazenado em posições diferentes no $dict$.

[3] <http://www.numpy.org/>
[4] <https://sklearn-crfsuite.readthedocs.io/en/latest/>
[5] <http://www.chokkan.org/software/crfsuite/>

5 RESULTADOS E ANÁLISE

Os resultados obtidos nos experimentos empregando as variações de atributos e algoritmos descritos na Seção 4 se encontram resumidos nas tabelas e nas figuras a seguir. Os resultados em termos de *f1-score* obtidos com cada combinação de características, cada *corpus*, cada modelo de WE, cada dimensão de vetor de WE e cada forma de representação de WE para uma sentença se encontram nas tabelas do Apêndice A. Nas tabelas do apêndice e na Tabela 9, RBF corresponde ao SVM com *kernel* RBF e SVM corresponde ao SVM de *kernel* linear.

A Tabela 9 mostra as melhores médias de *f1-score* obtidas com cada combinação de características para cada classificador avaliado. Observa-se em negrito a maior porcentagem alcançada por cada algoritmo.

Tabela 9 – Melhores resultados obtidos em cada combinação de atributos

Características	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
TF-IDF	46%	57%	29%	40%	51%	48%	63%	40%
WE	48%	48%	37%	42%	44%	33%	39%	55%
AZPort	58%	48%	59%	27%	40%	63%	71%	93%
TF-IDF + AZPort	63%	57%	55%	33%	52%	**64%**	76%	92%
WE + TF-IDF	54%	54%	40%	**50%**	**54%**	45%	56%	58%
WE + AZPort	**65%**	61%	**63%**	48%	46%	59%	72%	**94%**
Todos	**65%**	**64%**	58%	37%	**54%**	61%	**77%**	**94%**
Média	57%	56%	49%	40%	49%	53%	65%	75%
Desvio Padrão	8%	6%	13%	8%	5%	12%	14%	23%

Fonte: autoria própria

Conforme mostra a Tabela 9, o melhor desempenho do SVM-RBF foi de 65%. Tal valor foi obtido para o *corpus* 466 usando a combinação de AZPort-*features* e WE (média), bem como usando a combinação de todos os atributos. Detalhes desse resultado podem ser encontrados nas Tabelas 19, 21 e 29 do Apêndice A. Para o SVM-linear o valor de 64% foi obtido no *corpus* 466 usando a combinação de todos os atributos: WE (Word2Vec–CBOW–1000 e Word2Vec–Skip-gram–600) com o AZPort-*features* e TF-IDF. O K-NN, com WE (Wang2Vec–Skip-gram–média) de dimensão 300 e AZPort-*features*, atingiu 63%. O algoritmo Naive Bayes alcançou 50% e 54% com as variações Gaussian e Bernoulli, respectivamente. O valor de 50% foi atingido com a combinação de WE (Glove–soma) de dimensão 1000 com TF-IDF no *corpus* 366; e o de 54% com as combinações de WE (Glove–soma) com TF-IDF e de todas as representações de características (Glove–soma e média), ambas com WE de dimensão 100 no *corpus* 832. Assim como o SVM, o DT também alcançou 64% para o *corpus* 466, porém com a combinação de AZPort-*features* com TF-IDF. Por fim, as versões 1 e 2 do CRF obtiveram *f1-scores* de 77% e 94%, respectivamente. O

CRF, versão 1, alcançou esse resultado com a combinação de todas as formas de representações de *features* (Wang2Vec–CBOW de dimensão 50 e Wang2Vec–Skip-gram de dimensões 50 e 600) para o *corpus* 466. O CRF2 atingiu seu maior desempenho com a mesma combinação (porém com os modelos Word2Vec–Skip-gram–soma de dimensão 1000 e GloVe–média de dimensão 300), além de ter obtido com a combinação de WE (Wang2Vec–Skip-gram–média de dimensão 100) com AZPort-*features*, também para o *corpus* 466.

Ainda na Tabela 9, é possível observar que as características extraídas de TF-IDF e de *word embeddings*, individualmente, não proporcionou melhor desempenho para nenhum dos algoritmos testados. Além disso, o TF-IDF não mostrou ter influência no desempenho de algoritmos como SVM-RBF e CRF2. Em suas respectivas colunas, nas duas últimas linhas, em que foram obtidos suas melhores medidas, elas permanecem as mesmas. Outra observação a ser feita é o fato da utilização de características profundas (extraídas do AZPort) ter mostrado melhorias mais significativas nos desempenhos dos classificadores. No caso do CRF2, as características profundas se mostram fundamentais, já que utilizando apenas AZPort-*features* o CRF2 atingiu uma medida F1 de 93% para o *corpus* 466. Tal valor ficou apenas 1% abaixo de seus maiores *f1-scores* obtidos das combinações de WE com AzPort-*features* e das três formas de representação de características.

As duas últimas linhas da Tabela 9 mostram a média e o desvio padrão das medidas F1 obtidas para cada classificador. Em relação às médias, observa-se que o melhor algoritmo foi o CRF2, com média de aproximadamente 75%. O CRF2 foi também o classificador com maior dispersão (desvio padrão igual a 23%) de desempenho. O segundo melhor classificador foi o CRF, com média de aproximadamente 65% e com menor dispersão do que o CRF2. Algoritmos como os SVMs (RBF e linear) mostraram médias similares (57% e 56%, respectivamente) e, além de terem as terceiras melhores médias, mostraram baixos desvios padrões (8% para o RBF e 6% para o linear).

Para complementar a análise em termos de resultado médios, a Tabela 10 mostra a média e o desvio padrão dos melhores desempenhos obtidos com cada combinação de características. Os dados dessa tabela deixam mais evidente o impacto das características profundas do AZPort no desempenho dos classificadores, independente do algoritmo usado. A média de desempenho com AzPort-*features* foi de 57%, que foi a melhor média obtida para cada classe de características sozinha. A melhor média de desempenho foi atingida para as combinações de *word embeddings* com as características do AZPort e de todas as características, 64%. A adição de características TF-IDF não mostrou melhora no desempenho. Isso mostra que a adição de informações aos vetores de características não melhora o desempenho necessariamente. De fato, para *corpora* pequenos, como os usados neste trabalho, o aumento exagerado das dimensões dos vetores de características pode prejudicar o desempenho do classificador.

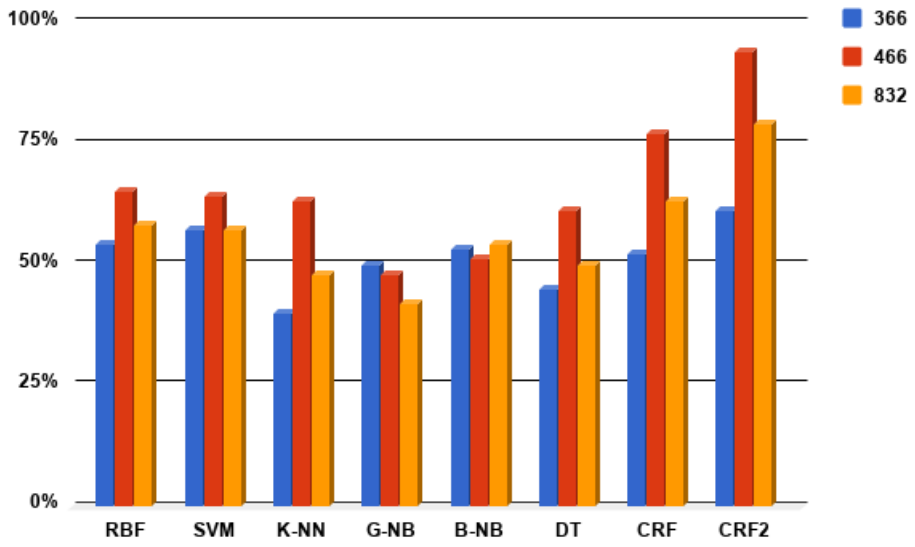
Tabela 10 – Média e Desvio Padrão dos melhores desempenhos obtidos com cada combinação de características

Características	Média	Desvio Padrão
TF-IDF	47%	11%
WE	43%	7%
AZPort	57%	20%
TF-IDF + AZPort	62%	17%
WE + TF-IDF	51%	6%
WE + AZPort	64%	15%
Todos	64%	17%

Fonte: autoria própria

Para uma melhor visualização dos resultados por algoritmo e por *corpora*, a Figura 11 mostra um gráfico de colunas com as maiores porcentagens obtidas por cada algoritmo utilizando cada um dos três *corpora*. No geral, o *corpus* que proveu melhores resultados foi o *corpus* 466. Para o CRF, o *corpus* 466 permitiu resultados acima de 75%; e, para o CRF2, acima de 90%. Em seguida, o corpus 832 foi melhor em relação ao corpus 366. Apenas no caso do *Naive Bayes* com variação de Bernoulli (B-NB), tal corpus foi o melhor; e, apenas no caso do *Naive Bayes* com variação Gaussian (G-NB), utilizar o corpus 366 mostrou o maior resultado.

