INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Luiz Filipe Martins Ramos
Bernardo Monteiro Rufino

Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural

Trabalho de Graduação
2015

Computação

Número da CDU (tamanho 10)

Luiz Filipe Martins Ramos
Bernardo Monteiro Rufino

Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural

Orientador
Prof. Dr. Paulo André Lima de Castro

Engenharia de Computação

SÃO JOSÉ DOS CAMPOS
INSTITUTO TECNOLÓGICO DE AERONÁUTICA

2015

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Ramos, Luiz Filipe Martins
Rufino, Bernardo Monteiro
Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural / Luiz Filipe Martins Ramos, Bernardo Monteiro Rufino
São José dos Campos, 2015
Número de folhas no formato 999f.

Trabalho de Graduação – Divisão de Ciência da Computação –
Instituto Tecnológico de Aeronáutica, 2015. Orientador: Prof. Dr. Paulo André Lima de Castro

1. Inteligência Artificial. 2. Processamento de Linguagem Natural. 3. Redes Bayesianas. I. Nome do segundo autor, se houver. II. Instituto Tecnológico de Aeronáutica. III. Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural

REFERÊNCIA BIBLIOGRÁFICA

RAMOS, Luiz Filipe Martins; RUFINO, Bernardo Monteiro. **Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural**. 2015. **Total de folhas**. Trabalho de Conclusão de Curso. (Graduação) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS – (NEGRITO, TAMANHO 12)

NOME DO AUTOR: Luiz Filipe Martins Ramos; Bernardo Monteiro Rufino
TÍTULO DO TRABALHO: Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural
TIPO DO TRABALHO/ANO: Graduação / 2015

Luiz Filipe Martins Ramos
Pça Mal-do-Ar Eduardo Gomes, 50
12228-900 – São José dos Campos – SP

Bernardo Monteiro Rufino
Pça Mal-do-Ar Eduardo Gomes, 50
12228-900 – São José dos Campos - SP

DESENVOLVIMENTO DE UM CLASSIFICADOR DE POSTAGENS EM REDE SOCIAL UTILIZANDO PROCESSAMENTO DE LINGUAGEM NATURAL

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação

_____

Luiz Filipe Martins Ramos
Autor

_____

Bernardo Monteiro Rufino
Autor

_____

Prof. Dr. Paulo André Lima de Castro (ITA)
Orientador

_____

Prof. Dr. Carlos Henrique Costa Ribeiro (ITA)
Coordenador do Curso de Engenharia de Computação

São José dos Campos, _____ de _____ de _____

Agradecimentos

4.0141 There is a general rule by means of which the musician can obtain the symphony from the score, and which makes it possible to derive the symphony from the groove on the gramophone record, and, using the first rule, to derive the score again. That is what constitutes the inner similarity between these things which seem to be constructed in such entirely different ways. And that rule is the law of projection which projects the symphony into the language of musical notation. It is the rule for translating this language into the language of gramophone records.

Ludwig Wittgenstein. Tractatus Logico-Philosophicus.

Resumo

Abstract

Sumário

Agradecimentos p. 6

Resumo p. 8

Abstract p. 9

Lista de Figuras p. 15

Lista de Tabelas p. 17

Lista de Algoritmos p. 18

Lista de Abreviaturas p. 20

1 Introdução p. 21

1.1 Motivação . . . . . p. 21

1.2 Objetivos . . . . . p. 21

1.3 Organização do Texto . . . . . p. 22

2 Definição do Problema p. 23

2.1 O Problema de Classificação . . . . . p. 23

2.1.1 Classificação Binária . . . . . p. 24

2.1.2 Classificação *Multi-Class* . . . . . p. 24

2.1.3 Classificação *Multi-Class / Multi-Label* . . . . . p. 24

2.2 Objeto de Estudo deste Trabalho . . . . . p. 25

3 Fundamentação Teórica p. 26

3.1 Definição formal . . . . . p. 26
3.2 Tipos de classificadores . . . . . p. 26
3.3 Redes Bayesianas . . . . . p. 27
3.3.1 Definição . . . . . p. 27
3.3.2 Regra de Bayes . . . . . p. 27
3.3.3 Naïve Bayes . . . . . p. 28
3.3.3.1 Problema a ser resolvido . . . . . p. 28
3.3.3.2 O que são Naïve Bayes . . . . . p. 29
3.3.3.3 Independência Condicional . . . . . p. 29
3.3.3.4 Inferência . . . . . p. 30
3.3.3.5 Classificação utilizando Naïve Bayes . . . . . p. 31
3.3.3.6 Smoothing . . . . . p. 32
3.3.4 Modelagens para classificação de texto . . . . . p. 33
3.3.4.1 Naïve Bayes Binária . . . . . p. 33
3.3.4.2 Naïve Bayes de Bernoulli . . . . . p. 33
3.3.4.3 Multinomial Naïve Bayes . . . . . p. 33
3.3.4.4 Outras features . . . . . p. 33
3.4 Weighted Naïve Bayes . . . . . p. 33
3.5 Métodos de avaliação de um classificador . . . . . p. 34
3.5.1 Classificador Binário . . . . . p. 35
3.5.1.1 Acurácia . . . . . p. 35
3.5.1.2 Precisão . . . . . p. 36
3.5.1.3 Abrangência (*Recall*) . . . . . p. 36
3.5.1.4 Fscore . . . . . p. 37
3.5.2 Classificador *Multi-Class* . . . . . p. 38
3.5.2.1 Matriz de Confusão (*Confusion Matrix*) . . . . . p. 38

3.5.2.2 Médias micro e macro (*Micro and Macro Averaging*) . p. 39
3.5.2.3 Fscore . . . . . p. 40
3.5.2.4 Acurácia . . . . . p. 40
3.5.2.5 Medida Kappa . . . . . p. 41
3.6 Revisão Bibliográfica . . . . . p. 43

4 Metodologia p. 44
4.1 Plano de desenvolvimento . . . . . p. 44
4.1.1 Estudo basico sobre Processamento de Linguagem Natural . . p. 44
4.1.2 Coleta de dados . . . . . p. 45
4.1.3 Pré-processamento dos textos e normalização . . . . . p. 45
4.1.4 Criação dos classificadores de Redes Bayesianas . . . . . p. 45
4.1.5 Estudo dos resultados obtidos no conjunto de validação para diferentes features e parâmetros . . . . . p. 46
4.1.6 Definição da arquitetura do classificador e sua implementação p. 46
4.1.7 Desenvolvimento do plugin . . . . . p. 46
4.1.8 Testes de usabilidade . . . . . p. 46
4.2 Aquisição de dados . . . . . p. 46
4.2.1 Extensão do Chrome de classificação de postagens . . . . . p. 47
4.2.1.1 Funcionamento de extensões do Chrome . . . . . p. 47
4.2.1.2 Manifesto da extensão . . . . . p. 48
4.2.1.3 Estrutura do DOM do Facebook . . . . . p. 49
4.2.1.4 UI desenvolvida . . . . . p. 51
4.2.1.5 Servidor . . . . . p. 53
4.2.1.6 Crawler . . . . . p. 54
4.3 Processamento do texto . . . . . p. 54
4.3.1 Tokenização . . . . . p. 54

4.3.2 Normalização . . . . . p. 55
4.3.3 Stemming . . . . . p. 55
4.3.4 Processamento utilizado . . . . . p. 55
4.4 Tecnologias utilizadas . . . . . p. 56
4.4.1 Extensão para Chrome . . . . . p. 56
4.4.2 HTML / Javascript / CSS . . . . . p. 56
4.4.3 Git . . . . . p. 56
4.4.4 Google App Engine . . . . . p. 56
4.4.5 Python . . . . . p. 56

5 Propostas de Classificadores p. 57
5.1 Classes adotadas . . . . . p. 57
5.2 Naïve Bayes utilizando apenas o texto . . . . . p. 58
5.3 Naïve Bayes com Features Adicionais . . . . . p. 58
5.4 Weighted Naïve Bayes utilizando apenas o texto . . . . . p. 59
5.5 Weighted Naïve Bayes com Features Adicionais . . . . . p. 59
5.6 Utilização dos links . . . . . p. 59
5.7 Concatenação do texto dos links . . . . . p. 60
5.8 Fusão de classes pequenas . . . . . p. 60

6 Validação - Experimentos, resultados e análise p. 61
6.1 Base de dados . . . . . p. 61
6.2 Naïve Bayes utilizando apenas o texto . . . . . p. 62
6.3 Naïve Bayes com Features Adicionais . . . . . p. 65
6.4 Weighted Naïve Bayes utilizando apenas o texto . . . . . p. 70
6.5 Weighted Naïve Bayes com Features Adicionais . . . . . p. 73
6.6 Utilização dos links . . . . . p. 77

6.7 Concatenação do texto dos links . . . . . p. 77

6.8 Fusão de classes pequenas . . . . . p. 77

6.9 Extensão final desenvolvida . . . . . p. 77

6.10 Teste de usabilidade . . . . . p. 77

7 Conclusões e Trabalhos Futuros p. 78

Referências p. 79

Lista de Figuras

1 Esquema ilustrativo da função de um classificador . . . . . p. 23

2 Exemplo de Redes Bayesianas . . . . . p. 28

3 Exemplo de uma Naïve Bayes . . . . . p. 29

4 Exemplo de Matriz de Confusão para um classificador retirado de (1) p. 39

5 Exemplo de HTML do Facebook, com algumas classes aleatórias e algumas de nome legível . . . . . p. 50

6 Exemplo ilustrativo da estrutura básica da árvore de DOM do Facebook p. 51

7 Exemplo em uma postagem do cabeçalho contendo as possíveis classes p. 52

8 Exemplo da UI de uma postagem classificada pelo usuário . . . . . p. 53

9 Barra de opções para modificar a escolha do assunto na extensão do Chrome . . . . . p. 54

10 Histograma representativo da base de dados de postagens adquirida p. 61

11 Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para NB simples . . . . . p. 63

12 Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para NB simples . . . . . p. 63

13 Heatmap da matriz de confusão da Tabela 3 . . . . . p. 65

14 Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para NB com features extras (alem das palavras do texto) . . . . . p. 66

15 Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para NB com features extras (alem das palavras do texto) . . . . . p. 67

16 Sobreposição dos gráficos do Kappa em função da quantidade de postagens na base de dados para NB com e sem as features extras . . . p. 67

17 Heatmap da matriz de confusão da Tabela 5 . . . . . p. 69

18 Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB utilizando apenas o texto . . . . . p. 70

19 Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB utilizando apenas o texto p. 71

20 Sobreposição dos gráficos do Kappa em função da quantidade de postagens na base de dados para o NB Simples e o Weighted NB . . . . . p. 71

21 Heatmap da matriz de confusão da Tabela 7 . . . . . p. 72

22 Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB com features extras . p. 74

23 Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB com features extras . . . p. 75

24 Comparação dos quatro classificadores propostos, mostrando o Kappa de cada um como uma função do tamanho da base de dados. . . . . p. 75

25 Heatmap da matriz de confusão da Tabela 9 . . . . . p. 76

Lista de Tabelas

1 Resumo das medidas de performance para classificadores binários . p. 35

2 Nomeclatura para diferentes intervalos de $Kappa$ segundo os pesquisadores Landis e Koch . . . . . p. 43

3 Matriz de confusao para a NB apenas com o texto . . . . . p. 64

4 Precisão e abrangencia para NB apenas com o texto . . . . . p. 66

5 Matriz de confusao para a NB com features adicionais . . . . . p. 68

6 Precisão e abrangencia para NB com features adicionais . . . . . p. 69

7 Matriz de confusao para a Weighted NB apenas com texto . . . . . p. 72

8 Precisão e abrangencia para Weighted NB apenas com texto . . . . . p. 73

9 Matriz de confusao para a Weighted NB com features extras . . . . . p. 76

10 Precisão e abrangencia para Weighted NB com features extras . . . . p. 77

Lista de Algoritmos

Lista de Listagens

4.1 Manifesto da extensão do Chrome para coleta de dados . . . . . p. 48

Lista de Abreviaturas

NB Naïve Bayes

1 Introdução

1.1 Motivação

Um dos temas atualmente mais ativos na área de Ciência da Computação é o de Inteligência Artificial, sendo o Processamento de Linguagem Natural um dos seus principais tópicos. Hoje em dia vemos diversas aplicações dessa área na indústria como, por exemplo, os assistentes digitais Google Now, Siri (Apple) e Cortana (Microsoft), os quais, dado um processamento inicial do áudio, interpretam perguntas dos usuários, produzindo respostas satisfatórias.

Paralelamente a isto, observa-se um crescimento acelerado de redes sociais conectando usuários do mundo inteiro. Elas possuem uma abrangente variedade de assuntos, desde discussões sobre política até comentários sobre novelas. Muitas vezes os usuários não se interessam por uma boa parte das postagens e simplesmente as ignoram. As próprias redes sociais empregam grande esforço em ranquear seus conteúdos e apenas mostrar para seus usuários o que lhes interessa, entretanto não há hoje uma forma do próprio usuário informar à rede social seus interesses e a partir daí filtrar postagens não desejadas.

O Processamento de Linguagem Natural pode ser empregado para se analisar o conteúdo dos diversos textos postados, os categorizando em diferentes classes, o que permite a realização de uma filtragem dos conteúdos por parte do usuário.

1.2 Objetivos

Este trabalho tem, como objetivo, a criação de um classificador de textos baseado em Redes Bayesianas com aprendizado supervisionado. Este classificador será acoplado a uma rede social específica (o Facebook), através de um plugin para o navegador Google Chrome de modo a permitir que o usuário tenha contato apenas

com postagens de assuntos de seu interesse.

1.3 Organização do Texto

Este texto foi organizado em capítulos de modo a proporcionar uma leitura simples, com uma progressão natural dos assuntos abordados. A estrutura escolhida foi a seguinte:

- **Capítulo 2:** Este capítulo irá definir o problema que será atacado ao longo do trabalho, clarificando os aspectos teóricos e práticos a serrem considerados.
- **Capítulo 3:**
- **Capítulo 4:**
- **Capítulo 5:**

2 Definição do Problema

Antes de se abordarem as soluções propostas, é necessário entender com detalhes o problema que está sendo analisado.

2.1 O Problema de Classificação

Em aprendizado de máquina, o problema de classificação consiste na tarefa de atribuir a objetos uma ou mais das diversas categorias pré-definidas (14). Ou seja, dado um objeto, o classificador deve ser capaz de identificar em qual categoria tal objeto se encaixa melhor, como pode ser visualizado na Figura 1.

O problema de classificação em geral é o exemplo clássico de aprendizado de máquina supervisionado. A partir de um conjunto de exemplos previamente classificados por um supervisor experiente (em geral um ou mais humanos com conhecimento de domínio), o classificador é capaz de aprender e generalizar, podendo assim categorizar corretamente novos objetos encontrados.

Existem vários tipos diferentes de classificação baseados na quantidade de categorias pré-definidas assim como na quantidade de classes as quais cada objeto pode pertencer.

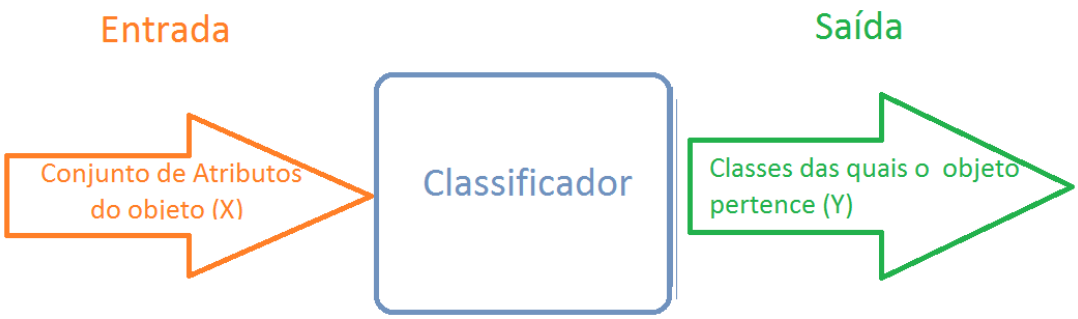

Figura 1: Esquema ilustrativo da função de um classificador

2.1.1 Classificação Binária

A classificação binária é o caso mais simples que pode ser estudado. Neste problema, há apenas duas classes diferentes e cada objeto pertence a exatamente uma delas.