Figura 11 – Melhores resultados obtidos em cada *corpus*

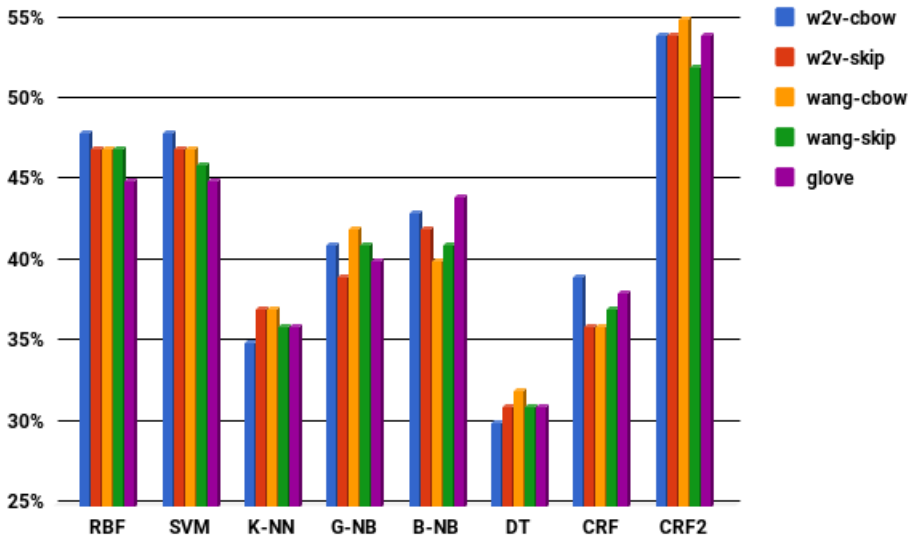


Fonte: autoria própria

O gráfico da Figura 12 mostra o melhor desempenho obtido usando apenas características WE extraídas a partir dos cinco modelos de *embeddings* avaliados (Word2Vec–CBOW,

Word2Vec–Skip-gram, Wang2Vec–CBOW, Wang2Vec–Skip-gram ou GloVe). Os classificadores SVM-linear, K-NN e CRF mostraram mesmo desempenho para os modelos Word2Vec–Skip-gram e Wang2Vec–CBOW. De maneira similar, o RBF apresentou mesmo desempenho com os modelos Word2Vec–Skip-gram e Wang2Vec (CBOW e Skip-gram). O CRF2 já apresenta mesmo desempenho com os modelos de Word2Vec. Um último classificador que apresenta mesmo desempenho entre os modelos é o *Decision Trees*, com os modelos Wang2Vec–Skip-gram e GloVe. No geral, o melhor modelo observado é o Wang2Vec–CBOW, uma vez que mostrou melhores resultados para os classificadores K-NN (empatado com Word2Vec–Skip-gram), G-NB, *Decision Trees* e CRF2. Em seguida, o segundo melhor modelo é o Word2Vec–CBOW, mostrando melhores desempenhos nos classificadores SVM-RBF, SVM-linear e CRF. Esse modelo é o pior para K-NN e DT.

Figura 12 – Melhores resultados obtidos em cada modelo de *word embedding*



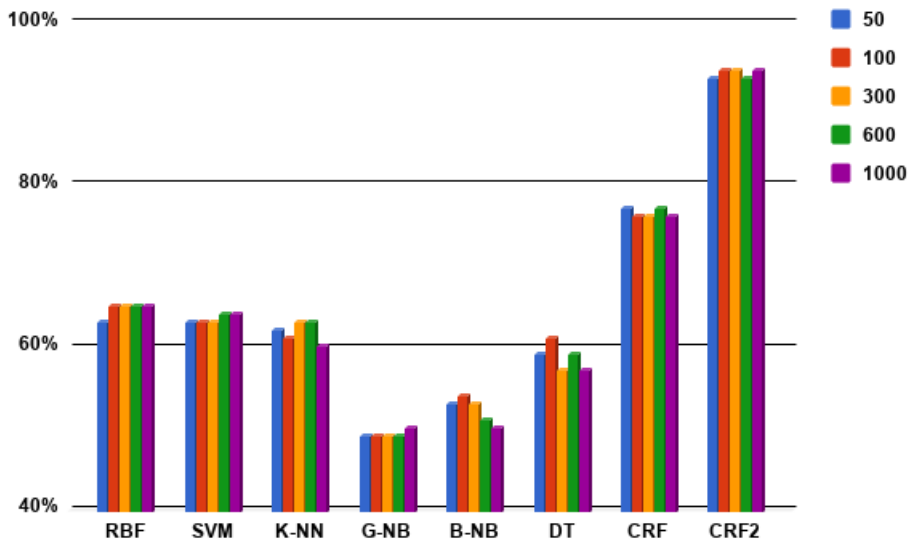
Fonte: autoria própria

Ainda na Figura 12, é possível notar que o modelo Wang2Vec–Skip-gram não é o melhor modelo em classificador algum, sendo o pior para o CRF2. Curiosamente, uma das combinações com as quais o CRF2 atinge o melhor desempenho, de 94%, é com o modelo Wang2Vec–Skip-gram combinado com AZPort-*features*, como já foi dito anteriormente. O modelo Word2Vec–Skip-gram é o melhor modelo apenas no K-NN e, ainda, divide a melhor posição com o Wang2Vec–CBOW, como já foi dito. Tal modelo é o pior apenas para o G-NB. Por fim, apenas o B-NB mostrou melhores resultados com o modelo GloVe, quase alcançando 45%.

Ainda analisando os modelos de WE, no gráfico da Figura 13 são comparados os melhores resultados obtidos com diferentes dimensões de *word embeddings*. Para os algoritmos

SVM-RBF, SVM-linear, K-NN, G-NB, CRF e CRF2, a variação da dimensão dos vetores de *word embeddings* não ofereceu variações significantes nos resultados. Já para algoritmos como B-NB e DT, o aumento da dimensão mostrou piora no desempenho. Uma possível causa para essa piora pode ser o tamanho reduzido dos *corpora*, uma vez que o aumento da dimensão do espaço de características mantendo um mesmo tamanho de *corpus* torna esse espaço mais esparso e, consequentemente, pode levar a um pior desempenho.

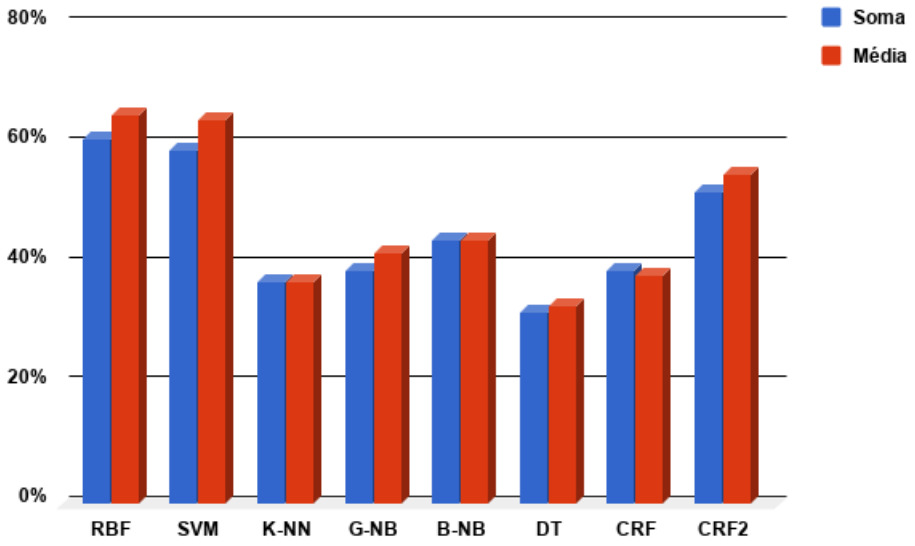
Figura 13 – Melhores resultados obtidos em cada dimensão do vetor de *word embedding*



Fonte: autoria própria

No gráfico da Figura 14 são comparados os melhores resultados obtidos com as duas formas de representação de WE para uma sentença – soma e média. Nesse gráfico, da mesma forma que o gráfico da Figura 12, os resultados utilizados foram resultantes de características extraídas apenas com WE. Conforme pode ser observado, os algoritmos K-NN e B-NB se mostraram indiferentes quanto à forma de representação. Para os algoritmos SVM-RBF, SVM-linear, G-NB, DT e CRF2, a melhor forma de representação foi a média. O único que mostrou melhores resultados com a soma foi o CRF.

Figura 14 – Melhores resultados obtidos em cada forma de representação de *word embedding* de uma sentença



Fonte: autoria própria

6 CONCLUSÃO

O objetivo deste trabalho foi extrair e avaliar o impacto de diferentes conjuntos de características na implementação de classificadores retóricos sentenciais para resumos científicos escritos em português. Trabalhos anteriores relatavam F1 de 72% usando características profundas e um classificador bayesiano (FELTRIM, 2004) e 57% usando características superficiais (TF-IDF) e um classificador SVM de *kernel* linear (IRIGUTI; FELTRIM, 2017). Neste trabalho, esses mesmos conjuntos de características foram avaliados sobre três *corpora* com diferentes algoritmos de aprendizado. Também foram avaliadas características geradas a partir de modelos de *word embeddings* (WE) pré-treinados, bem como as combinações dos diferentes conjuntos de características.

Os classificadores anteriormente implementados (SVM-linear, K-NN e *Decision Trees*) foram adaptados e novos foram implementados (SVM-RBF, G-NB, B-NB e CRF). Vale destacar que apenas alterando a parametrização do SVM – com base no algoritmo *Grid Search* – foi possível melhorar o desempenho dos algoritmos, principalmente no caso do SVM com kernel RBF.

Em termos das características avaliadas, constatou-se que a utilização de WE, individualmente, não mostrou melhoras significativas nos desempenhos dos classificadores em relação à utilização de apenas TF-IDF. Já a utilização dos atributos do AZPort, mesmo sozinhos, mostrou desempenho superior a utilização de TF-IDF e WE em mais de 20% para os algoritmos K-NN e CRF2. No caso do CRF2, o atributos profundos se mostraram bem efetivos, já que, apenas com eles, o CRF2 alcançou 93% em um dos *corpora* – apenas 1% a menos que o melhor desempenho obtido. Entretanto, a combinação de WE com outros atributos foi vantajosa, uma vez que classificadores como o G-NB e B-NB obtiveram seus melhores resultados com a combinação de TF-IDF e WE. Outros, como o SVM-RBF, K-NN e CRF2, obtiveram melhoras com a combinação de AZPort-*features* e WE. De fato, o maior desempenho atingido, de 94%, foi alcançado com essa combinação pelo CRF2.

Dentre os modelos de *word embeddings* utilizados, o melhor, no geral, foi o Wang2Vec-CBOW. Esse resultado é coerente com a conclusão de Hartmann *et al.* (2017), que, em seu trabalho, destacou o Wang2Vec como superior, e o de Sousa (2016), que destacou o CBOW. Em termos da variação da dimensão do vetor de WE, dimensões entre 100 e 300 mostraram-se mais adequadas, uma vez que foram com essas dimensões que o CRF2 alcançou máximo desempenho. Dimensões pequenas, como 50, e grandes, como 600 e 1000, não mostraram melhoria nos resultados. Isso pode estar relacionado à dimensão dos *corpora*. Por possuírem um número reduzido de sentenças, vetores de características muito pequenos ou exageradamente grandes acabam prejudicando os desempenhos dos classificadores.

Para a representação do vetor de WE de uma sentença, a melhor estratégia foi a média. Como foi explicado na Seção 5 e mostrado na Figura 14, a maioria dos classificadores mostraram melhores resultados com a média. Apenas dois se mostraram indiferentes e apenas um mostrou melhores resultados com a soma.

Em termos do algoritmo de aprendizado, o melhor classificador foi o CRF, atingindo 94% com a versão 2 e 77% com a versão 1. Isso pode se dar pela maneira que o CRF recebe como entrada as características. Enquanto os outros classificadores recebem matrizes com as características numéricas das sentenças, o CRF recebe um vetor de dicionário com as características numéricas e categóricas. Um outro possível fator é a forma que a combinação de TF-IDF com *word embeddings* foi implementada. Como o CRF obteve seus melhores resultados em que os atributos do AZPort estavam presentes (mas não foi igualmente fundamental para os outros classificadores), talvez apenas a concatenação dos vetores de atributos extraídos utilizando TF-IDF com os extraídos usando WE não tenha sido a melhor estratégia. Outro fator que pode ter contribuído para o desempenho do CRF é o fato dele maximizar a probabilidade das sequências de categorias para um resumo em vez de maximizar apenas probabilidade da categoria de uma sentença isolada.

Os resultados apresentados mostram que a utilização de atributos profundos, como os do AZPort, têm grande importância para esse tipo de classificação e que a combinação com *word embeddings* pode melhorar o desempenho. Além disso, evidenciou-se a vantagem do CRF para a tarefa em questão.

6.1 TRABALHOS FUTUROS

Um possível trabalho futuro é a implementação de outras formas de combinação do TF-IDF com os *word embeddings*, como, por exemplo, realizando a multiplicação ao invés de concatenação. Um outro possível trabalho é aplicar a seleção dos melhores atributos (usando qui-quadrado) nos *word embeddings* e nas combinações (através da concatenação e/ou da multiplicação) de TF-IDF com *word embeddings*. Além disso, como trabalho futuro, destaca-se a necessidade de uma análise mais cuidadosa dos *corpora*, visando identificar os motivos que levam os classificadores a terem melhores resultados no *corpus* 466.

REFERÊNCIAS

ANDREANI, A. C. **Predição estruturada aplicada à detecção de estrutura retórica**. 2017. Dissertação (Pós Graduação em Ciência da Computação), UEM (Universidade Estadual do Paraná), Maringá, Brazil.

ANDREANI, A. C.; FELTRIM, V. D. Campos aleatórios condicionais aplicados à detecção de estrutura retórica em resumos de textos acadêmicos em português (conditional random fields applied to rhetorical structure detection in academic abstracts in portuguese). In: **Proceedings of the 10th Brazilian Symposium in Information and Human Language Technology**. [S.l.: s.n.], 2015. p. 111–120.

ANTHONY, L. Writing research article introductions in software engineering: How accurate is a standard model? **IEEE transactions on Professional Communication**, IEEE, v. 42, n. 1, p. 38–46, 1999.

BRAGA, A. d. P.; CARVALHO, A.; LUDERMIR, T. B. **Redes neurais artificiais: teoria e aplicações**. [S.l.]: Livros Técnicos e Científicos, 2000.

BREIMAN, L. *et al.* **Classification and regression trees**. [S.l.]: Routledge, 1984.

FELTRIM, V. D. **Uma abordagem baseada em corpus e em sistemas de crítica para a construção de ambientes Web de auxilio à escrita acadêmica em português**. Tese (Doutorado) — USP (Universidade de São Paulo), São Carlos, Brazil, 2004.

FELTRIM, V. D.; ALUÍSIO, S. M.; NUNES, M. d. G. V. Analysis of the rhetorical structure of computer science abstracts in portuguese. In: **Proceedings of Corpus Linguistics**. [S.l.: s.n.], 2003. v. 16, p. 212–218.

HARTMANN, N. *et al.* Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **CoRR**, abs/1708.06025, 2017. Disponível em: <http://arxiv.org/abs/1708.06025>.

HAYKIN, S. Neural networks: a comprehensive foundation, 1999. **Mc Millan, New Jersey**, 2010.

IRIGUTI, A. H.; FELTRIM, V. D. Criação de classificadores retóricos para resumos científicos da pubmed. In: **Anais do XXVI Encontro Anual de Iniciação Científica (EAIC 2017)**. Maringá, PR: [s.n.], 2017.

JUNIOR, J. C. **Uma Ferramenta Baseada em Cenários para Elicitação e Modelagem de Requisitos**. Tese (Doutorado) — Universidade de São Paulo, 1998.

LAFFERTY, J. D.; MCCALLUM, A.; PEREIRA, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: **Proceedings of the Eighteenth International Conference on Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann publishers Inc., 2001. (ICML '01), p. 282–289. ISBN 1-55860-778-1. Disponível em: <http://dl.acm.org/citation.cfm?id=645530.655813>.

LEVY, O.; GOLDBERG, Y. Dependency-based word embeddings. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. [S.l.: s.n.], 2014. v. 2, p. 302–308.

LING, W. *et al.* Two/too simple adaptations of word2vec for syntax problems. In: **Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. [S.l.: s.n.], 2015. p. 1299–1304.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43–67, 2007.

MIKOLOV, T. *et al.* Efficient estimation of word representations in vector space. **CoRR**, abs/1301.3781, 2013. Disponível em: <http://arxiv.org/abs/1301.3781>.

____. Distributed representations of words and phrases and their compositionality. **CoRR**, abs/1310.4546, 2013. Disponível em: <http://arxiv.org/abs/1310.4546>.

MOISEN, G. Classification and regression trees. Elsevier, 2008.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

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)**. [S.l.: s.n.], 2014. p. 1532–1543.

REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. [S.l.]: Editora Manole Ltda, 2003.

ROMEIRO, A. K. Q. **UM ESTUDO SOBRE O USO DA TEORIA DA ESTRUTURA RETÓRICA (RST) PARA SUMARIZAR A SABEDORIA DA COLETIVIDADE**. 2016.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016.

SOUSA, S. d. **Estudo de modelos de word embedding**. 2016. Monografia (Bacharel em Ciência da Computação), UTFPR (Universidade Tecnológica Federal do Paraná), Medianeira, Brazil.

SWALES, J. M.; FEAK, C. B. Academic writing for graduate students: Essential tasks and skills: A course for nonnative speakers of english (english for specific purposes). **Ann Arbor, MI: The University of Michigan Press**, 1994.