Exemplos comuns que podem ser citados são a classificação de emails em spam ou não-spam, a classificação de tumores em benignos ou malignos, a determinação quais produtos deverão ser descartados em uma linha de produção, a detecção de transações financeiras fraudulentas, etc.

2.1.2 Classificação *Multi-Class*

Quando há mais de duas classes possíveis, chama-se o problema de classificação *multi-class*. Neste caso, cada objeto deve pertencer a exatamente uma dentre as várias categorias pré definidas. O classificador deve ser capaz de determinar qual é a categoria de cada objeto.

Vários exemplos de classificação podem ser considerados. Dadas imagens de frutas (que podem ser maçãs, peras, ou laranjas) determinar qual é o tipo de cada fruta. Ou então dadas imagens de telescópio de galaxias distantes, determinar o tipo da galaxia em questão (elíptica, espiral, etc).

2.1.3 Classificação *Multi-Class / Multi-Label*

Quando há várias classes possíveis (mais de duas) e cada objeto pode pertencer a mais de uma classe, chama-se o problema de *multi-class / multi-label*. Este é o mais difícil dos três problemas apresentados, uma vez que há diversas possíveis combinações de classes para cada objeto.

A classificação de textos é um exemplo clássico de classificação *multi-class / multi-label*, uma vez que o texto pode pertencer a uma ou varias classes simultaneamente.

Para resolver o problema de classificação *multi-class / multi-label* podem ser utilizadas duas abordagens diferentes. Uma delas é a abordagem *one vs all* e a outra é a abordagem dos subconjuntos.

Na abordagem *one vs all*, para cada classe, realiza-se a classificação binaria do

objeto em *pertence* ou *não pertence* a esta classe. Desta forma determina-se todas as classes às quais o dado objeto pertence. O problema desta abordagem é que pequenos erros que façam com que o classificador de uma das classes não fique bom (dados ruins para uma classe por exemplo) torna o resultado inteiro ruim.

Na abordagem de subconjuntos, consideram-se todos os possíveis subconjuntos das classes (com 1, 2, 3, ..., L elementos, onde L é o total de classes) e escolhe-se o subconjunto desejado (utilizando um classificador multi-class). O grande problema desta abordagem é que ha 2^{L-1} possíveis subconjuntos da classes (o que pode ser muito).

2.2 Objeto de Estudo deste Trabalho

Como foi visto, a classificação de textos é um problema do tipo *multi-class / multi-label*, entretanto seria necessária uma quantidade muito grande de dados para se obter resultados satisfatórios neste problema. Portanto, fez-se a simplificação de considerar o problema apenas *multi-class*, ou seja, cada texto só será classificado em um única classe.

3 Fundamentação Teórica

3.1 Definição formal

Utilizando a notação adotada por Dan Jurafsky e Christopher Manning em seu curso de Processamento de Linguagem Natural para Stanford (7), define-se o problema da classificação supervisionada de textos da seguinte forma.

Seja $C = \{c_1, c_2, ..., c_J\}$ um conjunto fixo de classes, $D = \{d_1, d_2, ..., d_n\}$ um conjunto de documentos, e $\mathcal{T} = \{(d_1, c_{d_1}), (d_2, c_{d_2}), ..., (d_m, c_{d_m})\}$ um conjunto de treinamento (subconjunto de D) com m documentos classificados manualmente, o classificador consiste em uma função $\gamma : D \to C$ que relaciona um documento a uma classe, e um algoritmo de aprendizado de máquina supervisionado é um algoritmo que recebe como parâmetros C e $\mathcal{T}$ e retorna γ.

3.2 Tipos de classificadores

Existem diversos tipos diferentes de classificadores que possuem resultados muito bons dependendo do problema analisado. Segue abaixo uma lista dos principais classificadores existentes.

- Árvores de Decisão
- Naïve Bayes
- Regressão Logística
- Support Vector Machines
- k-Nearest Neightbors
- Redes Neurais

- Dentre outros

A performance de todos estes métodos varia consideravelmente dependendo da aplicação. Estudos mostram que para bases de dados grandes o suficiente, ótimos resultados podem ser atingidos independentemente do método utilizado (6). Entretanto, para uma quantidade pequena de dados as Naïve Bayes apresentam resultados bons por serem classificadores de alto *bias* / baixa variância (3).

As NB possuem as vantagens de serem fáceis de se implementar, serem bem rápidas na hora da execução e mostrarem bons resultados práticos.

3.3 Redes Bayesianas

3.3.1 Definição

Em modelagem gráfica probabilística, Redes Bayesianas são grafos direcionados que representam relações de dependência condicionais entre diferentes variáveis aleatórias (11). A partir da visualização de uma Rede Bayesiana é possível utilizar a regra de Bayes para realizar inferências e descobrir a probabilidade de eventos, dadas algumas variáveis observadas.As arestas direcionadas representam as noções de causalidade entre as variáveis aleatórias e geram as dependências condicionais. Para cada nó do gráfico deve haver uma tabela de probabilidades condicionais para a variável em questão.

A Figura 2, retirada do curso de Modelagem Gráfica Probabilística da professora Daphne Koller de Stanford (8), ilustra um exemplo de uma Rede Bayesiana simples. Neste caso, pode-se ver que a nota do aluno é influenciada pela dificuldade da prova e pela sua inteligencia. Alem disso A possibilidade do professor escrever uma carta de recomendação depende apenas da nota do aluno e o SAT do aluno depende apenas de sua inteligencia. Em cada nó do grafo há uma tabela de distribuições de probabilidades condicionais.

3.3.2 Regra de Bayes

Para realizar inferências nas redes Bayesianas utiliza-se a regra de Bayes, como definida a seguir.

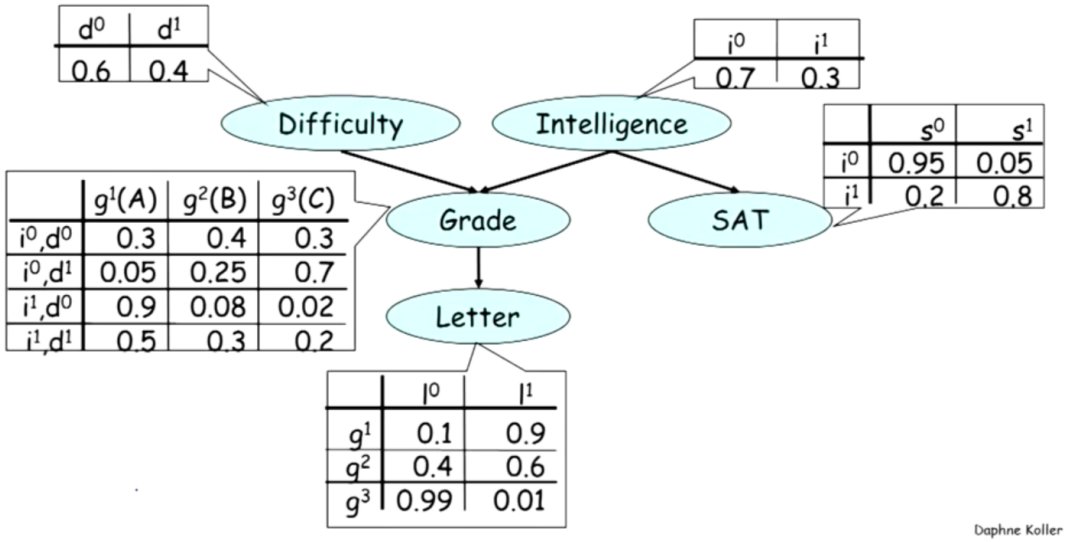

Figura 2: Exemplo de Redes Bayesianas

Sejam A e B dois eventos com probabilidades de ocorrência $P(A)$ e $P(B)$ e sendo $P(A|B)$ e $P(B|A)$ as probabilidades condicionais de A dado B e de B dado A, respectivamente, tem-se que:

$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$

Esse resultado parte da noção probabilidade conjunta $P(A, B)$.

$P(A, B) = P(A)P(B|A) = P(B)P(A|B) \rightarrow P(A|B) = \frac{P(A)P(B|A)}{P(B)}$

No caso do exemplo da Figura 2, podemos calcular a probabilidade conjunta da rede da seguinte forma:

$P(D, I, G, S, L) = P(D)P(I)P(G|I, D)P(S|I)P(L|G)$

Onde $D = Difficulty$, $G = Grade$, $I = Intelligence$, $S = SAT$ e $L = Letter$.

3.3.3 Naïve Bayes

3.3.3.1 Problema a ser resolvido

Redes Bayesianas são ferramentas excelentes para modelar problemas complexos, entretanto elas possuem um grande problema prático. A realização de inferências em redes genéricas é um problema NP-Hard, conforme demonstrado por Cooper (4).

O que é realizado na prática é, ou realizar inferências aproximadas nas redes com algoritmos polinomiais, ou simplificar as redes a alguns tipos específicos mais

simples.

3.3.3.2 O que são Naïve Bayes

As Naïve Bayes (bayes ingênuas) são simplificações feitas na modelagem de um problema por redes Bayesianas de modo a tornar possível realizar a inferência de forma rápida. Elas assumem que as variáveis do problema são condicionalmente independentes (mesmo que na prática elas não sejam, o que explica o nome *ingênuas*).

A Figura 3 mostra um exemplo de uma NB comum. Ela possui uma variável $Classe$ que depende de um série de outras varáveis $x_1, x_2, x_3, ..., x_n$ (que serão chamadas a partir daqui de features). É importante notar que a estrutura da rede mostra que as features são condicionalmente independentes umas das outras e a $Classe$ depende de cada uma das features individualmente. É fácil entender a grande vantagem desta abordagem. Cada tabela de probabilidades condicionais será pequena. Alem disso a inferência da variável classe, dadas algumas das features será bem simples, como será mostrado nas próximas seções.

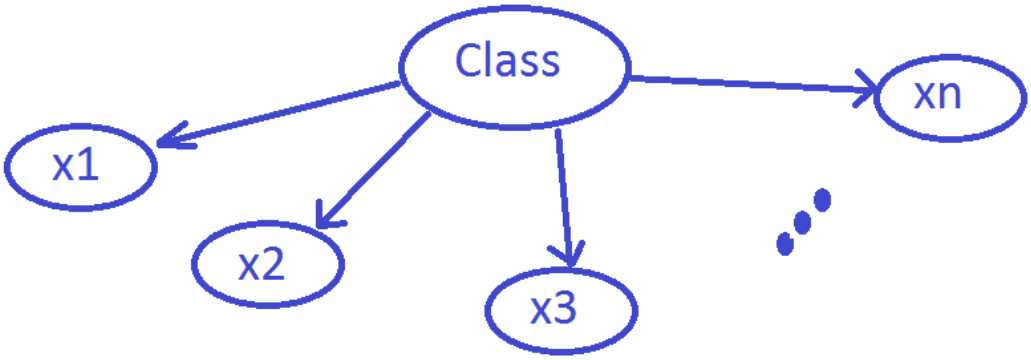

Figura 3: Exemplo de uma Naïve Bayes

Na prática o NB é um ótimo classificador de textos, pois, além de ser simples, traz resultados comparáveis a outros classificadores substancialmente mais complexos e lentos.

3.3.3.3 Independência Condicional

Antes de explicar a inferência em NB é importante entender o que significa o conceito de independência condicional.

Dados dois eventos A e B, dizemos que eles são independentes se a probabilidade de ocorrência de um deles não é influenciada pelo fato do outro ter ocorrido. Ou seja:

$P(A|B) = P(A)\ P(B|A) = P(B)$

Esta propriedade é muito relevante na simplificação de expressões pois podemos usar o fato de que a probabilidade conjunta é igual ao produto das probabilidades individuais.

$P(A, B) = P(A)P(B|A) = P(A)P(B)$

3.3.3.4 Inferência

A regra de Bayes seguida pela propriedade de independência condicional pode ser utilizada para a realização de inferência em NB da seguinte forma.

Sejam $C = \{c_1, c_2, c_3, ..., c_J\}$ um conjunto de classes e $x_1, x_2, x_3, ..., x_n$ as features a serem analisadas. Como trata-se de NB, assume-se independência condicional das features. Deseja-se saber para cada i:

$P(C = c_i|x_1, x_2, ..., x_n)$

Ou seja, deseja-se saber qual a probabilidade da classe possuir o valor c_i, dadas as features observadas.

Pela regra de Bayes temos:

$$P(C = c_i|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|C = c_i)P(C = c_i)}{P(x_1, x_2, ..., x_n)} \tag{3.1}$$

Como $x_1, x_2, x_3, ..., x_n$ são condicionalmente independentes entre si, temos:

$$
\begin{aligned}
P(x_1, x_2, ..., x_n|C = c_i) &= P(x_1, x_2, ..., x_{n-1}|x_n, C = c_i)P(x_n|C = c_i) \\
&= P(x_1, x_2, ..., x_{n-1}|C = c_i)P(x_n|C = c_i) \\
&= P(x_1, x_2, ..., x_{n-2}|x_{n-1}, C = c_i)P(x_{n-1}|C = c_i)P(x_n|C = c_i) \\
&= P(x_1, x_2, ..., x_{n-2}|C = c_i)P(x_{n-1}|C = c_i)P(x_n|C = c_i) \\
&= ... = P(x_1|C = c_i)P(x_2|C = c_i)P(x_3|C = c_i)...P(x_n|C = c_i) \\
&= \prod_{j=1}^{n} P(x_j|C = c_i)
\end{aligned}
\tag{3.2}
$$

Substituindo 3.2 em 3.1 temos:

$$
P(C = c_i|x_1, x_2, ..., x_n) = \frac{P(C = c_i)(\prod_{j=1}^{n} P(x_j|C = c_i))}{P(x_1, x_2, ..., x_n)} \tag{3.3}
$$

3.3.3.5 Classificação utilizando Naïve Bayes

No problema de classificação temos um documento d que será representado pelas features $x_1, x_2, ..., x_n$ e deseja-se saber de qual classe $c \in C$ este documento pertence.

Matematicamente, deseja-se saber de qual classe é mais provável que o documento pertença. Isto é, a classe que maximiza $P(d|c)$.

$\gamma(d) = argmax_c(P(d|c)) = argmax_c(P(x_1, x_2, ..., x_n|c))$

Pela equação 3.3, temos que:

$argmax_c(P(x_1, x_2, ..., x_n|c)) = argmax_c(\frac{P(C=c_i)(\prod_{j=1}^{n} P(x_j|C=c_i))}{P(x_1,x_2,...,x_n)})$

Como $P(x_1, x_2, ..., x_n)$ não depende de c, pode-se cortá-lo do denominador, chegando em:

$\gamma(d) = argmax_c(P(C = c_i)(\prod_{j=1}^{n} P(x_j|C = c_i))$

Agora, para descobrir a classe do documento, basta calcular $P(C = c_i)$ e $P(x_k|C = c_i)$ e ambas estas probabilidades são fáceis de serem estimadas (se tivermos um conjunto de treino suficientemente grande).

Assumindo que as features sejam binarias, isto é, ou estão presentes no docu-

mento, ou não estão. Seja $\#(x)$ um operador que indica quantas vezes a feature x aparece no conjunto de treinamento e $\#(c)$ quantos documentos do conjunto de treinamento possuem a classe c. Além disso, seja $\#(x \wedge c)$ a quantidade de vezes que a feature x aparece num documento de classe c e seja N o total de documentos do conjunto de treinamento. É fácil ver que:

$P(C = c_i) \simeq \frac{\#(c_i)}{N}$

e

$P(x_j|C = c_i) \simeq \frac{\#(x_j \wedge c_i)}{\#(c_i)}$

Ou seja, o problema de classificação se tornou basicamente um problema de contagem.

3.3.3.6 Smoothing

Um dos problemas práticos encontrados por NB é a ocorrência de contagens nulas. O grande problema do 0 é que se ele for apenas um dos fatores da multiplicação o resultado inteiro será 0, inutilizando o método.