TEUFEL, S.; MOENS, M. Summarizing scientific articles: experiments with relevance and rhetorical status. **Computational linguistics**, MIT Press, v. 28, n. 4, p. 409–445, 2002.

WEBBER, B.; EGG, M.; KORDONI, V. Discourse structure and language technology. **Natural Language Engineering**, Cambridge University Press, v. 18, n. 4, p. 437–490, 2012.

WEISSBERG, R.; BUKER, S. **Writing up research**. [S.l.]: Prentice Hall Englewood Cliffs, NJ, 1990.

WITTEN, I. H. *et al.* **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2016.

A TABELAS COM AS MEDIDAS F1 OBTIDAS EM CADA COMBINAÇÃO DE ATRIBUTOS

Tabela 11 – *f1-scores* utilizando TF-IDF

Corpus	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
366	46%	57%	29%	33%	51%	46%	44%	29%
466	40%	50%	26%	40%	44%	48%	63%	40%
832	44%	53%	28%	37%	51%	48%	53%	38%

Fonte: autoria própria

Tabela 12 – *f1-scores* utilizando AZPort*features*

Corpus	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
366	35%	35%	30%	18%	40%	33%	46%	51%
466	58%	48%	59%	27%	35%	63%	71%	93%
832	47%	37%	46%	7%	32%	51%	61%	79%

Fonte: autoria própria

Tabela 13 – *f1-scores* utilizando TF-IDF + AZPort-*features*

Corpus	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
366	45%	51%	36%	33%	49%	43%	49%	55%
466	63%	57%	55%	25%	44%	64%	76%	92%
832	52%	56%	48%	26%	52%	53%	62%	78%

Fonte: autoria própria

Tabela 14 – *f1-scores* Word2Vec (soma)

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	36%	34%	33%	32%	33%	27%	25%	41%
		100	34%	33%	31%	32%	36%	23%	25%	38%
		300	42%	42%	29%	35%	39%	30%	23%	44%
		600	45%	46%	29%	33%	41%	28%	25%	49%
		1000	46%	45%	31%	36%	42%	29%	25%	47%
	skip-gram	50	32%	32%	30%	29%	27%	22%	25%	42%
		100	35%	32%	32%	31%	34%	31%	30%	41%
		300	40%	40%	35%	35%	37%	27%	25%	47%
		600	42%	43%	37%	36%	41%	28%	25%	48%
		1000	43%	44%	34%	34%	42%	27%	27%	48%
Corpus 466	**CBOW**	50	34%	32%	31%	29%	31%	28%	34%	42%
		100	34%	36%	33%	31%	36%	26%	32%	44%
		300	40%	40%	32%	38%	41%	29%	34%	45%
		600	42%	41%	35%	37%	43%	27%	33%	49%
		1000	44%	43%	35%	37%	43%	27%	39%	50%
	skip-gram	50	33%	32%	32%	30%	31%	29%	34%	43%
		100	37%	33%	32%	34%	33%	30%	32%	44%
		300	41%	39%	31%	35%	37%	28%	34%	51%
		600	43%	42%	32%	35%	36%	28%	35%	48%
		1000	42%	44%	32%	39%	39%	26%	34%	52%
Corpus 832	**CBOW**	50	35%	37%	32%	33%	33%	26%	30%	45%
		100	36%	38%	32%	32%	37%	29%	28%	43%
		300	42%	41%	34%	37%	39%	24%	28%	45%
		600	46%	47%	32%	38%	41%	23%	30%	51%
		1000	48%	46%	30%	37%	43%	28%	29%	51%
	skip-gram	50	35%	37%	30%	31%	30%	25%	28%	46%
		100	38%	37%	33%	33%	33%	27%	31%	47%
		300	41%	43%	34%	34%	38%	25%	28%	48%
		600	45%	45%	33%	35%	37%	24%	28%	48%
		1000	47%	47%	34%	36%	40%	25%	36%	49%

Fonte: autoria própria

Tabela 15 – *f1-scores* Word2Vec (média)

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	35%	34%	28%	33%	33%	25%	25%	43%
		100	36%	33%	28%	33%	36%	21%	25%	45%
		300	41%	42%	32%	36%	39%	28%	23%	44%
		600	42%	43%	27%	36%	41%	25%	24%	45%
		1000	44%	48%	23%	40%	42%	24%	25%	42%
	skip-gram	50	34%	35%	30%	32%	27%	30%	25%	42%
		100	37%	34%	31%	34%	34%	28%	25%	42%
		300	38%	41%	29%	37%	37%	29%	25%	45%
		600	41%	46%	29%	39%	41%	27%	24%	43%
		1000	45%	43%	26%	38%	42%	29%	29%	42%
Corpus 466	**CBOW**	50	33%	30%	34%	34%	31%	27%	34%	48%
		100	35%	31%	35%	35%	36%	29%	32%	54%
		300	40%	42%	35%	41%	41%	26%	34%	50%
		600	42%	41%	34%	39%	43%	27%	35%	54%
		1000	45%	44%	33%	39%	43%	21%	38%	52%
	skip-gram	50	32%	33%	34%	34%	31%	28%	32%	50%
		100	36%	35%	35%	36%	33%	30%	35%	51%
		300	43%	42%	35%	37%	37%	29%	32%	52%
		600	44%	41%	34%	37%	36%	27%	35%	51%
		1000	44%	42%	35%	37%	39%	27%	35%	51%
Corpus 832	**CBOW**	50	37%	37%	30%	36%	33%	27%	31%	48%
		100	37%	37%	31%	34%	37%	25%	28%	50%
		300	42%	40%	32%	34%	39%	26%	31%	50%
		600	43%	44%	32%	37%	41%	29%	28%	51%
		1000	46%	44%	30%	35%	43%	26%	27%	54%
	skip-gram	50	35%	36%	33%	31%	30%	30%	28%	51%
		100	38%	36%	33%	33%	33%	26%	32%	52%
		300	43%	40%	34%	35%	38%	28%	28%	49%
		600	44%	42%	33%	37%	37%	23%	29%	54%
		1000	45%	42%	31%	36%	40%	27%	36%	52%

Fonte: autoria própria

Tabela 16 – *f1-scores* Word2Vec (soma) + TF-IDF

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	41%	39%	33%	49%	51%	45%	46%	43%
		100	42%	40%	34%	49%	51%	37%	45%	38%
		300	47%	46%	36%	49%	50%	37%	46%	44%
		600	48%	48%	36%	49%	48%	37%	46%	49%
		1000	50%	48%	34%	49%	48%	37%	44%	47%
	skip-gram	50	39%	40%	33%	49%	53%	40%	45%	43%
		100	42%	44%	34%	49%	52%	42%	46%	43%
		300	47%	43%	36%	49%	49%	39%	45%	47%
		600	48%	49%	37%	49%	48%	39%	45%	47%
		1000	49%	49%	40%	49%	47%	39%	46%	46%
Corpus 466	**CBOW**	50	37%	42%	33%	43%	46%	39%	51%	44%
		100	39%	39%	32%	43%	46%	43%	52%	46%
		300	44%	41%	37%	43%	48%	30%	52%	45%
		600	44%	44%	36%	43%	50%	33%	53%	48%
		1000	47%	47%	35%	43%	49%	31%	51%	50%
	skip-gram	50	37%	42%	34%	43%	46%	39%	52%	44%
		100	40%	37%	34%	43%	50%	37%	51%	45%
		300	44%	42%	35%	43%	47%	35%	52%	50%
		600	45%	45%	35%	43%	46%	31%	52%	49%
		1000	46%	46%	37%	43%	46%	32%	51%	52%
Corpus 832	**CBOW**	50	38%	45%	34%	40%	53%	38%	46%	50%
		100	38%	42%	33%	40%	54%	39%	46%	44%
		300	45%	43%	35%	40%	50%	35%	46%	45%
		600	48%	47%	31%	40%	48%	34%	46%	51%
		1000	49%	49%	31%	40%	47%	34%	44%	51%
	skip-gram	50	40%	44%	32%	40%	51%	41%	46%	53%
		100	40%	40%	34%	40%	51%	35%	43%	51%
		300	43%	44%	33%	40%	49%	38%	46%	47%
		600	49%	45%	33%	40%	47%	36%	41%	47%
		1000	51%	48%	36%	40%	45%	36%	46%	49%

Fonte: autoria própria

Tabela 17 – *f1-scores* Word2Vec (média) + TF-IDF

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	51%	51%	39%	34%	51%	42%	42%	49%
		100	53%	50%	39%	34%	53%	41%	45%	50%
		300	51%	52%	35%	34%	50%	35%	44%	51%
		600	54%	54%	34%	35%	48%	33%	48%	52%
		1000	53%	52%	34%	36%	48%	39%	44%	51%
	skip-gram	50	52%	50%	39%	34%	**53%**	41%	45%	50%
		100	52%	50%	38%	34%	52%	39%	45%	50%
		300	51%	51%	36%	35%	49%	39%	45%	46%
		600	53%	50%	35%	35%	48%	38%	44%	51%
		1000	52%	52%	35%	36%	47%	38%	48%	49%
Corpus 466	**CBOW**	50	46%	45%	33%	26%	46%	36%	53%	53%
		100	50%	49%	32%	26%	46%	36%	53%	57%
		300	54%	53%	32%	27%	48%	34%	54%	55%
		600	52%	52%	31%	27%	50%	33%	53%	57%
		1000	53%	51%	31%	28%	49%	34%	53%	57%
	skip-gram	50	46%	46%	30%	25%	46%	38%	54%	57%
		100	49%	48%	30%	25%	46%	38%	54%	57%
		300	52%	51%	31%	27%	47%	36%	53%	54%
		600	52%	51%	30%	28%	46%	35%	56%	54%
		1000	51%	51%	32%	29%	46%	30%	53%	54%
Corpus 832	**CBOW**	50	48%	50%	38%	27%	53%	36%	46%	56%
		100	50%	47%	36%	27%	54%	35%	45%	57%
		300	52%	48%	36%	28%	50%	34%	45%	55%
		600	52%	51%	34%	28%	48%	33%	45%	56%
		1000	54%	52%	33%	29%	47%	33%	47%	55%
	skip-gram	50	47%	47%	36%	27%	51%	39%	45%	58%
		100	48%	47%	35%	27%	51%	38%	46%	57%
		300	49%	50%	35%	28%	49%	35%	45%	54%
		600	51%	50%	36%	28%	47%	34%	45%	57%
		1000	52%	52%	34%	29%	45%	33%	48%	58%

Fonte: autoria própria

Tabela 18 – *f1-scores* Word2Vec (soma) + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	35%	31%	33%	28%	35%	31%	45%	45%
		100	37%	36%	29%	27%	39%	26%	46%	46%
		300	42%	40%	30%	33%	41%	28%	46%	50%
		600	45%	43%	31%	35%	41%	28%	45%	55%
		1000	45%	44%	32%	38%	43%	28%	44%	53%
	skip-gram	50	33%	33%	33%	26%	30%	29%	46%	50%
		100	37%	33%	32%	29%	37%	30%	46%	49%
		300	42%	39%	34%	34%	39%	27%	46%	51%
		600	43%	43%	35%	34%	42%	25%	45%	49%
		1000	53%	42%	34%	34%	42%	28%	45%	55%
Corpus 466	**CBOW**	50	52%	50%	50%	39%	37%	54%	70%	90%
		100	53%	52%	47%	40%	41%	57%	71%	89%
		300	58%	59%	42%	42%	43%	55%	69%	90%
		600	57%	57%	44%	40%	45%	55%	71%	91%
		1000	58%	57%	43%	40%	46%	55%	72%	90%
	skip-gram	50	51%	47%	44%	38%	41%	53%	71%	92%
		100	52%	51%	43%	44%	36%	56%	71%	91%
		300	55%	55%	40%	41%	39%	52%	71%	88%
		600	56%	57%	40%	40%	40%	59%	70%	90%
		1000	56%	57%	42%	41%	41%	57%	70%	92%
Corpus 832	**CBOW**	50	53%	44%	40%	27%	38%	43%	60%	72%
		100	45%	44%	38%	35%	40%	44%	62%	70%
		300	49%	48%	39%	38%	40%	45%	61%	72%
		600	51%	50%	37%	42%	42%	42%	61%	72%
		1000	51%	52%	34%	40%	43%	39%	62%	74%
	skip-gram	50	44%	43%	38%	26%	34%	45%	61%	73%
		100	45%	42%	41%	35%	36%	45%	61%	73%
		300	46%	44%	38%	36%	39%	42%	61%	72%
		600	51%	49%	37%	37%	37%	42%	60%	72%
		1000	51%	57%	37%	37%	41%	41%	61%	75%

Fonte: autoria própria

Tabela 19 – *f1-scores* Word2Vec (média) + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	41%	36%	36%	27%	35%	30%	49%	56%
		100	40%	35%	35%	30%	39%	24%	46%	55%
		300	41%	40%	36%	35%	41%	29%	46%	55%
		600	44%	39%	37%	41%	41%	24%	45%	55%
		1000	43%	42%	37%	41%	43%	30%	44%	55%
	skip-gram	50	40%	35%	36%	29%	30%	32%	46%	55%
		100	39%	33%	36%	35%	37%	29%	47%	54%
		300	42%	38%	36%	36%	39%	29%	46%	55%
		600	43%	38%	35%	38%	42%	29%	46%	55%
		1000	42%	39%	35%	40%	42%	28%	44%	56%
Corpus 466	**CBOW**	50	63%	50%	55%	42%	37%	54%	69%	91%
		100	64%	54%	55%	42%	41%	58%	69%	92%
		300	65%	58%	56%	46%	43%	53%	70%	92%
		600	64%	59%	56%	46%	45%	53%	70%	92%
		1000	63%	58%	56%	45%	46%	49%	71%	92%
	skip-gram	50	62%	49%	55%	39%	41%	53%	69%	92%
		100	65%	54%	55%	43%	36%	55%	70%	93%
		300	64%	58%	56%	43%	39%	54%	70%	92%
		600	65%	59%	56%	43%	40%	55%	70%	92%
		1000	65%	61%	56%	42%	41%	54%	69%	92%
Corpus 832	**CBOW**	50	50%	46%	47%	22%	38%	38%	61%	78%
		100	54%	44%	46%	30%	40%	45%	62%	77%
		300	54%	46%	46%	34%	40%	39%	61%	77%
		600	56%	50%	46%	38%	42%	41%	61%	77%
		1000	55%	51%	46%	36%	43%	41%	62%	77%
	skip-gram	50	51%	45%	47%	24%	34%	42%	62%	76%
		100	54%	45%	46%	29%	36%	40%	61%	77%
		300	55%	45%	47%	34%	39%	45%	62%	77%
		600	53%	50%	48%	37%	37%	41%	61%	78%
		1000	56%	50%	47%	36%	41%	39%	62%	78%

Fonte: autoria própria

Tabela 20 – *f1-scores* Word2Vec (soma) + TF-IDF + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	40%	38%	37%	34%	50%	39%	49%	48%
		100	41%	41%	35%	34%	52%	33%	49%	43%
		300	45%	43%	36%	34%	49%	38%	49%	51%
		600	48%	47%	36%	34%	49%	40%	48%	53%
		1000	48%	48%	36%	35%	49%	38%	48%	56%
	skip-gram	50	39%	37%	35%	34%	51%	39%	49%	50%
		100	42%	36%	35%	34%	52%	43%	48%	48%
		300	47%	48%	35%	34%	48%	38%	49%	51%
		600	49%	49%	37%	34%	48%	37%	48%	51%
		1000	49%	48%	37%	35%	48%	37%	51%	56%
Corpus 466	**CBOW**	50	52%	52%	44%	26%	47%	57%	76%	90%
		100	54%	51%	43%	26%	49%	53%	76%	90%
		300	61%	59%	41%	27%	51%	54%	76%	91%
		600	59%	59%	43%	28%	51%	54%	75%	91%
		1000	60%	57%	40%	28%	49%	54%	75%	90%
	skip-gram	50	53%	46%	43%	26%	47%	54%	76%	91%
		100	52%	52%	43%	26%	51%	53%	76%	91%
		300	57%	56%	42%	27%	49%	51%	76%	90%
		600	58%	58%	42%	28%	48%	54%	76%	90%
		1000	56%	59%	39%	29%	48%	53%	75%	94%
Corpus 832	**CBOW**	50	42%	45%	40%	27%	53%	48%	61%	72%
		100	44%	44%	39%	27%	52%	47%	62%	70%
		300	48%	48%	40%	28%	51%	48%	60%	72%
		600	51%	50%	36%	28%	49%	45%	62%	72%
		1000	51%	50%	37%	28%	48%	46%	62%	74%
	skip-gram	50	53%	46%	40%	27%	51%	48%	62%	72%
		100	47%	42%	38%	27%	52%	49%	61%	74%
		300	46%	45%	38%	27%	50%	47%	62%	71%
		600	50%	48%	38%	28%	47%	42%	61%	72%
		1000	53%	52%	38%	29%	46%	42%	61%	74%