Isto não é algo incomum, principalmente se o conjunto de treinamento for pequeno. Basta ter uma palavra no conjunto de teste que nunca ocorreu em uma determinada classe no conjunto de treinamento.

Existem técnicas que são utilizadas para resolver este problema e elas são chamadas de Smoothing. Neste trabalho utilizou-se uma das mais comuns: o Laplace Smoothing.

O Laplace consiste em assumir que todas as features foram vistas pelo menos α vezes em cada uma das classes. Isso se traduz nas seguintes formulas, sendo L o número total de classes e V o numero total de features.

$P(C = c_i) \simeq \frac{\#(c_i)+\alpha}{N+\alpha L}$

e

$P(x_j|C = c_i) \simeq \frac{\#(x_j \wedge c_i)+\alpha}{\#(c_i)+\alpha V}$

No caso especial em que $\alpha = 1$, tem-se o *Add-One Smoothing*.

3.3.4 Modelagens para classificação de texto

Uma vez que já foi entendida a forma de classificar o texto, resta apenas representá-lo por um conjunto de features.

3.3.4.1 Naïve Bayes Binária

Uma forma comum de representar um texto em features é considerá-lo como um conjunto de palavras. Assume-se que cada palavra é uma feature que pode estar presente ou não no documento. Nesta representação a ordem das palavras não é importante.

Esta representação não leva em consideração a frequência com a qual cada palavra aparece no texto. A fórmula para calcular a classe mais provável é aquela que foi desenvolvida acima.

3.3.4.2 Naïve Bayes de Bernoulli

3.3.4.3 Multinomial Naïve Bayes

É interessante levar em consideração a frequência da ocorrência das palavras uma vez que esta pode trazer informação relevante sobre a classe.

3.3.4.4 Outras features

É possível enriquecer o classificador utilizando outras features (que não sejam as próprias palavras do texto). Exemplos de features que podem ser utilizadas são: combinações de palavras, o tamanho do texto, quantidade de sinais de pontuação, quantidade de palavras com iniciais maiúsculas, etc.

Para o caso específico de postagens em redes sociais existem ainda outras features que podem ser incluídas. Pode-se considerar o autor da postagem, o momento em que ela foi publicada, a presença de fotos, vídeos ou links, etc.

3.4 Weighted Naïve Bayes

Um dos problemas das NB é que muitas vezes nas aplicações reais não é possível assumir a independência condicional das features. Muitas vezes uma das

features tem um peso maior que as outras, por exemplo.

Um modo inicial de relaxar essa hipótese de independência, é eliminar features com alta correlação, fazendo com que o subconjunto restante se encaixe melhor na hipótese de independência condicional. Isto é chamado de seleção de features.

Neste caso temos:

$\gamma(d) = argmax_c(P(C = c_i)(\prod_{j=1}^{n} P(x_j|C = c_i)^{I(j)})$

Onde:

$I(j) \in \{0, 1\}$

Uma abordagem mais genérica é ponderar cada feature de acordo com sua relevância. Ou seja:

$\gamma(d) = argmax_c(P(C = c_i)(\prod_{j=1}^{n} P(x_j|C = c_i)^{w(j)})$

Onde:

$w(j) \in \mathbb{R}^+$

Nota-se que a seleção de features é um caso específico da ponderação de features (onde $w(j)$ só pode ser 0 ou 1).

Agora o grande problema passa a ser determinar os pesos $w(j)$ das features. Há diversos algoritmos que ja foram propostos para realizar esta tarefa.

Neste trabalho, estudou-se a utilização de um método baseado na Teoria da Informação, que além de ser comum na literatura, possui fundamentos teóricos bem embasados (10).

3.5 Métodos de avaliação de um classificador

Uma vez desenvolvido o classificador é importante ser capaz de avaliá-lo a fim de determinar o quão bom ele é. A utilização de métricas numéricas para avaliar os classificadores é interessante pois permite a realização de comparações entre as diferentes versões implementadas, tornando possível determinar se as modificações que foram feitas estão fazendo efeito.

Existem diversas métricas que podem ser utilizadas para avaliar classificadores, algumas delas serão analisadas nesta seção (13).

3.5.1 Classificador Binário

Primeiro serão explorados os métodos de avaliação da performance do classificador binário, que é o classificador mais simples.

A tabela 1 resume as diferentes medidas que podem ser utilizadas, onde tp é o verdadeiro positivo, tn é o verdadeiro negativo, fp é o falso positivo e fn é o falso negativo.

Medida	Fórmula	Intuição
Acurácia	$\frac{tp+tn}{tp+fn+fp+tn}$	Eficácia global do classificador
Precisão	$\frac{tp}{tp+fp}$	Quantos dos objetos classificados como positivo são efetivam
Abrangência	$\frac{tp}{tp+fn}$	Quantos dos objetos positivos foram classificados como
Fscore	$\frac{(\beta^2+1)tp}{(\beta^2+1)tp+\beta^2 fn+fp}$	Uma média harmônica entre a precisão e a abrangê

Tabela 1: Resumo das medidas de performance para classificadores binários

3.5.1.1 Acurácia

A acurácia mede a quantidade total de acertos (verdadeiro positivos ou verdadeiro negativos) em relação à quantidade total de objetos classificados.

$acuracia = \frac{tp+tn}{tp+fn+fp+tn}$

Esta é a medida de performance mais fácil de se entender, entretanto ela não é muito informativa, uma vez que dependendo da distribuição das classes, classificadores ruins podem ter alta acurácia.

Suponha, por exemplo, um classificador que classifica tumores em benignos ou malignos. Suponha também que 99.9% dos tumores sejam benignos. É possível criar um classificador que sempre responde que o tumor é benigno e ele teria uma acurácia de 99.9% (aparentemente muito boa). Entretanto este classificador falha miseravelmente na sua tarefa principal (que é determinar quais tumores são malignos para que as devidas providencias sejam tomadas).

3.5.1.2 Precisão

A precisão é uma estatística que indica quantos dos objetos classificados como positivo são efetivamente positivos. Quando um classificador com alto precisão classifica um objeto como positivo, é muito provável que ele seja positivo (entretanto nada se sabe sobre quantos objetos positivos ele esta classificando como negativo).

$$precisao = \frac{tp}{tp+fp}$$

Um classificador que indica para uma pessoa se ela deve investir ou não em uma determinada ação deve ser muito preciso. Pois para o investidor o importante é que quando ele siga o conselho do classificador e invista, ele não perca dinheiro. Não importa tanto se havia várias outras ações que eram boas, mas que o classificador desprezou.

3.5.1.3 Abrangência (*Recall*)

A abrangência de um classificador indica quantos dos objetos positivos são efetivamente classificados como positivos. Um classificador com grande abrangência provavelmente irá detectar a maior parte dos objetos positivos (apesar de também poder detectar alguns negativos como sendo positivos).

$$abrangencia = \frac{tp}{tp+fn}$$

Um médico que classifica pintas na pele das pessoas entre malignas ou benignas (câncer de pele ou não) deve ter uma grande abrangência (sendo a classificação 'maligno' a classe positiva do classificador), mas a precisão não é tão importante. Isso ocorre pois se o médico deixar de classificar alguma pinta que era maligna como maligna, o paciente pode acabar evoluindo a doença e morrer (ou seja, todos que são efetivamente malignos devem ser classificados como malignos). A precisão não é tão importante pois se por ventura o médico classificar uma pinta benigna como maligna, o paciente vai remover a mesma e na biópsia será possível identificar que não era nada demais (o paciente só terá que passar inutilmente pelo procedimento cirúrgico de remoção de pintas, mas isso não é muito problemático).

Existe, normalmente uma relação inversa entre a precisão e a abrangência. Em geral quanto se aumenta a precisão se reduz a abrangência e vice-versa. Pode-se exemplificar isto com o mesmo caso do médico que classifica pintas. Quando ele

fica com dúvida se a pinta é benigna ou maligna, ele pode dizer que ela é maligna e pedir a remoção ou dizer que ela é benigna. Se na dúvida ele sempre disser que a pinta é maligna, ele estará aumentando a abrangência, mas diminuindo a precisão (pois haverá mais verdadeiros positivos e mais falso positivos). No limite, dizer que todas pintas são malignas dá abrangência máxima (100%). Se ele sempre disser que a pinta duvidosa é benigna, ele aumentará a precisão, mas diminuirá a abrangência (pois haverá mais falso negativos e menos falso positivos).

3.5.1.4 Fscore

Como pode ser visto, a abrangência e a precisão possuem comportamentos antagônicos. Normalmente é importante que o classificador tenha tanto uma precisão aceitável, quanto uma abrangência razoável. Então combinou-se a abrangência e a precisão em um único número, chamado de Fscore.

O caso mais simples do Fscore (chamado de F1score) é uma média harmônica entre a precisão e a abrangência.

$F_1 = \frac{2}{\frac{1}{precisao} + \frac{1}{abrangencia}}$

Ou seja:

$F_1 = \frac{2.precisao.abrangencia}{precisao+abrangencia}$

Isto pode ser simplificado para:

$F_1 = \frac{2tp}{2tp+fn+fp}$

É interessante notar que se a precisão ou se a abrangência forem bem pequenas, o F1score será bem pequeno. Se ambas forem grandes, o F1score será algum valor entre as duas.

Como dependendo da aplicação, a precisão ou a abrangência podem ser mais importantes, há uma formulação genérica para o Fscore que utiliza o parâmetro β para determinar qual das duas métricas é mais importante.

$F_\beta = \frac{(1+\beta^2).precisao.abrangencia}{\beta^2 precisao+abrangencia}$

É fácil provar que:

$F_\beta = \frac{(\beta^2+1)tp}{(\beta^2+1)tp+\beta^2 fn+fp}$

Os dois valores de β mais comuns são 0.5 e 2. Nota-se que para β maior que

1 a abrangência é considerada mais importante. Para β menor que 1 a precisão é considerada mais importante.

3.5.2 Classificador *Multi-Class*

As mesmas métricas definidas acima podem ser aplicadas para problemas *multi-class*, entretanto alguns problemas surgem. Por exemplo, as métricas seriam calculadas individualmente para cada classe (gerando vários números), o que tornaria a avaliação do classificador um processo complicado (vários números são mais difíceis de serem analisados do que apenas um único).

A tabela **??** resume as diferentes medidas utilizadas na avaliação de performance de um classificador *multi-class*.

3.5.2.1 Matriz de Confusão (*Confusion Matrix*)

Para facilitar a análise de problemas *multi-class*, criou-se a matriz de confusão. Ela consiste em uma matriz de números na qual as linhas representam as classificações realizadas pelo classificador e as colunas representam as classes originais das quais os objetos pertencem. A célula a_{ij} da matriz possui um número que indica quantas vezes o classificador classificou objetos de tipo j como i.

É fácil de notar que uma matriz de confusão de um classificador perfeito possui vários números positivos na diagonal principal e zero em todas as outras células. Nota-se que qualquer numero maior que zero em uma célula que não pertence a diagonal principal representa um erro. Com a matriz de confusão fica fácil de visualizar os erros mais comuns feitos pelo classificador.

A Figura 4 mostra uma matriz de confusão de um classificador que deve classificar expressões faciais.

A partir da matriz de confusão é fácil de determinar a precisão e a abrangência para cada classe i, conforme segue:

$$precisao(i) = \frac{a_{ii}}{\sum_{j=1}^{L} a_{ij}}$$

$$abrangencia(i) = \frac{a_{ii}}{\sum_{j=1}^{L} a_{ji}}$$

Ou seja, a precisão é um dos elementos da diagonal principal dividido pela soma de sua linha, enquanto que a abrangência é um dos elementos da diagonal principal

	Angry	bored	disgusted	Afraid	blissful	sad	neutral
angry	**80.31**	0.79	3.15	3.94	8.66	0	3.15
bored	0	**66.67**	6.17	2.47	0	7.41	17.28
disgusted	3.15	17.39	**52.17**	4.35	4.35	10.87	2.17
afraid	10.24	4.35	5.80	**40.58**	4.35	10.14	15.94
blissful	18.11	0	4.23	5.63	**52.11**	0	5.63
sad	0	11.29	0	1.61	0	**82.26**	4.85
neutral	2.36	17.71	5.06	7.59	0	2.53	**63.29**

Figura 4: Exemplo de Matriz de Confusão para um classificador retirado de (1)

dividido pela sua coluna.

Apesar de ter números específicos é muito bom pra analisar o quão bom um classificador é, é sempre importante observar a Matriz de Confusão pra detectar possíveis problemas e os lugares onda há erros mais comuns.

3.5.2.2 Médias micro e macro (*Micro and Macro Averaging*)

Conforme o descrito, pode-se calcular a precisão, a abrangência, o Fscore e a acurácia para cada classe individualmente. Entretanto seria interessante juntar todos estes dados em um único número. Para tal, faz-se uma média. Existem duas formas de realizar esta média, que são chamadas de micro e macro.

A média micro consiste em somar os numeradores e denominadores das formulas das métricas supracitadas separadamente (na prática isto é equivalente a realizar uma média ponderada nos denominadores). Seguem as fórmulas de precisão, abrangência e acurácia micro.

$$precisao_{\mu} = \frac{\sum_{i=1}^{L} tp_i}{\sum_{i=1}^{L} tp_i + fp_i}$$

$$abrangencia_{\mu} = \frac{\sum_{i=1}^{L} tp_i}{\sum_{i=1}^{L} tp_i + fn_i}$$

Com uma matemática simples da pra concluir que a precisão micro é numericamente igual à abrangência micro (basta notar que o numerador dos dois é igual

e o numerador dos dois representa a soma de todos os elementos da matriz de confusão, um através da soma das linhas e o outro através da soma das colunas).

A média macro consiste em somar os valores finais das métricas calculadas em cada classe e dividir pelo total de classes (uma média aritmética simples). Seguem as fórmulas de precisão e abrangência macro.

$$precisao_M = \frac{\sum_{i=1}^{L} \frac{tp_i}{tp_i+fp_i}}{L}$$

$$abrangencia_M = \frac{\sum_{i=1}^{L} \frac{tp_i}{tp_i+fn_i}}{L}$$

A intuição por trás dessas duas médias é que a média macro dá igual importância para todas as classes, enquanto que a média micro da mais importância para as classes mais comuns (no caso da abrangência) ou para as classes com mais objetos classificados nelas (no caso da precisão).

3.5.2.3 Fscore

Da mesma forma que foi definido o Fscore para o caso do classificador binário, é possível definir um Fscore macro e um Fscore micro (dependendo da utilização da precisão e abrangência macro ou micro).

$$F_{\beta\mu} = \frac{(1+\beta^2).precisao_\mu.abrangencia_\mu}{\beta^2 precisao_\mu + abrangencia_\mu}$$

$$F_{\beta M} = \frac{(1+\beta^2).precisao_M.abrangencia_M}{\beta^2 precisao_M + abrangencia_M}$$

Uma observação interessante é que, como a precisão micro e a abrangência micro são numericamente iguais, temos que o Fscore micro tambem terá este mesmo valor (independentemente de β). Para a média macro isto não acontece.

3.5.2.4 Acurácia

É possível encontrar na literatura duas definições diferentes de acurácia (2) (13).

Em uma das definições, a acurácia de um classificador *multi-class* é a média aritmética (média macro) das acurácias individuais de cada classe.

$$accuracy = \frac{\sum_{i=1}^{L} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{L}$$

Na outra definição, a acurácia é a quantidade total de acertos (considerando todas as classes), dividida pela quantidade total de documentos.

$$accuracy = \frac{\sum_{i=1}^{L} tp_i}{N} \tag{3.4}$$

Considerando a matriz de confusão, temos que:

$accuracy = \frac{\sum_{i=1}^{L} a_{ii}}{\sum_{i=1}^{L}\sum_{j=1}^{L} a_{ij}} = \frac{\sum_{i=1}^{L} a_{ii}}{N}$

Neste trabalho será utilizada esta última definição por ser mais intuitiva e dar mais informação sobre a qualidade do classificador (observe que a primeira definição de acurácia tende a sempre resultar em valores altos para problemas com muitas classes, uma vez que a quantidade de verdadeiros negativos tende a ser alta para todas as classes).