Fonte: autoria própria

Tabela 21 – *f1-scores* Word2Vec (média) + TF-IDF + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	50%	48%	37%	34%	50%	39%	50%	**57%**
		100	49%	49%	38%	34%	52%	38%	49%	55%
		300	48%	47%	36%	35%	49%	35%	50%	55%
		600	49%	49%	37%	35%	49%	33%	51%	56%
		1000	47%	50%	37%	36%	49%	38%	48%	56%
	skip-gram	50	48%	46%	36%	33%	51%	39%	49%	55%
		100	50%	46%	36%	34%	52%	43%	50%	54%
		300	49%	48%	36%	34%	48%	38%	49%	55%
		600	49%	49%	35%	35%	48%	38%	47%	55%
		1000	49%	47%	35%	36%	48%	35%	51%	56%
Corpus 466	**CBOW**	50	63%	63%	55%	26%	47%	58%	76%	92%
		100	64%	63%	55%	26%	49%	61%	76%	92%
		300	64%	63%	56%	27%	51%	56%	75%	92%
		600	65%	63%	56%	27%	51%	52%	75%	93%
		1000	63%	64%	56%	28%	49%	45%	75%	92%
	skip-gram	50	62%	62%	55%	26%	47%	54%	76%	92%
		100	63%	62%	55%	26%	51%	55%	76%	93%
		300	65%	60%	56%	27%	49%	55%	76%	92%
		600	65%	64%	56%	28%	48%	51%	75%	92%
		1000	64%	63%	56%	30%	48%	55%	75%	92%
Corpus 832	**CBOW**	50	55%	53%	47%	27%	53%	48%	61%	78%
		100	54%	51%	46%	27%	52%	47%	62%	78%
		300	55%	51%	46%	28%	51%	44%	62%	78%
		600	56%	54%	46%	28%	49%	44%	61%	78%
		1000	56%	54%	46%	29%	48%	43%	62%	78%
	skip-gram	50	54%	51%	47%	27%	51%	48%	62%	77%
		100	57%	52%	46%	27%	52%	47%	62%	78%
		300	57%	51%	47%	28%	50%	47%	62%	78%
		600	57%	53%	48%	28%	47%	47%	61%	78%
		1000	58%	53%	47%	30%	46%	43%	61%	79%

Fonte: autoria própria

Tabela 22 – *f1-scores* Wang2Vec (soma)

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	35%	31%	31%	30%	32%	23%	26%	39%
		100	38%	37%	32%	32%	36%	25%	29%	34%
		300	45%	43%	31%	29%	37%	28%	24%	45%
		600	18%	45%	36%	35%	40%	27%	24%	43%
		1000	18%	43%	36%	34%	39%	30%	25%	46%
	skip-gram	50	38%	34%	29%	28%	28%	25%	27%	39%
		100	40%	39%	30%	28%	31%	27%	27%	39%
		300	43%	42%	35%	28%	37%	26%	27%	42%
		600	44%	46%	33%	31%	41%	30%	27%	43%
Corpus 466	**CBOW**	50	38%	37%	35%	36%	36%	26%	35%	46%
		100	42%	37%	36%	35%	39%	29%	33%	45%
		300	42%	41%	32%	31%	39%	29%	32%	48%
		600	21%	40%	33%	34%	34%	30%	33%	47%
		1000	21%	40%	32%	33%	37%	32%	34%	48%
	skip-gram	50	39%	34%	33%	31%	31%	30%	33%	43%
		100	44%	38%	34%	30%	37%	26%	32%	43%
		300	44%	43%	35%	31%	36%	31%	32%	50%
		600	44%	45%	32%	31%	41%	29%	34%	49%
Corpus 832	**CBOW**	50	39%	38%	33%	31%	33%	26%	32%	45%
		100	42%	41%	34%	34%	36%	25%	31%	47%
		300	43%	40%	35%	31%	40%	29%	28%	51%
		600	15%	46%	32%	35%	39%	26%	28%	50%
		1000	15%	47%	32%	35%	38%	30%	28%	48%
	skip-gram	50	37%	39%	32%	31%	30%	26%	31%	46%
		100	42%	39%	34%	29%	37%	27%	28%	44%
		300	43%	42%	34%	29%	37%	27%	28%	44%
		600	47%	46%	34%	31%	41%	27%	30%	50%

Fonte: autoria própria

Tabela 23 – *f1-scores* Wang2Vec (média)

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	38%	31%	32%	36%	32%	29%	33%	44%
		100	41%	39%	31%	34%	36%	25%	24%	39%
		300	47%	45%	31%	42%	37%	23%	24%	41%
		600	46%	45%	34%	36%	40%	27%	25%	47%
		1000	47%	44%	34%	38%	39%	28%	25%	55%
	skip-gram	50	36%	33%	35%	37%	28%	27%	26%	41%
		100	38%	39%	32%	38%	31%	28%	24%	43%
		300	43%	42%	28%	41%	37%	28%	25%	45%
		600	44%	44%	26%	41%	41%	28%	23%	44%
Corpus 466	**CBOW**	50	36%	32%	34%	39%	36%	27%	36%	50%
		100	41%	37%	37%	39%	39%	25%	35%	50%
		300	41%	43%	35%	37%	39%	31%	33%	52%
		600	39%	38%	35%	36%	34%	29%	33%	52%
		1000	41%	40%	37%	37%	37%	29%	32%	51%
	skip-gram	50	36%	31%	31%	38%	31%	29%	37%	49%
		100	41%	38%	36%	37%	37%	33%	33%	50%
		300	42%	42%	33%	41%	36%	27%	33%	52%
		600	47%	46%	33%	40%	41%	27%	32%	50%
Corpus 832	**CBOW**	50	39%	38%	33%	38%	33%	27%	33%	50%
		100	41%	38%	33%	37%	36%	29%	29%	52%
		300	43%	41%	36%	38%	40%	31%	28%	53%
		600	44%	44%	35%	37%	39%	30%	29%	54%
		1000	44%	44%	35%	36%	38%	31%	28%	53%
	skip-gram	50	38%	39%	31%	39%	30%	28%	31%	48%
		100	42%	40%	36%	37%	36%	25%	28%	50%
		300	43%	43%	35%	38%	37%	27%	27%	51%
		600	44%	43%	33%	40%	41%	27%	28%	52%

Fonte: autoria própria

Tabela 24 – *f1-scores* Wang2Vec (soma) + TF-IDF

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	42%	35%	34%	49%	50%	44%	45%	39%
		100	45%	44%	36%	49%	52%	38%	44%	34%
		300	48%	50%	34%	49%	48%	44%	45%	46%
		600	18%	46%	36%	49%	47%	41%	43%	44%
		1000	18%	44%	36%	49%	45%	38%	45%	45%
	skip-gram	50	45%	42%	31%	49%	49%	44%	42%	39%
		100	44%	44%	31%	49%	50%	43%	44%	42%
		300	47%	50%	37%	49%	48%	38%	46%	43%
		600	49%	51%	34%	48%	49%	40%	42%	44%
Corpus 466	**CBOW**	50	41%	42%	37%	43%	46%	39%	51%	46%
		100	45%	38%	39%	43%	46%	37%	52%	45%
		300	46%	42%	36%	43%	48%	35%	52%	48%
		600	21%	40%	33%	43%	43%	34%	47%	47%
		1000	21%	41%	31%	43%	41%	37%	52%	46%
	skip-gram	50	39%	43%	33%	43%	42%	40%	52%	47%
		100	45%	38%	35%	43%	48%	36%	50%	45%
		300	44%	43%	35%	43%	45%	33%	50%	50%
		600	45%	44%	35%	43%	47%	36%	51%	50%
Corpus 832	**CBOW**	50	41%	46%	32%	40%	52%	40%	44%	50%
		100	44%	43%	36%	40%	53%	38%	41%	50%
		300	47%	44%	35%	40%	52%	36%	45%	51%
		600	15%	46%	31%	40%	45%	40%	37%	49%
		1000	15%	46%	32%	40%	42%	32%	46%	48%
	skip-gram	50	40%	46%	31%	40%	51%	39%	45%	50%
		100	43%	42%	33%	40%	52%	40%	42%	49%
		300	46%	44%	36%	40%	49%	36%	44%	46%
		600	48%	47%	34%	40%	50%	39%	45%	50%

Fonte: autoria própria

Tabela 25 – *f1-scores* Wang2Vec (média) + TF-IDF

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	50%	46%	39%	34%	50%	40%	45%	50%
		100	51%	51%	36%	34%	52%	36%	43%	46%
		300	51%	53%	37%	34%	48%	36%	44%	49%
		600	50%	48%	34%	36%	47%	39%	45%	50%
		1000	49%	48%	35%	37%	45%	41%	45%	51%
	skip-gram	50	52%	49%	37%	34%	49%	39%	47%	47%
		100	51%	49%	34%	34%	50%	42%	42%	45%
		300	52%	50%	35%	34%	48%	39%	45%	46%
		600	52%	52%	34%	35%	49%	39%	45%	50%
Corpus 466	**CBOW**	50	46%	49%	34%	26%	46%	36%	52%	55%
		100	53%	50%	32%	26%	46%	39%	51%	56%
		300	48%	49%	32%	27%	48%	37%	50%	51%
		600	42%	41%	36%	29%	43%	32%	53%	53%
		1000	45%	42%	36%	30%	41%	37%	53%	49%
	skip-gram	50	48%	46%	31%	25%	42%	39%	52%	58%
		100	47%	50%	32%	26%	48%	35%	52%	54%
		300	48%	46%	34%	27%	45%	32%	50%	55%
		600	49%	50%	30%	28%	47%	35%	52%	56%
Corpus 832	**CBOW**	50	49%	49%	36%	27%	52%	38%	48%	58%
		100	50%	48%	35%	27%	53%	37%	47%	56%
		300	50%	46%	35%	28%	52%	34%	45%	56%
		600	45%	46%	36%	29%	45%	37%	46%	54%
		1000	47%	48%	36%	30%	42%	37%	45%	51%
	skip-gram	50	50%	48%	34%	27%	51%	40%	45%	56%
		100	48%	48%	36%	27%	52%	39%	46%	55%
		300	48%	48%	34%	28%	49%	40%	47%	54%
		600	50%	49%	34%	28%	50%	35%	48%	55%