Observa-se que a acurácia também é numericamente igual à abrangência e à precisão com média micro.

3.5.2.5 Medida Kappa

Como já foi discutido previamente, a acurácia é uma medida muito ruim pois é possível desenvolver classificadores com comportamento trivial que possuem uma boa acurácia. Para resolver este problema, utilizam-se outras estatísticas em vez da acurácia que possuem mais valor semântico e dão mais informações sobre a performance do classificador. Já foram explicadas outras medidas que possuem mais informações (abrangência, precisão e Fscore micros e macros) e a matriz de confusão que permite a visualização de problemas recorrentes no classificador. Entretanto, para o problema *multi-class* é comum utilizar uma outra métrica (chamada de Kappa) para avaliar a performance do classificador em questão.

A estatística Kappa é utilizada para comparar a Acurácia Observada com a Acurácia Esperada. Esta estatística é muito poderosa e permite a realização de comparações até mesmo entre classificadores diferentes (12). Ela utiliza a comparação em relação a um classificador aleatório de modo a tornar esta estatística mais confiável que uma simples acurácia.

A Acurácia Observada é a mesma acurácia que já foi definida previamente na equação 3.4. Ela representa a quantidade de objetos classificados corretamente em relação ao total.

Define-se como Acurácia Observada, a porcentagem esperada de acertos que um classificador aleatório teria se ele classificar os objetos com a mesma proporção

entre as classes que o classificador original.

Para se deduzir a fórmula da Acurácia Esperada primeiro calcula-se a quantidade de vezes que o classificador original escolheu a classe c_i. É fácil de ver que ele escolheu esta classe $tp_i + fp_i$ vezes. O valor esperado do número de acertos de um classificador aleatório que escolha a dada classe $tp_i + fp_i$ vezes é:

$E[c_i] = (tp_i + fp_i)\frac{tp_i+fn_i}{N}$

Isto ocorre pois $\frac{tp_i+fn_i}{N}$ é a probabilidade de que um objeto aleatório pertença à classe i. A quantidade esperada para o número de acertos em todas as classes é de:

$E[c_1, c_2, ..., c_L] = \sum_{i=1}^{L} (tp_i + fp_i)\frac{tp_i+fn_i}{N}$

Como a Acurácia Esperada é a porcentagem de acertos desse classificador aleatório tem-se:

$$acuracia\_esperada = \frac{E[c_1, c_2, ..., c_L]}{N} = \frac{\sum_{i=1}^{L} (tp_i + fp_i)(tp_i + fn_i)}{N^2} \tag{3.5}$$

Em termos da matriz de confusão, a acurácia esperada é a somatória dos produtos entre cada linha com a sua respectiva coluna (linha 1 com coluna 1, linha 2 com coluna 2 e assim por diante) dividida pela quantidade total de elementos ao quadrado.

A definição de Acurácia Esperada é importante pois um classificador de Acurácia Observada 80% é muito mais impressionante se a Acurácia Esperada for de 1% do que se ela for 79%.

A estatística $Kappa$ indica a porcentagem de melhoria que o classificador tem em relação à Acurácia Esperada (15). Um classificador com Acurácia Observada idêntica à Acurácia Esperada possui $Kappa = 0$, enquanto que um classificador com Acurácia Observada de 100% possui $Kappa = 1$. Valores intermediários do $Kappa$ indicam a melhoria do classificador em relação ao aleatório. Desta forma, a fórmula para o $Kapp$ é:

$Kappa = \frac{acuracia\_observada - acuracia\_esperada}{1 - acuracia\_esperada}$

A interpretação do $Kappa$ é interessante pois permite a comparação de diversos classificadores. Como pode ser visto na fórmula, a estatística $Kappa$ indica o ganho percentual entre um classificador randômico (que classifique com mesma proporção de classes que o classificador desenvolvido) e o classificador dado. Um $Kappa$

de 0% significa que não há ganho enquanto que um $Kappa$ indica uma classificação perfeita. Landis e Koch fizeram um estudo dos diferentes valores de Kappa e chegaram à Tabela 2 abaixo de benchmarks para o $Kappa$ (9).

$Kappa$	Strength of Agreement
<0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substantial
0.81-1.00	Almost Perfect

Tabela 2: Nomeclatura para diferentes intervalos de $Kappa$ segundo os pesquisadores Landis e Koch

Estas divisões foram arbitradas pelos autores supracitados e elas criam um benchmark útil para a verificação da qualidade de um classificador, definindo uma nomeclatura que pode ser utilizada pelo meio acadêmico. Obviamente, dependendo da aplicação pode ser necessário e desejável que se tenha um $Kappa$ de 0.9. Em outras aplicações um $Kappa$ de 0.4 pode já ser suficiente.

3.6 Revisão Bibliográfica

4 Metodologia

4.1 Plano de desenvolvimento

Dividiu-se o desenvolvimento do projeto em diversas etapas.

4.1.1 Estudo basico sobre Processamento de Linguagem Natural

Inicialmente realizou-se um estudo aprofundado sobre o assunto do projeto para aproveitar o conhecimento já desenvolvido ao longo dos anos pela comunidade científica e garantir que as melhores tecnologias e técnicas fossem adotadas. O plano de estudos adotado foi:

(a) Expressões regulares

(b) Processamento de texto

(c) Normalização

(d) Tokenização das strings

(e) Modelagem linguística e simplificação de N-gramas

(f) Classificação de texto

(g) Naïve Bayes

(h) Métricas de performance (Precisão, Abrangência, Acurácia, etc)

(i) Melhorias para Naïve Bayes

4.1.2 Coleta de dados

Como trata-se de um projeto de Inteligência Artificial, é essencial que se obtenha uma base de dados grande o suficiente para que o sistema desenvolvido seja capaz de realizar as generalizações necessárias para um bom funcionamento sem que haja overfit.

Esta base de dados consiste em um conjunto de postagens (variável X) e o assunto da mesma (variável Y).

Foram propostas duas formas possíveis de aquisição de dados.

- Uma delas foi desenvolver um plugin que colete as postagens e pergunte o assunto para o usuário. A utilização deste plugin por várias pessoas permitiu a aquisição de uma base de dados considerável.
- Paralelamente desenvolveram-se crawlers para vasculhar sites que contenham artigos e textos sobre cada assunto que se deseja classificar. É importante ter a capacidade de classificar artigos de sites genéricos, pois muitas das postagens em rede social possuem links para tais sites.

4.1.3 Pré-processamento dos textos e normalização

Como trata-se de linguagem natural e ainda por cima coloquial, para atingir um bom resultado com o classificador é essencial que haja um bom pré-processamento do texto corrigindo palavras erradas, frases mal-construídas, normalizando termos parecidos, etc.

4.1.4 Criação dos classificadores de Redes Bayesianas

Foram criados classificadores de NB e suas performances foram avaliadas contra um conjunto de validação. Os métodos mais efetivos foram selecionados, combinando os classificadores de postagens e de artigos e ajustando seus parâmetros para obter um bom resultado final.

4.1.5 Estudo dos resultados obtidos no conjunto de validação para diferentes features e parâmetros

Foi realizado um estudo detalhado dos resultados obtidos pelos classificadores para diferentes features, parâmetros e técnicas de classificação (NB e NB com pesos). A avaliação final da rede foi realizada em um conjunto de teste separado.

Separou-se a base de dados total em dois conjuntos. Um de treinamento e um de validação de forma aleatória, sendo 25% para validação e 75% para treinamento. Para a análise da qualidade do classificador realizou-se a média das estatísticas avaliadas para varias divisões aleatórias diferentes de modo a obter um resultado estável.

4.1.6 Definição da arquitetura do classificador e sua implementação

Definiu-se que a rede seria treinada e armazenada em um servidor central. Considerou-se também a possibilidade da rede ser dinâmica (ou seja, se a interação com o usuário fazer com que a rede aprenda online com as novas informações).

4.1.7 Desenvolvimento do plugin

Por fim desenvolveu-se o plugin para o Google Chrome, com uma interface gráfica amigável para o usuário para que ele entenda de forma fácil como filtrar as postagens em seu feed de notícias.

4.1.8 Testes de usabilidade

Por fim foram realizados testes de usabilidade com colegas de turma baseados nas heurísticas de Nielsen.

4.2 Aquisição de dados

Como já foi dito, foram desenvolvidas duas formas diferente de realizar a aquisição de dados. Uma delas consiste numa extensão para o Google Chrome que possibilita o usuário classificar cada postagens à qual ele se depara no Facebook,

enviando os dados e a classificação da mesma para o banco de dados. A outra forma de aquisição de dados é o desenvolvimento de um crawler que faz o download diversos artigos online.

4.2.1 Extensão do Chrome de classificação de postagens

4.2.1.1 Funcionamento de extensões do Chrome

Extensões são softwares que melhoram as funcionalidades de um navegador. O Google Chrome disponibiliza um Framework de desenvolvimento de extensões com muitas capacidades (5).

A extensão consiste em um conjuntos de arquivos zipados que incluem HTML, Javascript, CSS, imagens, etc. O Javascript de uma extensão pode ser dividido em 3 partes diferentes: código de extensão (*extension code*), scripts de conteúdo (*content scripts*) e scripts injetados (*injected scripts*). Estes três modos foram descritos abaixo.

- **Código de extensão:** Trata-se do código injetado diretamente no browser, tendo portanto acesso a todas as funcionalidades da API do Chrome, como tabs de background, pop-ups do navegador (aqueles pequenos ícones das extensões que ficam no canto superior direito do chrome), etc.
- **Scripts de conteúdo:** Trata-se de um código que é executado quando uma determinada página é carregada pelo usuário. Este script possui um escopo entre o do código de extensão e o do script injetado. Os scripts de conteúdo têm acesso à algumas das funcionalidades da API do Chrome e ao mesmo tempo pode acessar e modificar o DOM da página. Por estar em um escopo diferente ao escopo do javascript da própria página ele não tem acesso às funções e objetos definidos no mesmo. Por outro lado, ele não possui diversas das restrições de segurança que scripts injetados possuem. O script de conteúdo pode, por exemplo, executar cross-origin requests (ou seja, acessar servidores de outra origem).
- **Scripts injetados:** Scripts injetados são, como o nome diz, pedaços de código javascript que são injetados numa determinada página, executando em seu escopo. Desta forma eles tem acesso à todas as funções e variáveis definidos pelo javascript original da página. Também podem modificar como eles

quiserem estas variáveis e o próprio DOM.

No caso da aplicação deste trabalho, é necessário ser capaz de mandar os dados das postagens com suas classificações para um servidor central (que obviamente não é do próprio Facebook), logo scripts injetados não são uma boa solução (uma vez que o javascript do Facebook impede a execução de cross-origin requests). Ou seja, a extensão foi desenvolvida predominantemente com scripts de conteúdo.

Uma observação importante é que scripts de conteúdo ainda possuem algumas restrições de segurança ao fazer requests para outros domínios. Se o site principal for https, o script de conteúdo só poderá realizar requests para outros servidores por https também. Para tal, o servidor desenvolvido neste trabalho deve ser capaz de responder requests https.

4.2.1.2 Manifesto da extensão

As extensões do Chrome possuem um arquivo de configuração chamado de manifesto. Este arquivo encontra-se no formato de JSON.

Listagem 4.1: Manifesto da extensão do Chrome para coleta de dados

```
{
  "manifest_version": 2,
  "name": "Demeter",
  "version": "1.0.0",
  "description": "Collect data from facebook to use in NLP studies.
     This plugin has academic purposes.",

  "icons": {
    "128" : "icon_128.png",
    "180" : "icon_180.png"
  },

  "content_scripts": [{
    "matches": [ "https://*.facebook.com/*" ],
    "js": [
      "jquery-1.11.3.min.js",
      "poo_utilities.js",
```

```
      "facebook_tree.js",
      "demeter_dom.js",
      "story_classification.js",
      "contentscript.js"
    ],
    "css": [ "demeter.css" ]
  }],

  "permissions": [ "tabs", "https://*.facebook.com/*", "https://
     demeter-1075.appspot.com/*" ],

  "web_accessible_resources": [
    "contentscript.js",
    "jquery-1.11.3.min.js",
    "poo_utilities.js",
    "facebook_tree.js",
    "story_classification.js",
    "demeter_dom.js",
    "demeter.css",
    "three-dots.png"
  ]
}
```

O código 4.1 ilustra o manifesto da extensão desenvolvida. Ele identifica o nome da extensão (Demeter), a sua versão, os ícones utilizados, os scripts de conteúdo e as permissões (acessar o facebook e o servidor desenvolvido).

Note que além dos códigos desenvolvidos, utilizou-se a biblioteca do JQuery para facilitar a manipulação do DOM.

4.2.1.3 Estrutura do DOM do Facebook

Para injetar um pedaço de HTML no meio do Feed de notícias do Facebook, é importante entender a sua estrutura, básica.

Este projeto foi feito assumindo que o Facebook não iria realizar grandes mudanças em seu design e em sua estrutura básica de HTML em um curto prazo.

Caso houvesse tal modificação, a extensão desenvolvida iria parar de funcionar, sendo necessário realizar algumas adaptações para que ela fosse consertada. Todo código foi desenvolvido de forma modularizada de modo a tentar diminuir as dificuldades de adaptação caso este evento infortúnio ocorresse. Todavia, desde o começo até o final do desenvolvimento do projeto isto não aconteceu.

O HTML do Facebook passa por um processo de ofuscação e compressão antes de ser enviado para as máquinas cliente. Essa ofuscação troca as classes e ids dos elementos por nomes aleatórios e curtos. Então um elemento HTML que originalmente tinha uma classe 'facebook_feed', por exemplo, passará a ter a classe '_u_s8v4'. Este processo atrapalha um pouco no desenvolvimento de uma extensão que se acople ao site do Facebook (pois esses ids e classes são aleatórios e podem mudar). Entretanto, por algum motivo (provavelmente se trata de um código antigo), algumas classes e ids continuam com nomes legíveis. Considerou-se que estes nomes legíveis são mais estáveis e portanto, baseou-se a extensão desenvolvida em elementos HTML que possuíam estes nomes.

Figura 5: Exemplo de HTML do Facebook, com algumas classes aleatórias e algumas de nome legível

A Figura 5 ilustra o que foi explicado no parágrafo anterior. Algumas das classes são strings aleatórias('_5pcr', '_3ccb', '_1dwg', etc), enquanto que algumas possuem valores legíveis ('userContentWrapper').

Observando estes elementos nomeados, chegou-se à seguinte estrutura simplificada para o DOM do Facebook. Todo o conteúdo do site, exceto a barra azul superior e o chat que fica ao lado direito, se encontra dentro de um div chamado de mainContainer. Dentro deste mainContainer há um div chamado de feed_stream, que possui todas as postagens do feed de notícias. O feed_stream contem um ou mais substreams. Cada substream é um conjunto de postagens. Quando o usuário desce a página até a parte inferior, um novo substream é criado com as novas postagens. Cada substream possui uma ou mais postagens que são representadas por divs chamados de userContentWrapper. É possível que userContentWrapper's

contenham outros userContentWrapper's (por exemplo quando uma pessoa compartilha a postagem de outra pessoa). A Figura 6 mostra um exemplo de uma árvore de DOM simplificada para o Facebook.

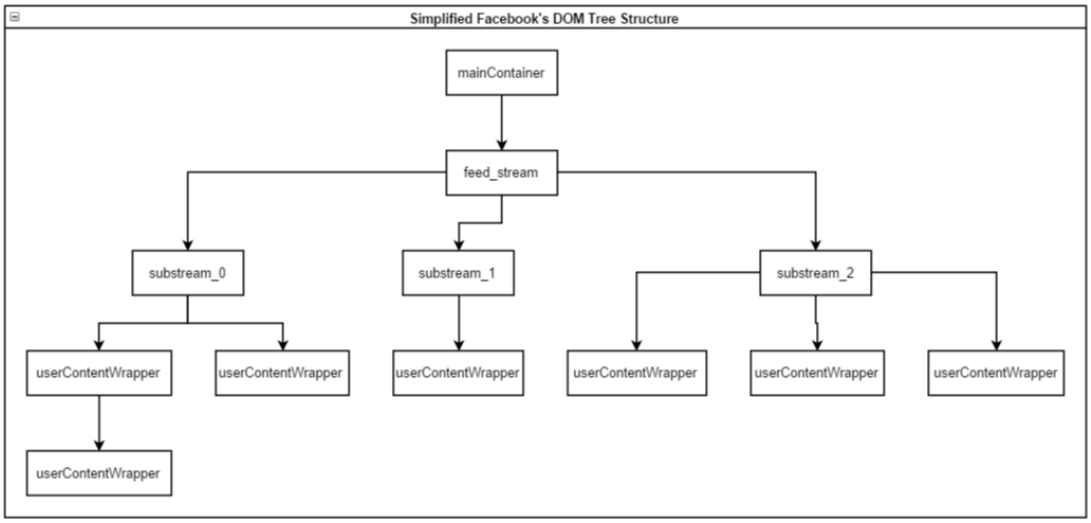

Figura 6: Exemplo ilustrativo da estrutura básica da árvore de DOM do Facebook

4.2.1.4 UI desenvolvida

A extensão desenvolvida utiliza a API de javascript JQuery para injetar um pedaço de HTML logo acima do userContentWrapper, de modo a colocar um cabeçalho no topo de cada postagem com as classes pré definidas. O usuário deverá agir como um supervisor para o classificador indicando qual o assunto da postagem em questão. A Figura 7 ilustra o cabeçalho em uma postagem.

Uma vez que o usuário clique na classe à qual a postagem pertence, o cabeçalho passa a conter apenas o nome da classe escolhida e uma barra opções a direita que pode ser expandida, conforme a Figura 8

A barra de opções possui duas possíveis ações: 'Mudar Assunto' ou 'Adicionar Assunto'. Se o usuário clicar na primeira opção ele poderá escolher um novo assunto para a postagem. Se o usuário clicar na segunda opção ele poderá adicionar um novo assunto para a postagem (que então possuirá duas classes). A Figura 9 ilustra a barra de opções.

Observe que apesar de na hora de realizar a classificação foi feita a simplificação de que o problema se trata apenas de *multi-class* (não *multi-class / multi-label*), na hora da coleta de dados é possível classificar uma única postagem em vários assuntos diferentes. Isto foi feito primordialmente por dois motivos. Primeiramente,

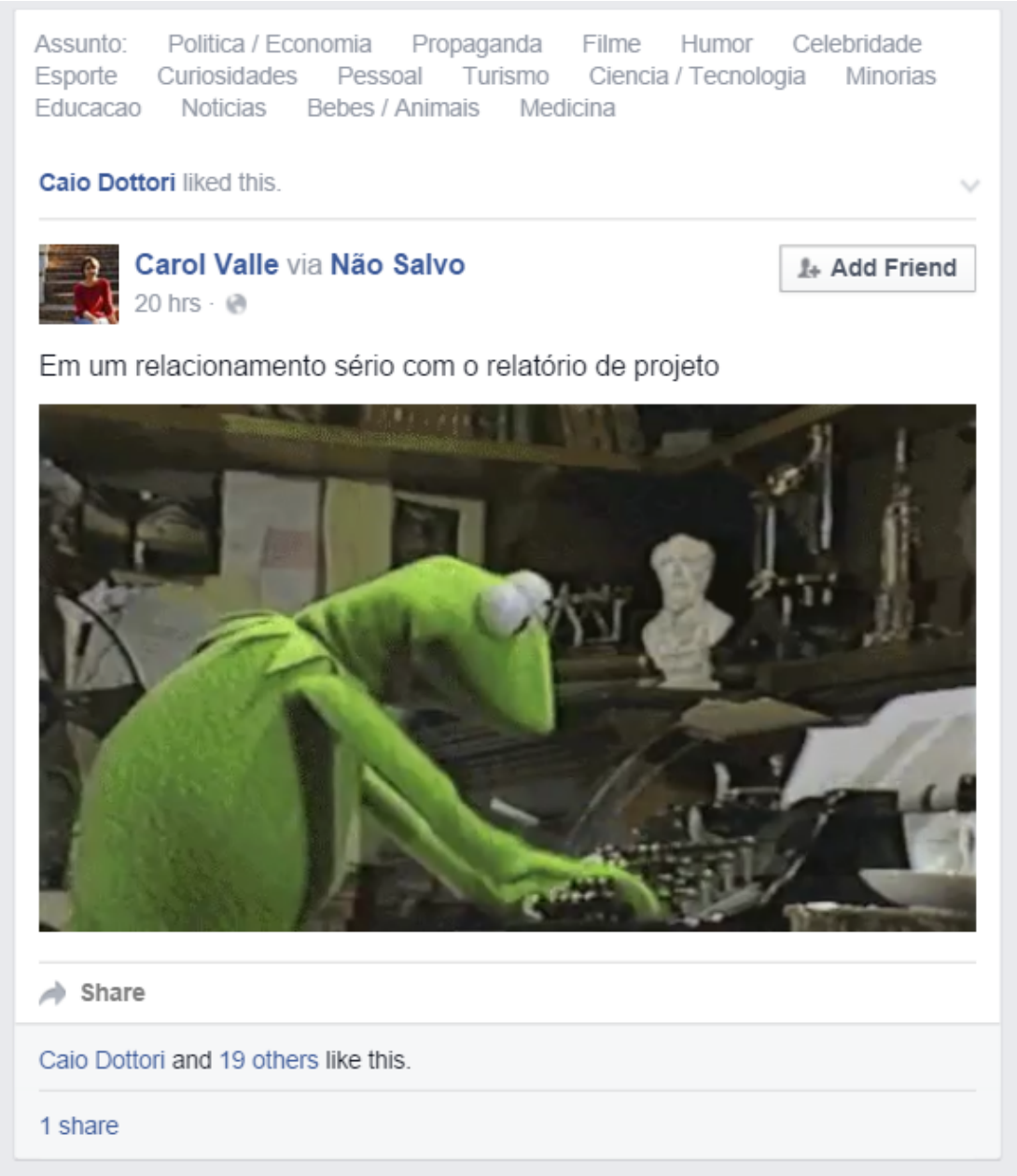

Figura 7: Exemplo em uma postagem do cabeçalho contendo as possíveis classes

é interessante já possuir uma base de dados com postagens com classificações múltiplas para se poder estudar o classificador *multi-class / multi-label* em trabalhos futuros. Além disto, vários usuários diferentes podem classificar as postagens de forma distinta (não concordam entre si). Neste caso, o servidor guarda a contagem de classificações para cada classe e é considerado que a postagem pertence à classe mais frequente (empates são quebrados manualmente).

Figura 8: Exemplo da UI de uma postagem classificada pelo usuário

4.2.1.5 Servidor

Sempre que um usuário seleciona uma classe para uma determinada postagem, é realizada uma requisição POST para o servidor do Demeter (nome do projeto), contendo as informações relevantes. O servidor salva estas informações em um banco de dados.

O banco de dados contem postagens indexadas por sua url (que é um valor único para cada postagem) e é associado a um conjunto de classes e frequências. Por exemplo, se um post foi classificado 5 vezes como Política e 2 como educação, essa informação ficará registrada no banco de dados.

Figura 9: Barra de opções para modificar a escolha do assunto na extensão do Chrome

É importante notar que não é apenas o texto da postagem que é salva no banco de dados. Também armazena-se o usuário que realizou criou a postagem, o momento em que ela foi publicada (timestamp), a presença de fotos ou vídeos, a existência de informações sobre o local de onde o usuário postou, etc. Essas informações podem ser relevantes na hora de se criar features para a NB.

4.2.1.6 Crawler

4.3 Processamento do texto

Um bom pré processamento do texto é essencial para um bom resultado de classificação. Isso é especificamente válido em um ambiente como uma rede social, onde muitas vezes há abreviações, interjeições, utilização de linguagem informal, etc.

4.3.1 Tokenização

Muitas vezes as palavras puras não são muito boas para utilizar como features na classificação. Por isso para cada palavra associa-se um token. Este processo é chamado de tokenização. Deve-se decidir o que será feito com a pontuação, se palavras serão modificadas, etc. As vezes um token pode ser um conjunto de palavras. Por exemplo 'Rio de Janeiro' pode ser um único token.

Outros exemplos de tokenização são trocar todos os números por um único token (normalmente para um classificador não é tão importante o valor numérico, mas sim a presença de um número em si). O mesmo vale para datas, porcentagens,

links, etc.

4.3.2 Normalização

Normalização é o processo de se criar classes de equivalência de palavras. Por exemplo: as palavras U.S.A e USA são a mesma, porém escritas de forma diferente. Alem disso palavras possuem a primeira letra maiúscula quando iniciam uma frase ou podem estar completamente escritas em caixa alta (se o escritor quer passar a noção de que está gritando, por exemplo). Transformar todas as letras para caixa baixa é um tipo de normalização.

4.3.3 Stemming

Stemming é o processo de trocar todas as palavras que possuem sentidos parecidos por um mesmo radical. Por exemplo, os verbos escrevo, escrevi, escrevemos, escreverei, escrevera e escreveu podem ser substituídos pelo radicar escrev.

4.3.4 Processamento utilizado

Para este trabalho foram feitas as seguintes etapas de pré processamento do texto.

- **Remover caracteres unicode:** Acentos, cedilhas, tremas e outros caracteres unicode são substituídos por seus equivalentes em ASCII.
- **Remover letras maiúscula:** Todas as letras maiúsculas são mudadas para minúscula.
- **Remoção números:** Quando há um número no texto, seu valor exato não importa muito para o classificador. O que importa é a sua presença. O mesmo vale para datas, porcentagens, urls, valores de dinheiro e datas. No caso, os números são substituídos por ‘{number}’, as datas por ‘{date}’ e assim por diante.
- **Remover a pontuação:** Toda a pontuação é removida.
- **Encontrar risadas:** Todas as ocorrências de risadas (que puderam ser identificadas com algumas heurísticas simples) são substituídas por ‘laughter’

- **Remover letras duplas:** Em redes sociais é muito comum o usuário repetir a mesma letras diversas vezes para enfatizar a palavra. Por exemplo: 'Que FOOOOOOOOOOFO!'. Isto atrapalharia o classificador. Por isso letras repetidas são removidas. Note que, apesar de ser permitido no português 'r' e 's' duplos, removê-los não atrapalhará muito o classificador.
- **Filtrar palavras comuns e que não agregam muito valor ao classificador:** Palavras comuns do português como artigos, preposições, etc, são desconsideradas.

4.4 Tecnologias utilizadas

4.4.1 Extensão para Chrome

4.4.2 HTML / Javascript / CSS

4.4.3 Git

4.4.4 Google App Engine

4.4.5 Python

5 Propostas de Classificadores

Foram propostas diversas formas diferentes de classificadores, e cada uma será posteriormente avaliada e comparada.

5.1 Classes adotadas

Foram consideradas as seguintes classes para todos os classificadores desenvolvidos:

- Politica / Economia
- Propaganda
- Filmes / Series
- Humor
- Celebridade
- Esporte
- Pessoal
- Turismo
- Ciencia / Tecnologia / Meio Ambiente
- Minorias
- Educacao
- Noticias
- Bebes / Animais
- Saude

5.2 Naïve Bayes utilizando apenas o texto

O classificador mais simples que pode ser criado utilizando NB, consiste em utilizar apenas as palavras do texto como features.

Neste caso, realizou-se primeiramente o processamento conforme o descrito na Seção 4.3, normalizando as palavras de forma apropriada. Durante o processo de normalização, foram realizadas as seguintes modificações.

- Todos os links foram trocados pelo token '{link}'
- Todas as referências feitas com @ (por exemplo @NomeDaPessoa) foram trocadas pelo token '{tag}'
- Todos os valores monetários (tanto em real como em dólar) foram trocados pelo token 'money'
- Todas as percentagens foram trocadas pelo token '{percentage}'
- Todas as datas foram trocadas pelo token '{date}'
- Todos os outros números foram trocados pelo token '{number}'

Em seguida, realizou-se o treinamento utilizando base de dados obtida, realizando-se a contagem da ocorrência dos tokens em cada uma das classes. Finalmente testou-se a performance do classificador obtido em um conjunto de testes.

5.3 Naïve Bayes com Features Adicionais

Várias outras features foram adicionadas na tentativa de melhorar a performance do classificador. As postagens em redes sociais em geral contem muito mais informações do que o texto escrito. Tentaram-se capturá-las utilizando as seguintes features:

- Usuário ou página que é autor(a) da postagem
- Tipo de compartilhamento (se é apenas um status update, ou se é um vídeo ou uma foto que estão sendo compartilhados)

- Se a postagem possui uma localização de origem (o valor do local em si de onde a postagem foi feita não foi considerado, pois seria necessária uma base de dados muito grande para que este local fizesse diferença)
- Se há algum outro usuário marcado na postagem (considerou-se esta feature interessante pelo fato de que postagens pessoas costumam ter pessoas marcadas).
- Se a postagem foi promovida por meio de pagamentos (tem a tag Sponsor no Facebook) ou não
- O tamanho do texto (discretizou-se o tamanho em 4 categorias: tiny, small, medium, large)

É importante notar que o ideal seria recuperar informações a partir das imagens e vídeos compartilhados, entretanto este problema foge ao escopo deste trabalho.

5.4 Weighted Naïve Bayes utilizando apenas o texto

Utilizaram-se as Weighted Naïve Bayes descritas na Seção 3.4 para ponderar a importância das features relevando a noção independência condicional das NBs.

5.5 Weighted Naïve Bayes com Features Adicionais

Repetiu-se a mesma ideia exposta na Seção 5.3, porem desta vez com a NB ponderada.

5.6 Utilização dos links

Até agora, o conteúdo dos links compartilhados tinha sido completamente ignorado. Por exemplo, se alguém postasse um link sobre economia e escrevesse algo do tipo ‘Muito interessante esta análise’, o classificador iria tentar classificar a potagem utilizando apenas o texto (o que é bem complicado). Para evitar este problema, fez-se um HTTP GET request para o link, extraiu-se o texto principal e ele foi classificado por um segundo classificador. O resultado deste segundo classificador passa a ser uma feature do classificador de postagens.

Este segundo classificador (que será referenciado como classificador de artigos) foi treinado a partir da base de dados adquirida pelo crawler.

5.7 Concatenação do texto dos links

5.8 Fusão de classes pequenas

6 Validação - Experimentos, resultados e análise

6.1 Base de dados

A base de dados para a classificação de postagens foi adquirida a partir da extensão do Chrome descrita na Seção 4.2.1. A classificação foi feita manualmente por diversos supervisores, classificando cada postagem nas diferentes categorias explicitadas na Seção 5.1.

O gráfico da Figura 10 ilustra a proporção entre as diferentes classes na base de dados obtida.