Fonte: autoria própria

Tabela 26 – *f1-scores* Wang2Vec (soma) + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	39%	31%	32%	27%	34%	30%	44%	46%
		100	41%	37%	34%	27%	40%	30%	46%	41%
		300	43%	44%	29%	30%	37%	28%	45%	47%
		600	18%	46%	36%	35%	41%	31%	46%	44%
		1000	18%	44%	36%	35%	40%	31%	46%	44%
	skip-gram	50	37%	33%	29%	24%	31%	31%	47%	47%
		100	39%	37%	30%	25%	36%	29%	46%	47%
		300	43%	42%	32%	29%	40%	29%	45%	48%
		600	46%	45%	31%	30%	44%	28%	45%	50%
Corpus 466	**CBOW**	50	53%	53%	44%	41%	43%	59%	70%	90%
		100	54%	53%	43%	42%	43%	59%	72%	87%
		300	53%	55%	40%	39%	43%	55%	71%	87%
		600	21%	40%	34%	40%	36%	53%	70%	50%
		1000	21%	42%	33%	37%	38%	54%	71%	48%
	skip-gram	50	49%	51%	39%	36%	38%	55%	70%	89%
		100	55%	52%	40%	38%	42%	58%	71%	90%
		300	55%	55%	39%	38%	40%	57%	70%	87%
		600	54%	54%	38%	37%	42%	56%	71%	85%
Corpus 832	**CBOW**	50	45%	44%	39%	27%	38%	46%	61%	74%
		100	48%	48%	38%	37%	40%	44%	61%	73%
		300	49%	47%	37%	33%	40%	42%	62%	72%
		600	15%	46%	32%	37%	40%	45%	61%	53%
		1000	15%	47%	32%	36%	38%	41%	61%	52%
	skip-gram	50	45%	43%	35%	31%	32%	43%	60%	74%
		100	47%	43%	37%	34%	41%	42%	62%	75%
		300	47%	48%	36%	31%	38%	41%	62%	71%
		600	50%	49%	36%	33%	41%	46%	61%	71%

Fonte: autoria própria

Tabela 27 – *f1-scores* Wang2Vec (média) + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	40%	35%	33%	31%	34%	27%	43%	57%
		100	40%	36%	33%	31%	40%	28%	44%	56%
		300	44%	42%	32%	41%	37%	26%	45%	56%
		600	44%	42%	32%	37%	41%	27%	44%	59%
		1000	46%	43%	31%	38%	40%	26%	46%	58%
	skip-gram	50	40%	34%	31%	31%	31%	33%	48%	54%
		100	42%	34%	31%	35%	36%	28%	46%	56%
		300	45%	44%	32%	39%	40%	27%	44%	58%
		600	45%	43%	31%	41%	44%	27%	43%	56%
Corpus 466	**CBOW**	50	61%	55%	59%	43%	43%	57%	71%	92%
		100	64%	57%	61%	46%	43%	56%	71%	92%
		300	64%	57%	62%	45%	43%	52%	70%	92%
		600	57%	57%	52%	41%	36%	56%	70%	92%
		1000	56%	57%	52%	40%	38%	51%	70%	93%
	skip-gram	50	60%	50%	62%	45%	38%	55%	69%	93%
		100	62%	54%	61%	46%	42%	54%	70%	94%
		300	64%	59%	63%	48%	40%	57%	69%	93%
		600	62%	61%	63%	47%	42%	58%	69%	93%
Corpus 832	**CBOW**	50	52%	45%	47%	26%	38%	43%	61%	78%
		100	53%	46%	47%	30%	40%	40%	61%	78%
		300	55%	45%	47%	37%	40%	42%	61%	77%
		600	50%	51%	44%	36%	40%	42%	61%	76%
		1000	49%	50%	43%	36%	38%	43%	62%	77%
	skip-gram	50	53%	44%	48%	26%	32%	43%	60%	76%
		100	52%	43%	48%	32%	41%	44%	63%	76%
		300	53%	47%	47%	36%	38%	43%	62%	77%
		600	55%	51%	46%	39%	41%	45%	62%	78%

Fonte: autoria própria

Tabela 28 – *f1-scores* Wang2Vec (soma) + TF-IDF + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	42%	34%	35%	34%	48%	42%	49%	51%
		100	45%	45%	34%	34%	51%	35%	49%	41%
		300	47%	49%	31%	35%	47%	40%	50%	50%
		600	18%	47%	36%	35%	46%	39%	47%	43%
		1000	18%	45%	36%	36%	45%	40%	49%	46%
	skip-gram	50	42%	37%	32%	34%	48%	43%	50%	48%
		100	42%	41%	33%	35%	49%	41%	52%	49%
		300	47%	47%	35%	34%	48%	41%	50%	46%
		600	49%	51%	33%	34%	48%	41%	50%	50%
Corpus 466	**CBOW**	50	53%	53%	43%	26%	47%	56%	77%	91%
		100	57%	53%	43%	26%	48%	56%	76%	87%
		300	56%	55%	41%	27%	50%	51%	75%	89%
		600	21%	41%	34%	28%	44%	52%	76%	49%
		1000	21%	44%	32%	29%	42%	53%	76%	49%
	skip-gram	50	49%	48%	39%	26%	42%	56%	77%	89%
		100	54%	50%	38%	26%	49%	54%	74%	89%
		300	56%	56%	39%	27%	48%	55%	74%	87%
		600	56%	55%	39%	29%	48%	53%	77%	85%
Corpus 832	**CBOW**	50	45%	49%	39%	27%	53%	50%	62%	74%
		100	48%	47%	40%	27%	53%	48%	62%	73%
		300	49%	47%	39%	27%	53%	47%	61%	71%
		600	15%	47%	32%	29%	45%	48%	62%	51%
		1000	15%	48%	32%	30%	43%	44%	62%	52%
	skip-gram	50	46%	47%	33%	27%	51%	48%	62%	73%
		100	48%	43%	35%	27%	52%	46%	62%	73%
		300	49%	48%	39%	27%	50%	46%	61%	69%
		600	52%	49%	37%	28%	50%	45%	60%	70%

Fonte: autoria própria

Tabela 29 – *f1-scores* Wang2Vec (média) + TF-IDF + AZPort-*features*

		Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	**CBOW**	50	49%	47%	36%	34%	48%	38%	49%	58%
		100	49%	45%	36%	34%	51%	39%	49%	56%
		300	49%	51%	35%	35%	47%	35%	50%	57%
		600	50%	47%	36%	36%	46%	38%	47%	59%
		1000	49%	48%	36%	37%	45%	38%	49%	59%
	skip-gram	50	48%	48%	35%	33%	48%	40%	48%	57%
		100	49%	44%	36%	34%	49%	44%	49%	59%
		300	48%	46%	35%	35%	48%	39%	49%	57%
		600	48%	48%	35%	35%	48%	38%	50%	57%
Corpus 466	**CBOW**	50	61%	61%	56%	26%	47%	56%	76%	92%
		100	63%	61%	57%	26%	48%	56%	76%	92%
		300	63%	60%	57%	27%	50%	54%	75%	92%
		600	56%	57%	49%	29%	44%	53%	74%	92%
		1000	55%	57%	49%	31%	42%	50%	76%	93%
	skip-gram	50	61%	57%	56%	25%	42%	57%	76%	93%
		100	60%	57%	57%	26%	49%	54%	75%	93%
		300	64%	61%	58%	27%	48%	53%	75%	93%
		600	65%	63%	58%	28%	48%	48%	76%	92%
Corpus 832	**CBOW**	50	54%	54%	47%	27%	53%	49%	62%	79%
		100	54%	52%	47%	27%	53%	45%	60%	77%
		300	55%	49%	46%	28%	53%	46%	61%	78%
		600	51%	49%	42%	29%	45%	46%	61%	76%
		1000	50%	50%	42%	30%	43%	45%	62%	76%
	skip-gram	50	54%	51%	46%	27%	51%	46%	61%	76%
		100	53%	51%	46%	27%	52%	46%	61%	77%
		300	52%	49%	46%	28%	50%	45%	61%	77%
		600	54%	54%	46%	29%	50%	43%	61%	78%

Fonte: autoria própria

Tabela 30 – *f1-scores* GloVe (soma)