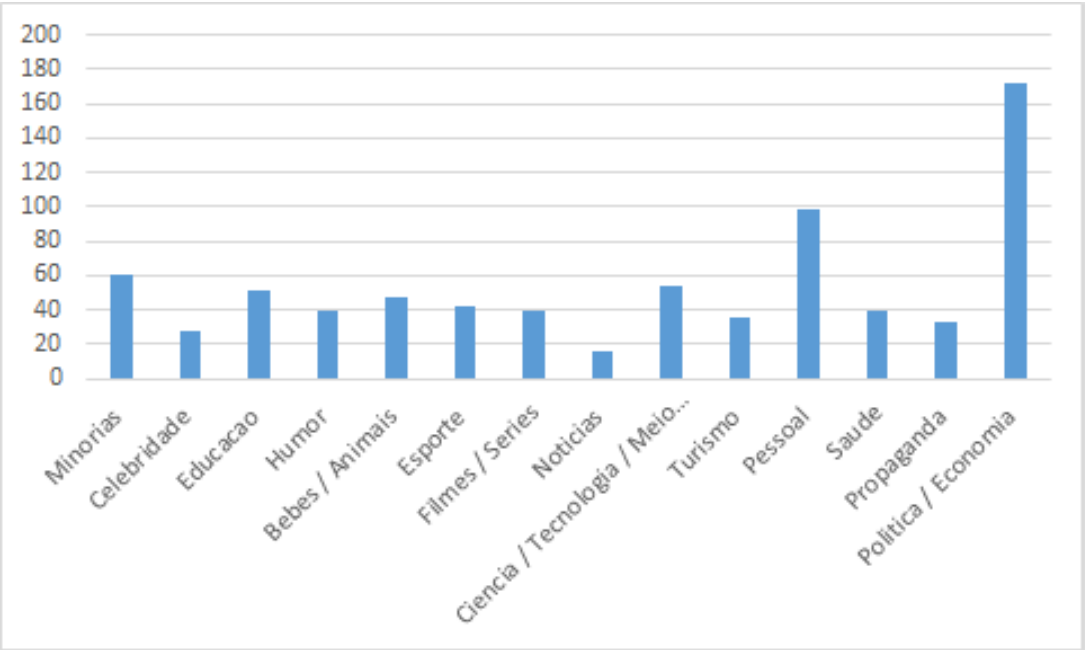

Figura 10: Histograma representativo da base de dados de postagens adquirida

A menor base é de ‘Notícias’, com apenas 16 postagens, e a maior é ‘Política / Economia’ com 172 postagens. O total de postagens é 757. Observa-se que este não é um número muito grande para uma base de dados, ainda mais com um total

de 14 classes, mas foi o que foi possível de se adquirir.

6.2 Naïve Bayes utilizando apenas o texto

Nesta primeira abordagem, o classificador obteve uma acurácia média (ao longo de 100 partições aleatórias diferentes da base de dados em treinamento e validação, conforme o explicado na Seção 4.1.5) de 49.7%, ou seja, o classificador acerta basicamente 1 a cada 2 postagens. Como já foi dito na Seção 3.5.1.1, a acurácia é uma métrica bem ruim para avaliar um classifiador. Desta forma, foram consideradas as outras estatísticas explicadas no capítulo de metodologia, para uma análise mais profunda.

A acurácia esperada para este classificador, utilizando a Equação 3.5, é de 14.16%. O $Kappa$ neste caso foi de 41.4%. Segundo o benchmark de Landis e Koch (9), trata-se de um resultado moderado. Note que a estatística $Kappa$ agrega muito mais informação que uma simples acurácia.

É interessante observar como o Kappa e a acurácia variam conforme se aumenta o tamanho da base de dado. Para tal, repetiu-se o processo de treinamento e validação com a base de dados de tamanho variável (pegando-se subconjuntos aleatórios da base de dados original). Manteve-se sempre uma proporção de 75% pra treinamento e 25% para validação. A Figura 11 mostra o crescimento da acurácia, enquanto que a Figura 12 mostra o crescimento do Kappa.

Como pode ser observado nas figuras 11 e 12, quanto maior a base de dados, melhor o classificador fica. Alem disso, como este crescimento ainda não se estabilizou, fica claro que se a base de dados fosse substancialmente maior (dezenas de milhares de postagens), os resultados seriam muito melhores.

Entretanto, não é viável, para os propósitos deste trabalho, acumular tantos dados. Portanto trabalhou-se com essa quantidade reduzida de postagens, tentando-se introduzir melhoras qualitativas ao classificador com as modificações descritas na Seção 5.

Para se compreender onde exatamente o classificador está ruim, rodou-se novamente o programa com apenas uma iteração e construiu-se a matriz de confusão para os resultados obtidos, como pode ser visto na Tabela 3.

Para este exemplo teve-se:

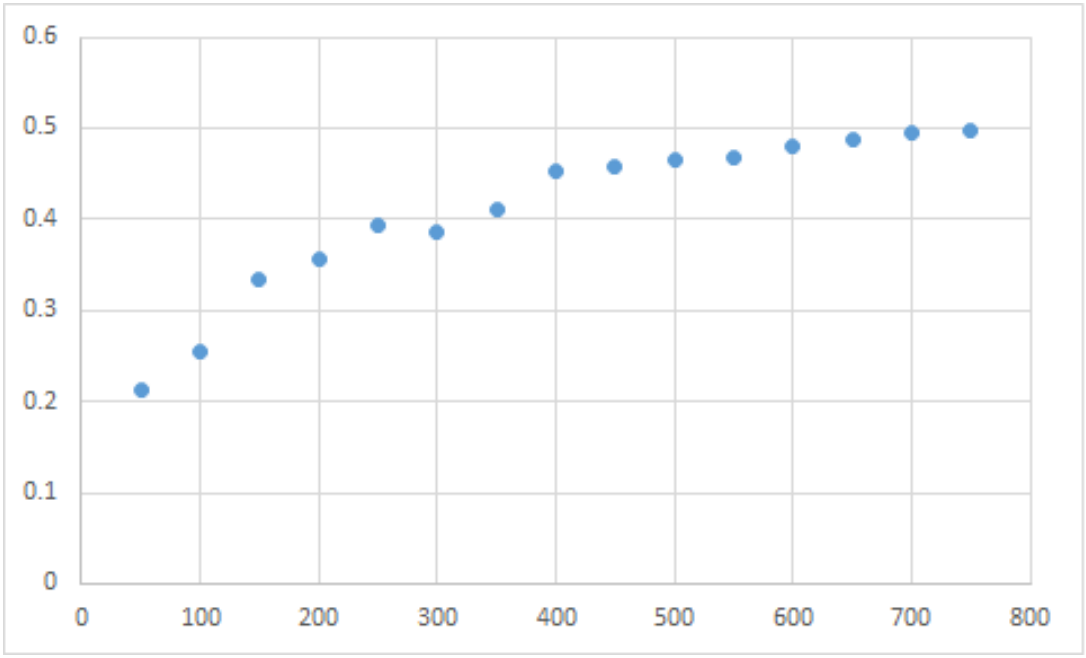

Figura 11: Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para NB simples

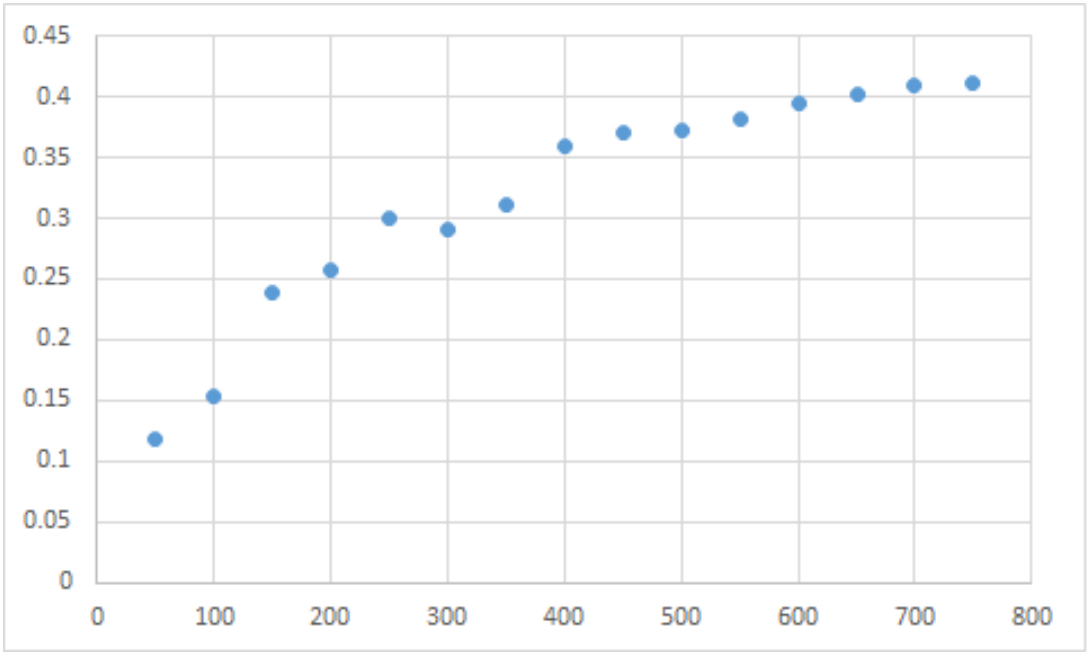

Figura 12: Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para NB simples

$acuracia = 0.529101$

$acuracia\_esperada = 0.138154$

$Kappa = 0.453615$

A Tabela 3 apresenta a matriz de confusão obtida. A Figura 13 consiste numa

representação gráfica da matriz, para facilitar a sua visualização. Nesta representação, quanto mais vermelha a célula, mais próximo seu valor está do 0. O amarelo representa valores maiores ou iguais a 7. Números intermediários possuem cores intermediárias.

Beb	Cel	Cie	Edu	Esp	Fil	Hum	Min	Not	Pes	Pol	Pro	Sau	Tur
7	0	1	0	0	0	0	0	0	1	0	2	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	0
0	1	2	0	0	0	1	0	0	0	1	0	0	0
0	0	2	8	0	1	1	0	0	0	1	0	1	0
0	0	0	1	5	0	0	0	0	0	0	1	0	0
0	0	1	0	0	4	1	0	0	1	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	1	1	0	1	1	9	1	3	2	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	0	1	2	0	0	1	0	21	0	3	0	1
1	1	4	2	2	1	7	4	3	1	38	2	4	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	1	2	1	2
0	0	0	0	0	0	0	0	0	0	0	0	0	3

Tabela 3: Matriz de confusao para a NB apenas com o texto

O Heatmap da Figura 13 e a Tabela 3 deixam claros vários problemas do classificador. Por exemplo, a linha de política está muito amarela como um todo. Isso mostra que o classificador está com alta tendência de escolher política como assunto, mesmo quando isso não está certo. Isto ocorre porque há muito mais postagens de política do que de outros assuntos, então é mais provável que uma nova postagem seja de política. Isso acaba gerando uma alta abrangência, porém baixa precisão para a este assunto, como pode ser visto na tabela 4.

Outras estatísticas analisadas são as precisões, as abrangências e os f1scores para cada uma das classes, conforme relacionado na Tabela 4.

Como pode ser visto na tabela 4, os resultados das precisões e abrangências estão muito ruins. Isso é reiterado pelo Heatmap da Figura 13. As classificações ainda estão muito espalhadas.

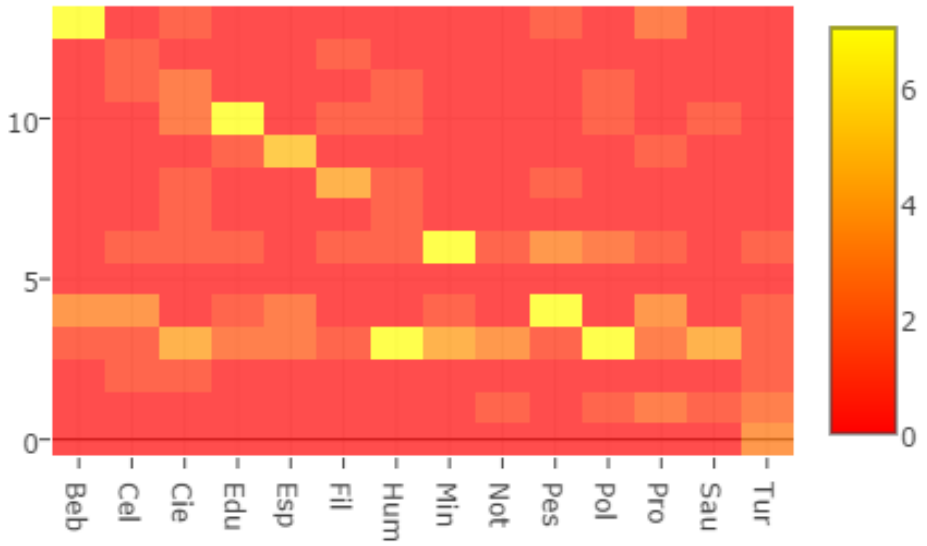

Figura 13: Heatmap da matriz de confusão da Tabela 3

6.3 Naïve Bayes com Features Adicionais

A acurácia média obtida depois da adição das features extras foi de 51.1%. A métrica Kappa foi de 42.5%. Observa-se portanto que houve uma pequena melhora em relação as redes Bayesianas com o texto puro(o Kappa variou de 41.4% para 42.5%).

O motivo da melhoria ser tão pequena é que ocorrem dois efeitos opostos quando se incluem essas novas features. Por um lado, essas features adicionam novas informações que possibilitam que haja uma melhor classificação. Por outro lado, elas são muitas vezes dependentes condicionalmente das palavras do texto (e entre si). Como as NB assumem independência condicional, a adição de features dependentes pode piorar a rede. O resultado da soma desses dois efeitos foi ligeiramente positivo neste caso.

A Figura 14 abaixo mostra a evolução da acurácia das NB com features extras em função do tamanho da base de dados. A Figura 15 mostra a relação do Kappa com o tamanho da base de dados. A Figura 16 compara os gráficos do Kappa para

Classe	Precisao	Abrangencia	F1score
Bebes	0.636364	0.636364	0.636364
Celebridade	0.500000	0.125000	0.200000
Ciencia	0.400000	0.153846	0.222222
Educacao	0.571429	0.615385	0.592593
Esporte	0.714286	0.555556	0.625000
Filmes	0.571429	0.500000	0.533333
Humor	0.500000	0.083333	0.142857
Minorias	0.409091	0.642857	0.500000
Noticias	1.000000	0.000000	1.000000
Pessoal	0.600000	0.777778	0.677419
Politica	0.535211	0.883721	0.666667
Propaganda	0.000000	1.000000	0.000000
Saude	0.142857	0.166667	0.153846
Turismo	1.000000	0.333333	0.500000
Media Micro	0.529101	0.529101	0.529101
Media Macro	0.541476	0.462417	0.498833

Tabela 4: Precisão e abrangencia para NB apenas com o texto

as NB com e sem features extras.

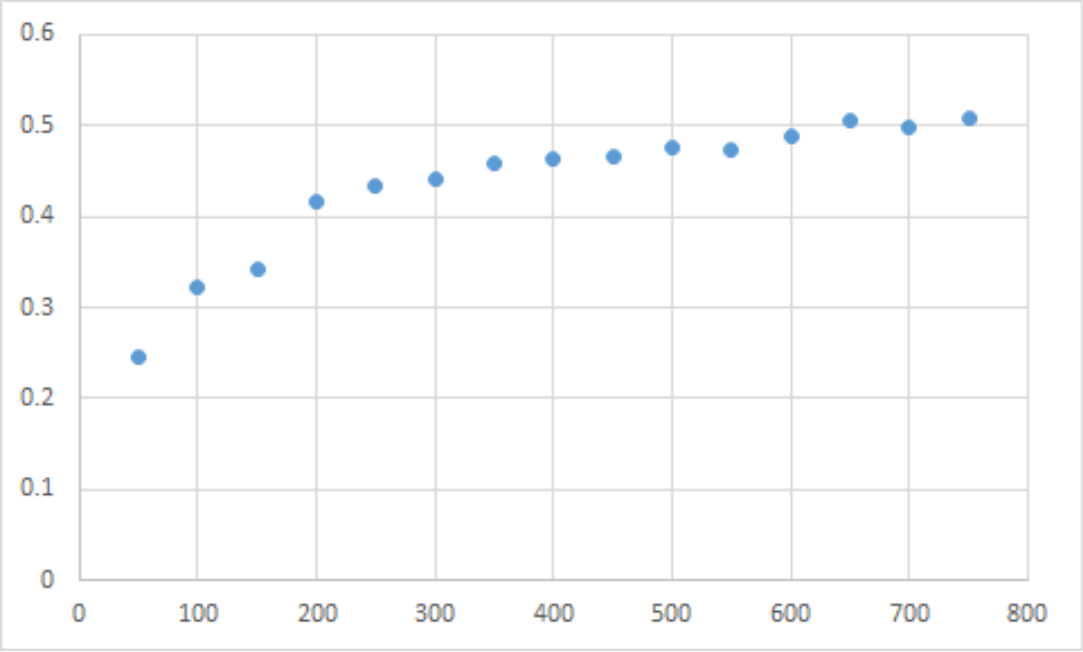

Figura 14: Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para NB com features extras (alem das palavras do texto)

O programa foi executado novamente, desta vez com apenas uma única iteração para se analisa a matriz de confusão gerada. Para este exemplo teve-se:

$acuracia = 0.513228$

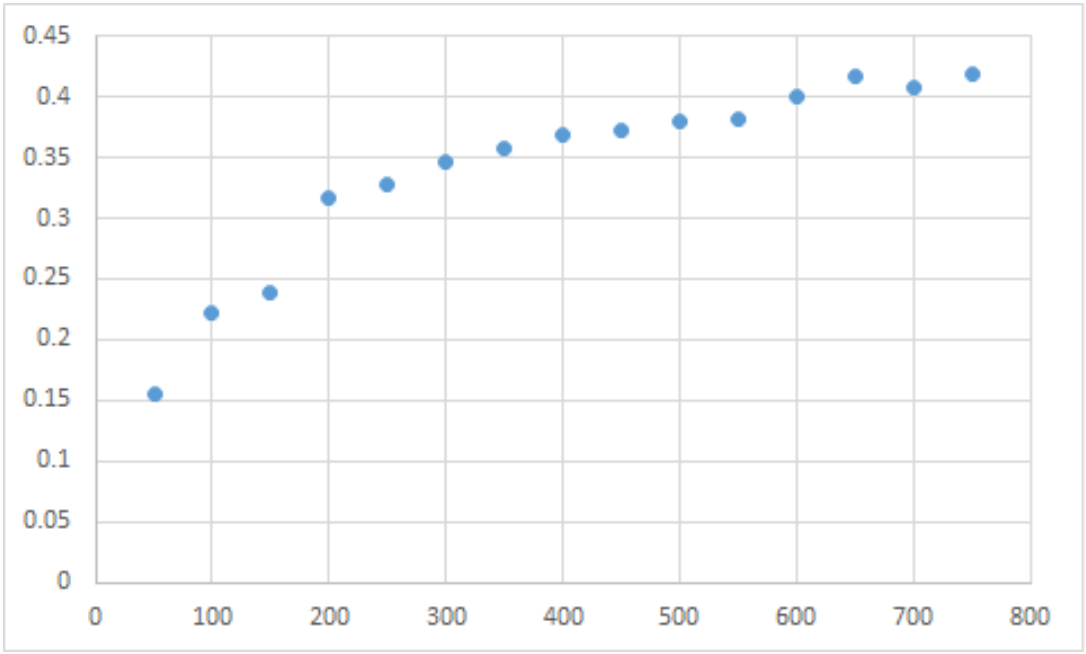

Figura 15: Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para NB com features extras (alem das palavras do texto)

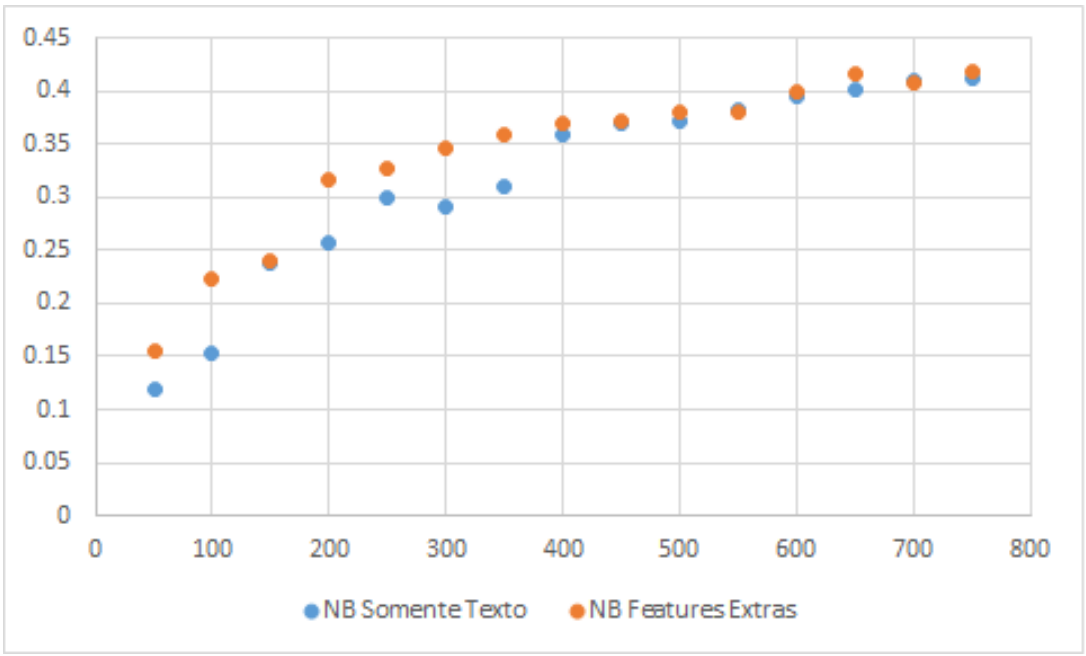

Figura 16: Sobreposição dos gráficos do Kappa em função da quantidade de postagens na base de dados para NB com e sem as features extras

$acuracia\_esperada = 0.140058$

$Kappa = 0.433948$

A Tabela 5 apresenta a matriz de confusão obtida.

A Figura 17 consiste numa representação gráfica da matriz de confusão. Como

Beb	Cel	Cie	Edu	Esp	Fil	Hum	Min	Not	Pes	Pol	Pro	Sau	Tur
6	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	3	0	0	1	0	0	1	0	0	1	1	0
0	0	3	3	1	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	5	0	0	0	0	0	0	0
1	0	0	2	1	1	0	6	2	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	0	3	6	3	4	2	1	22	1	1	1	2
0	2	2	4	1	2	2	5	5	2	39	0	7	0
0	0	1	0	1	3	0	0	0	0	0	3	0	1
0	0	0	0	0	0	0	0	0	0	1	0	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	4

Tabela 5: Matriz de confusao para a NB com features adicionais

pode ser observado neste Heatmap, os resultados ainda são bem ruins, com uma quantidade significativa de erros e alta tendência do classificador de escolher as classes Política ou Pessoal (as duas classes com maior quantidade de postagens na base de dados).

Outras estatísticas analisadas são as precisões, as abrangências e os f1scores para cada uma das classes, conforme relacionado na Tabela 6. Estas estatísticas também estão bem ruins. Na maior parte das classes nem a abrangência nem a precisão foram boas. As classes mais comuns (como Política e Pessoal) tiveram boa abrangência, porém baixa precisão.

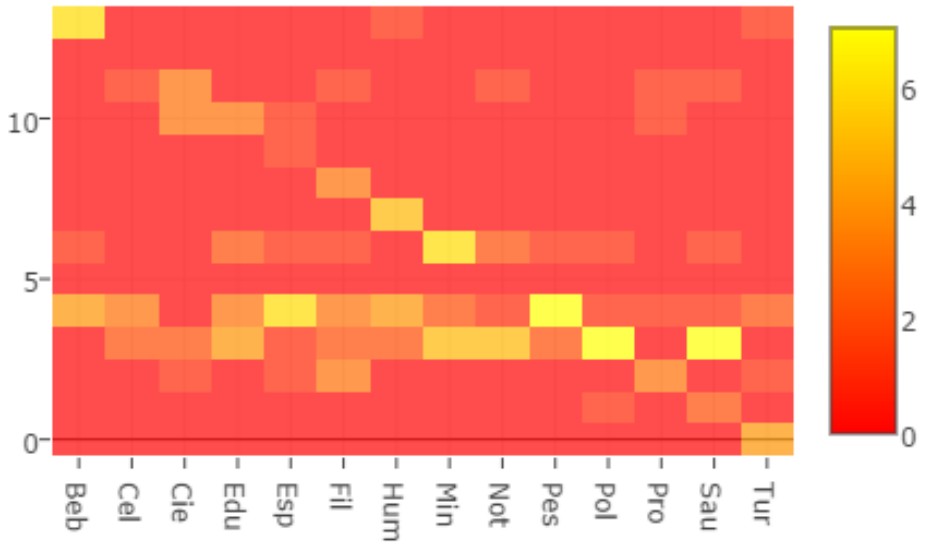

Figura 17: Heatmap da matriz de confusão da Tabela 5

Classe	Precisao	Abrangencia	F1score
Bebes	0.750000	0.545455	0.631579
Celebridade	1.000000	0.000000	0.000000
Ciencia	0.375000	0.333333	0.352941
Educacao	0.375000	0.250000	0.300000
Esporte	1.000000	0.090909	0.166667
Filmes	1.000000	0.230769	0.375000
Humor	1.000000	0.416667	0.588235
Minorias	0.375000	0.461538	0.413793
Noticias	1.000000	0.000000	0.000000
Pessoal	0.415094	0.880000	0.564103
Politica	0.549296	0.928571	0.690265
Propaganda	0.333333	0.500000	0.400000
Saude	0.666667	0.166667	0.266667
Turismo	1.000000	0.500000	0.666667
Media Micro	0.513228	0.513228	0.513228
Media Macro	0.702814	0.378850	0.492317

Tabela 6: Precisão e abrangencia para NB com features adicionais

6.4 Weighted Naïve Bayes utilizando apenas o texto

A acurácia média obtida, depois da adição de pesos nas features e utilizando apenas as palavras do texto, foi de 53.5%. A métrica Kappa foi de 48.2%.

Observa-se portanto que houve uma melhora considerável na métrica Kappa em relação às abordagens anteriores e uma melhora pequena na acurácia. A interpretação para isso é que o classificador está acertando mais de classes diversificadas. Enquanto nos casos anteriores, o classificador se limitava a escolher as classes mais frequentes e errar muito as menos frequentes, agora ele está acertando um pouco de cada classe. Isso faz com que o Kappa melhore mais do que a acurácia.

As Figuras 18 e 18 mostram, respectivamente, a evolução da acurácia e do Kappa em função do tamanho da base de dados. Ambos estão crescendo de forma semelhante, o que reitera a ideia de que quantidade de dados é muito importante independentemente do classificador usado. A Figura 20 compara o Kappa do NB tradicional com o Kappa do Weighted NB pro caso de só se considerar o texto da postagem. Dá pra se observar a melhora obtida. Isso comprova o fato de que é ingênuo se assumir a independência condicional entre as palavras do texto. Ao se ponderar de forma diferente as palavras, aliviou-se essa hipótese, tornando o classificador melhor.

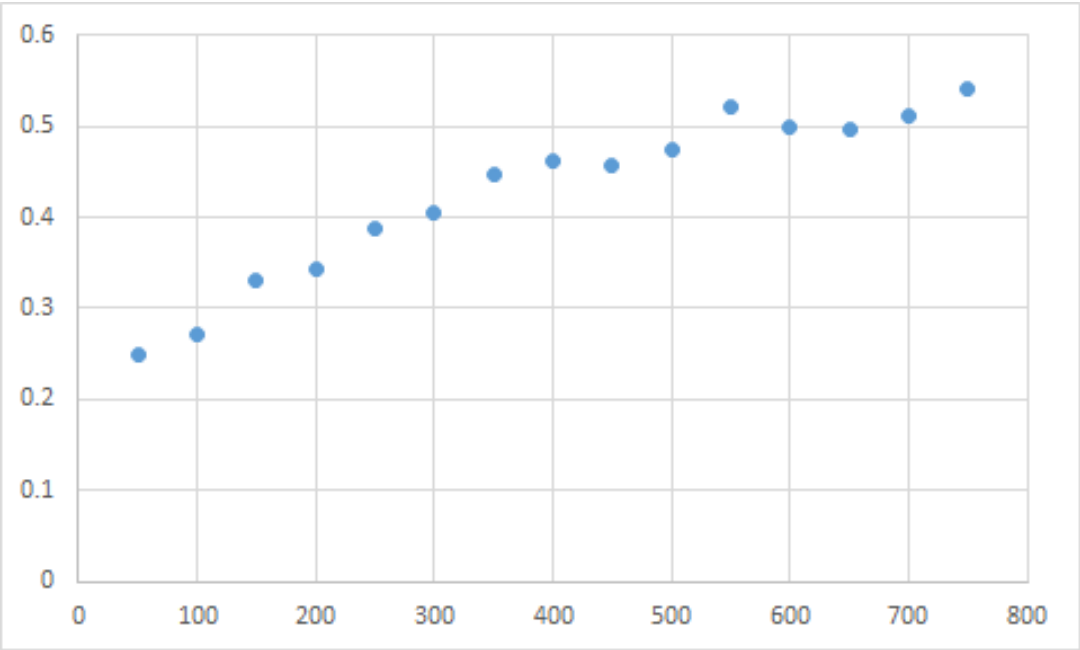

Figura 18: Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB utilizando apenas o texto

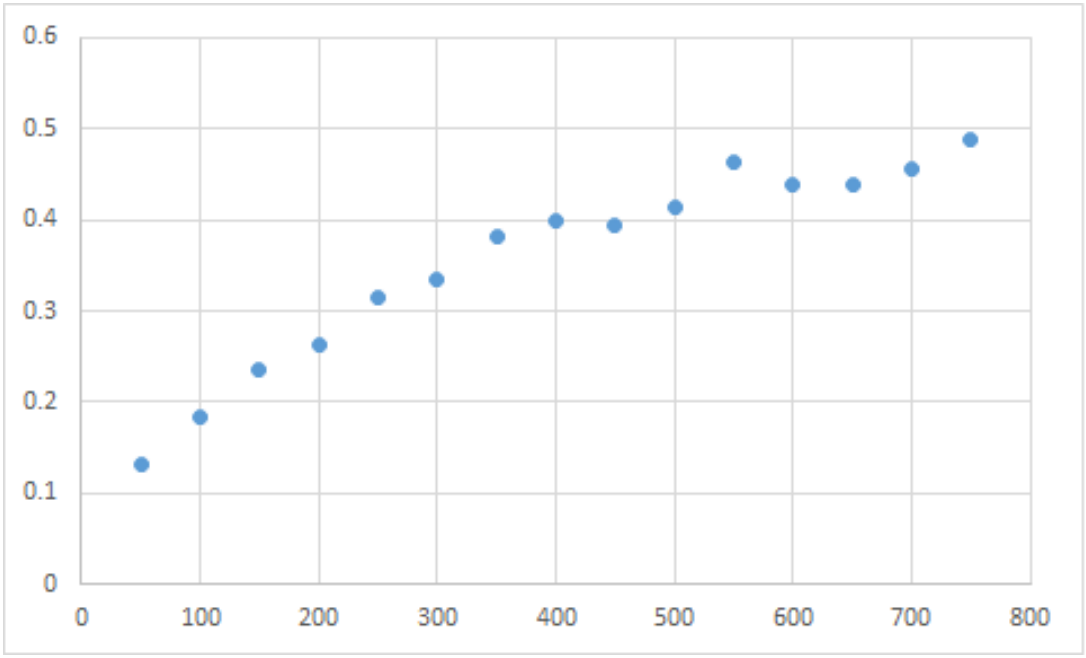

Figura 19: Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB utilizando apenas o texto

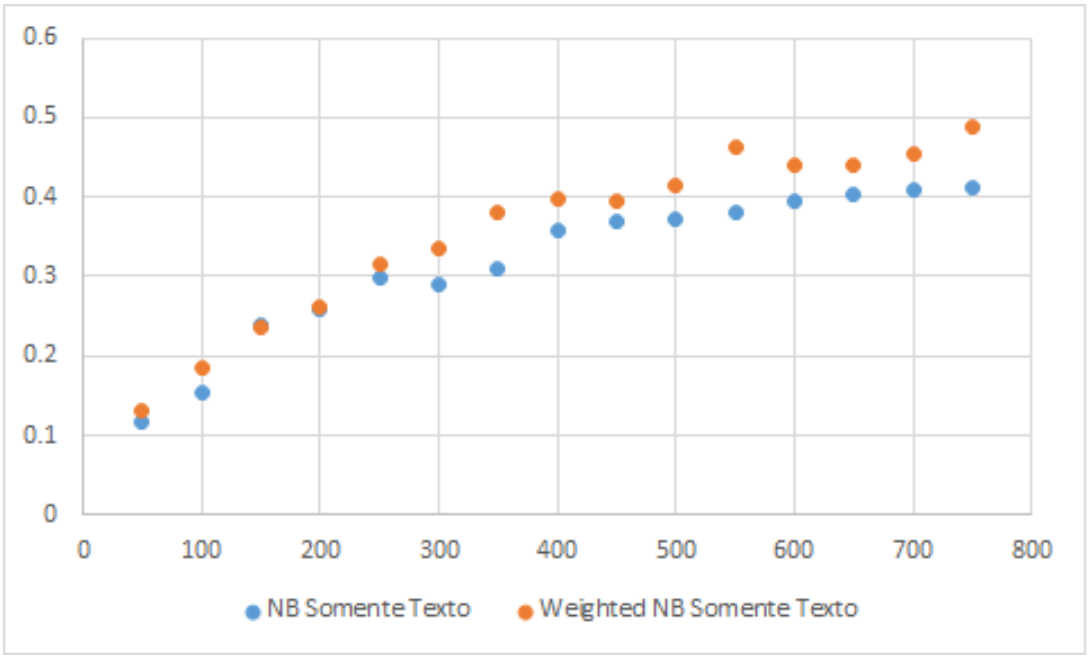

Figura 20: Sobreposição dos gráficos do Kappa em função da quantidade de postagens na base de dados para o NB Simples e o Weighted NB

Para analisar melhor os erros que este classificador está cometendo, rodou-se o mesmo novamente, gerando-se a matriz de confusão. Para este exemplo teve-se:

$acuracia = 0.544974$

$acuracia\_esperada = 0.104000$

$Kappa = 0.492158$

A Tabela 7 apresenta a matriz de confusão obtida. A Figura 21 consiste numa representação gráfica da matriz de confusão.