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	32%	33%	27%	25%	31%	26%	25%	39%
	100	38%	35%	32%	30%	40%	26%	33%	38%
	300	39%	44%	31%	34%	39%	29%	28%	47%
	600	38%	43%	35%	33%	40%	29%	25%	44%
	1000	38%	45%	33%	33%	44%	31%	25%	45%
Corpus 466	50	37%	34%	32%	32%	34%	27%	34%	42%
	100	41%	35%	33%	33%	36%	28%	33%	40%
	300	38%	37%	36%	35%	36%	30%	32%	41%
	600	38%	38%	33%	36%	36%	28%	35%	46%
	1000	39%	41%	36%	36%	37%	27%	34%	52%
Corpus 832	50	36%	39%	29%	30%	31%	28%	28%	45%
	100	40%	37%	33%	31%	36%	28%	35%	47%
	300	43%	40%	34%	35%	39%	26%	33%	46%
	600	42%	43%	32%	33%	37%	27%	29%	49%
	1000	42%	45%	32%	34%	40%	30%	28%	47%

Fonte: autoria própria

Tabela 31 – *f1-scores* GloVe (média)

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	37%	37%	34%	36%	31%	22%	25%	35%
	100	38%	34%	32%	39%	40%	25%	33%	42%
	300	43%	42%	30%	40%	39%	24%	26%	49%
	600	44%	44%	28%	38%	40%	24%	29%	47%
	1000	45%	45%	25%	37%	44%	26%	25%	50%
Corpus 466	50	37%	33%	30%	35%	34%	28%	32%	45%
	100	40%	34%	32%	36%	36%	29%	38%	46%
	300	39%	38%	33%	34%	36%	30%	35%	51%
	600	36%	37%	33%	40%	36%	27%	37%	49%
	1000	39%	38%	34%	38%	37%	27%	32%	54%
Corpus 832	50	38%	39%	32%	35%	31%	28%	28%	45%
	100	40%	39%	34%	35%	36%	28%	34%	46%
	300	42%	40%	32%	36%	39%	28%	34%	48%
	600	44%	45%	31%	37%	37%	29%	29%	49%
	1000	44%	44%	29%	38%	40%	28%	28%	52%

Fonte: autoria própria

Tabela 32 – *f1-scores* GloVe (soma) + TF-IDF

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	39%	37%	33%	48%	50%	44%	45%	42%
	100	44%	40%	33%	48%	52%	38%	45%	38%
	300	43%	48%	33%	49%	51%	40%	43%	48%
	600	42%	48%	37%	49%	48%	41%	47%	43%
	1000	43%	50%	35%	50%	49%	37%	45%	44%
Corpus 466	50	41%	38%	32%	43%	48%	42%	52%	47%
	100	42%	37%	33%	43%	47%	39%	52%	42%
	300	42%	38%	35%	43%	48%	35%	50%	42%
	600	40%	38%	34%	43%	44%	33%	51%	45%
	1000	41%	42%	36%	43%	43%	31%	52%	49%
Corpus 832	50	40%	46%	30%	40%	51%	41%	46%	51%
	100	41%	42%	32%	40%	54%	36%	46%	48%
	300	44%	43%	33%	40%	52%	38%	45%	48%
	600	43%	44%	32%	40%	47%	39%	44%	50%
	1000	43%	47%	32%	40%	45%	37%	46%	47%

Fonte: autoria própria

Tabela 33 – *f1-scores* GloVe (média) + TF-IDF

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	48%	47%	35%	34%	50%	44%	45%	44%
	100	49%	45%	37%	35%	52%	39%	48%	47%
	300	52%	51%	32%	34%	51%	38%	48%	50%
	600	52%	52%	30%	34%	48%	41%	49%	50%
	1000	52%	53%	29%	37%	49%	38%	45%	50%
Corpus 466	50	45%	44%	35%	25%	48%	35%	54%	52%
	100	47%	43%	35%	26%	47%	38%	53%	50%
	300	45%	45%	31%	27%	48%	34%	56%	50%
	600	43%	42%	33%	28%	44%	33%	55%	50%
	1000	45%	45%	34%	29%	43%	30%	54%	53%
Corpus 832	50	47%	49%	33%	27%	51%	36%	45%	53%
	100	48%	45%	36%	27%	54%	39%	47%	53%
	300	47%	45%	35%	28%	52%	34%	47%	50%
	600	48%	47%	33%	28%	47%	33%	45%	50%
	1000	48%	49%	33%	30%	45%	34%	45%	53%

Fonte: autoria própria

Tabela 34 – *f1-scores* GloVe (soma) + AZPort-*features*

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	37%	34%	31%	25%	37%	31%	46%	46%
	100	40%	35%	31%	26%	39%	28%	47%	48%
	300	41%	43%	31%	33%	38%	27%	47%	46%
	600	40%	43%	35%	32%	40%	30%	45%	44%
	1000	39%	44%	34%	35%	43%	29%	46%	47%
Corpus 466	50	53%	47%	36%	42%	39%	58%	71%	89%
	100	52%	47%	36%	43%	42%	55%	70%	86%
	300	48%	46%	39%	42%	39%	53%	70%	70%
	600	45%	46%	35%	40%	38%	53%	71%	74%
	1000	45%	47%	36%	41%	38%	54%	71%	77%
Corpus 832	50	43%	43%	33%	34%	37%	44%	61%	73%
	100	45%	43%	34%	38%	40%	44%	61%	72%
	300	46%	46%	33%	38%	41%	40%	62%	67%
	600	45%	47%	33%	37%	39%	43%	62%	67%
	1000	46%	50%	32%	37%	40%	44%	61%	68%

Fonte: autoria própria

Tabela 35 – *f1-scores* GloVe (média) + AZPort-*features*

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	37%	35%	31%	28%	37%	29%	46%	52%
	100	41%	36%	30%	32%	39%	25%	51%	56%
	300	41%	41%	31%	39%	38%	27%	43%	56%
	600	42%	42%	31%	41%	40%	23%	45%	59%
	1000	43%	42%	31%	39%	43%	28%	46%	59%
Corpus 466	50	63%	50%	62%	42%	39%	55%	69%	92%
	100	62%	53%	61%	44%	42%	58%	69%	92%
	300	59%	54%	63%	41%	39%	55%	70%	93%
	600	61%	58%	61%	45%	38%	55%	68%	93%
	1000	61%	58%	60%	42%	38%	54%	69%	93%
Corpus 832	50	52%	44%	48%	23%	37%	44%	62%	76%
	100	52%	44%	48%	31%	40%	43%	62%	77%
	300	49%	44%	47%	35%	41%	43%	62%	78%
	600	51%	49%	47%	37%	39%	42%	61%	78%
	1000	52%	49%	46%	38%	40%	44%	62%	78%

Fonte: autoria própria

Tabela 36 – *f1-scores* GloVe (soma) + TF-IDF + AZPort-*features*

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	41%	37%	34%	35%	50%	37%	49%	45%
	100	45%	40%	31%	35%	50%	39%	49%	50%
	300	45%	46%	33%	35%	50%	39%	50%	47%
	600	44%	47%	36%	36%	49%	42%	48%	45%
	1000	44%	51%	35%	37%	50%	37%	49%	47%
Corpus 466	50	53%	50%	36%	27%	48%	58%	76%	89%
	100	53%	48%	36%	27%	49%	54%	76%	85%
	300	50%	47%	40%	28%	51%	57%	76%	70%
	600	45%	48%	37%	28%	48%	49%	73%	73%
	1000	45%	49%	36%	31%	44%	52%	76%	76%
Corpus 832	50	43%	47%	33%	28%	52%	46%	62%	75%
	100	47%	43%	34%	28%	54%	46%	61%	72%
	300	48%	46%	34%	29%	51%	47%	62%	66%
	600	46%	47%	32%	30%	46%	45%	60%	67%
	1000	46%	50%	33%	32%	45%	46%	62%	67%

Fonte: autoria própria

Tabela 37 – *f1-scores* GloVe (média) + TF-IDF + AZPort-*features*

	Dim	RBF	SVM	K-NN	G-NB	B-NB	DT	CRF	CRF2
Corpus 366	50	46%	43%	37%	34%	50%	43%	49%	54%
	100	46%	44%	36%	34%	50%	39%	51%	59%
	300	48%	50%	37%	35%	50%	37%	50%	56%
	600	47%	48%	36%	35%	49%	36%	52%	61%
	1000	46%	47%	36%	36%	50%	37%	49%	59%
Corpus 466	50	63%	58%	57%	26%	48%	56%	76%	92%
	100	62%	58%	57%	26%	49%	58%	76%	93%
	300	60%	57%	57%	27%	51%	57%	74%	94%
	600	60%	59%	56%	28%	48%	54%	74%	93%
	1000	60%	59%	56%	29%	44%	55%	76%	93%
Corpus 832	50	53%	50%	46%	27%	52%	46%	62%	77%
	100	52%	47%	46%	27%	54%	46%	61%	78%
	300	49%	47%	46%	28%	51%	45%	63%	79%
	600	52%	49%	45%	29%	46%	48%	61%	79%
	1000	52%	51%	46%	30%	45%	45%	62%	78%

Fonte: autoria própria