Beb	Cel	Cie	Edu	Esp	Fil	Hum	Min	Not	Pes	Pol	Pro	Sau	Tur
7	0	1	0	0	1	0	2	0	1	0	1	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	0	6	1	0	0	1	0	0	0	1	0	1	1
1	0	3	8	1	0	0	1	0	0	2	1	2	0
0	0	0	0	8	0	0	0	0	0	2	1	0	0
0	1	0	0	0	12	0	0	0	1	2	1	0	2
0	0	0	0	0	0	2	0	0	0	0	0	0	0
2	1	1	1	0	0	2	9	1	4	12	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	1	1	0	12	0	0	0	1
1	2	0	1	0	1	3	0	0	2	24	0	2	1
0	0	0	1	0	1	1	1	0	0	1	3	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	0
0	0	0	1	1	0	0	0	0	0	1	0	0	5

Tabela 7: Matriz de confusao para a Weighted NB apenas com texto

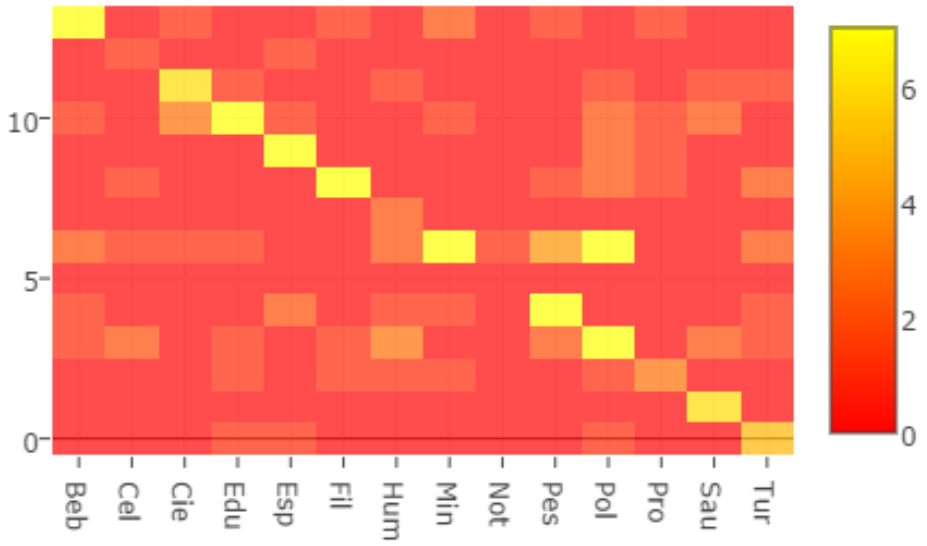

Figura 21: Heatmap da matriz de confusão da Tabela 7

Como pode ser visto na Figura 21, este novo classificador é qualitativamente melhor que os anteriores. O Heatmap possui muito menos células fora da diagonal principal com tonalidades fortes de amarelo. Ele tambem deixa claro o erro mais comum do classificador: considerar como Minorias várias postagens de Política. Este erro é de certa forma aceitável e justificável, podendo ser cometido até por um ser humano. Muitas das postagens de Minorias podem falar sobre medidas públicas a favor ou contra determinadas classes de pessoas, ou falar de partidos e deputados apoiadores (Jean Wyllys do PSOL, por exemplo) e criticar ou defender ideologias políticas. A verdade é que estas postagens deveriam ter sido classificadas com ambas as classes em um classificador *multi-class / multi-label* como explicado na Seção 2.1.3.

Outras estatísticas analisadas são as precisões, as abrangências e os f1scores para cada uma das classes, conforme relacionado na Tabela 8. Apesar de eles ainda estarem ruins, melhoraram em comparação com os classificadores anteriores.

Classe	Precisao	Abrangencia	F1score
Bebes	0.538462	0.583333	0.560000
Celebridade	0.500000	0.200000	0.285714
Ciencia	0.545455	0.545455	0.545455
Educacao	0.421053	0.615385	0.500000
Esporte	0.727273	0.615385	0.666667
Filmes	0.631579	0.800000	0.705882
Humor	1.000000	0.200000	0.333333
Minorias	0.257143	0.642857	0.367347
Noticias	1.000000	0.000000	0.000000
Pessoal	0.666667	0.600000	0.631579
Politica	0.648649	0.533333	0.585366
Propaganda	0.375000	0.428571	0.400000
Saude	1.000000	0.545455	0.705882
Turismo	0.625000	0.416667	0.500000
Media Micro	0.544974	0.544974	0.544974
Media Macro	0.638306	0.480460	0.548248

Tabela 8: Precisão e abrangencia para Weighted NB apenas com texto

6.5 Weighted Naïve Bayes com Features Adicionais

Como será apresentado a seguir, adicionar as features extras no classificador com Weighted NB trás uma melhora considerável no resultado obtido. Isto ocorre

pois as Weighted NB não fazem a consideração de independência e muitas dessas features extras são muito determinantes na classificação. Por exemplo, postagens Pessoais costumam ter pessoas e lugais marcados, certas páginas do Facebook costumam postar sempre sobre o mesmo assunto (partidos políticos postam de Política, páginas de produtos fazem propaganda, páginas de humor sempre postam humor, etc). Todas essas informações introduzidas pelas features extras são muito bem capturadas pelas Weighted NB.

A acurácia média obtida neste caso, depois da adição das features extras nas Weighted NB, foi de 61.0%. A métrica Kappa foi de 56.6%. Este resultado mostra uma melhora considerável em relação a todos os outros classificadores desenvolvidos, conforme pode ser visto na Figura 24. As Figuras 22 e 23 mostram, respectivamente, a acurácia e o Kappa em função do tamanho da base de dados para este classificador.

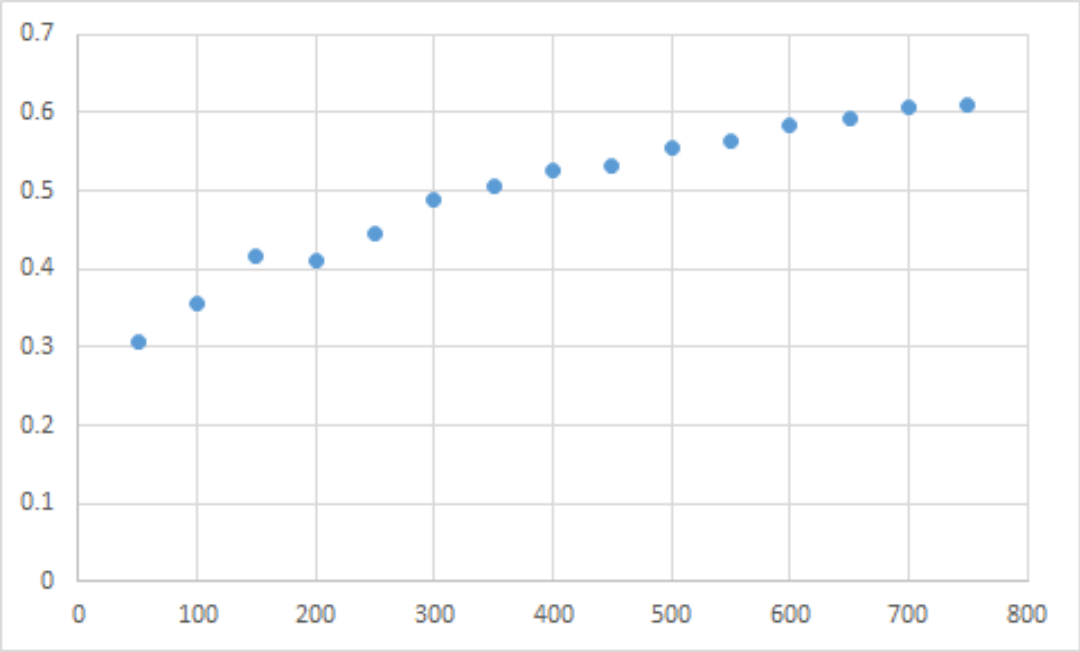

Figura 22: Acurácia em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB com features extras

Analisou-se a matriz de confusão para este classificador, rodando novamente o programa. Para este exemplo teve-se:

$acuracia = 0.656085$

$acuracia\_esperada = 0.108060$

$Kappa = 0.614419$

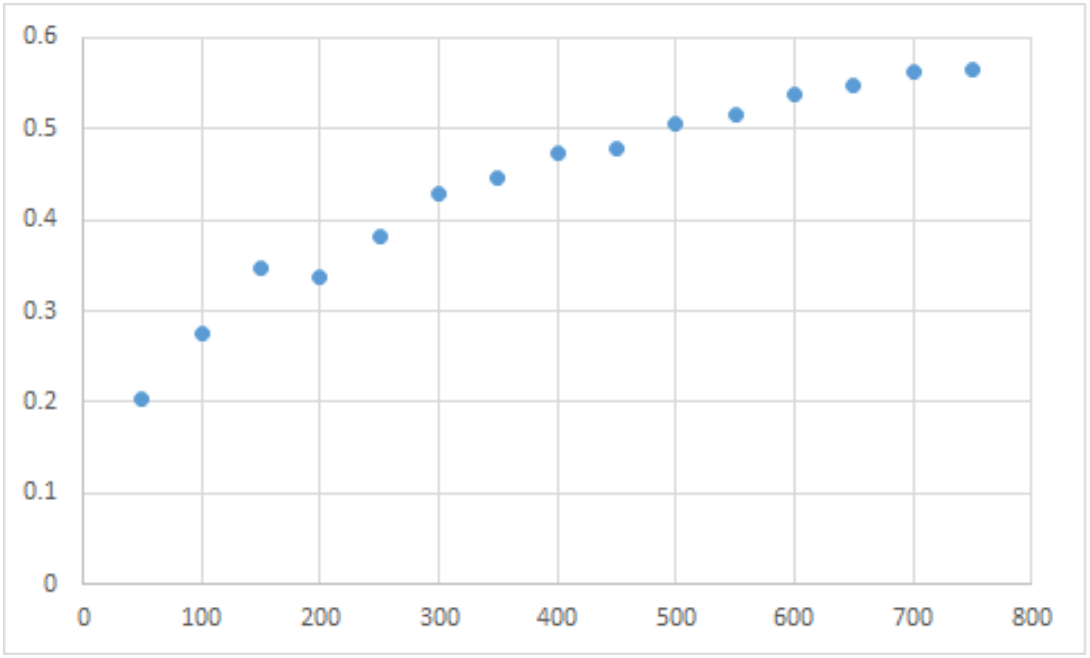

Figura 23: Kappa em função da quantidade de postagens na base de dados (treinamento + validação) para a Weighted NB com features extras

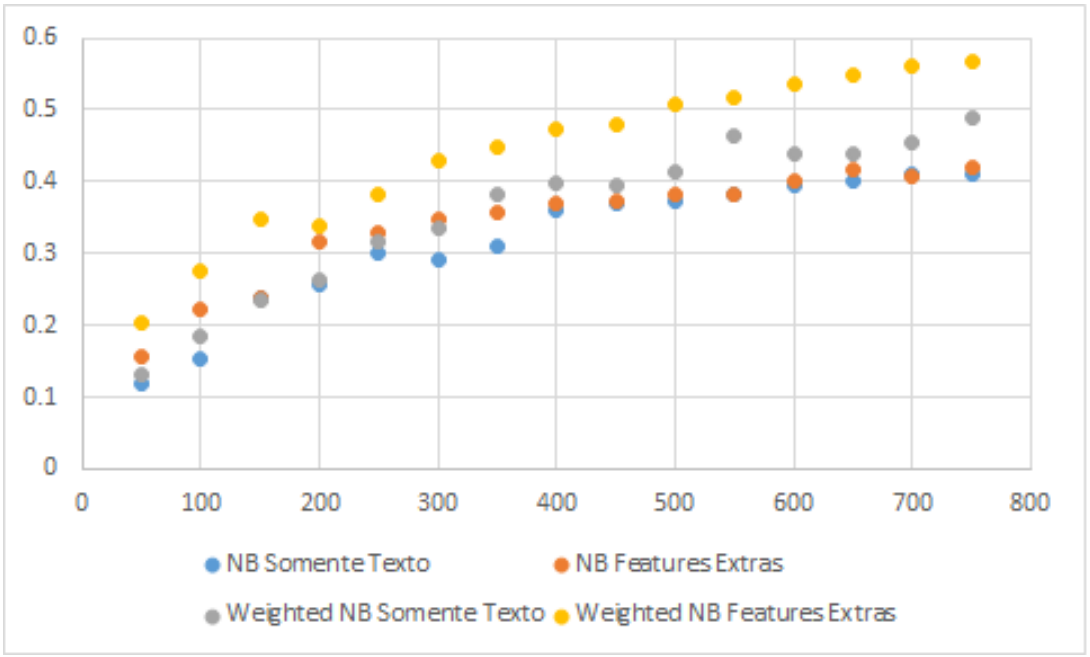

Figura 24: Comparação dos quatro classificadores propostos, mostrando o Kappa de cada um como uma função do tamanho da base de dados.

Uma observação interessante é que com este valor de Kappa, este classificador em específico poderia ser considerado *'Substantial'* pela interpretação do Kappa citada na Seção 3.5.2.5.

A Tabela 9 apresenta a matriz de confusão gerada. A Figura 25 consiste numa

representação gráfica da matriz.

Beb	Cel	Cie	Edu	Esp	Fil	Hum	Min	Not	Pes	Pol	Pro	Sau	Tur
5	1	1	1	0	0	0	0	0	0	0	1	0	0
0	3	0	0	0	0	0	0	0	0	0	0	0	0
0	0	6	0	0	0	0	0	1	1	1	0	0	0
0	0	5	7	1	1	0	3	0	0	4	1	1	0
0	0	1	0	7	0	0	0	0	0	0	0	0	1
0	0	0	1	0	7	0	0	0	0	0	1	1	0
0	0	0	0	0	0	5	0	0	0	0	0	0	0
2	1	0	0	0	0	0	14	2	0	5	0	1	0
0	0	0	0	0	0	0	0	1	0	1	0	0	0
2	1	0	0	0	0	0	1	0	24	1	2	0	2
1	0	0	0	0	1	0	1	0	0	30	0	1	1
0	0	1	1	0	1	0	0	0	1	1	4	0	0
0	0	0	1	0	1	0	0	1	0	2	0	6	0
0	1	0	0	0	0	0	0	0	0	0	0	0	5

Tabela 9: Matriz de confusao para a Weighted NB com features extras

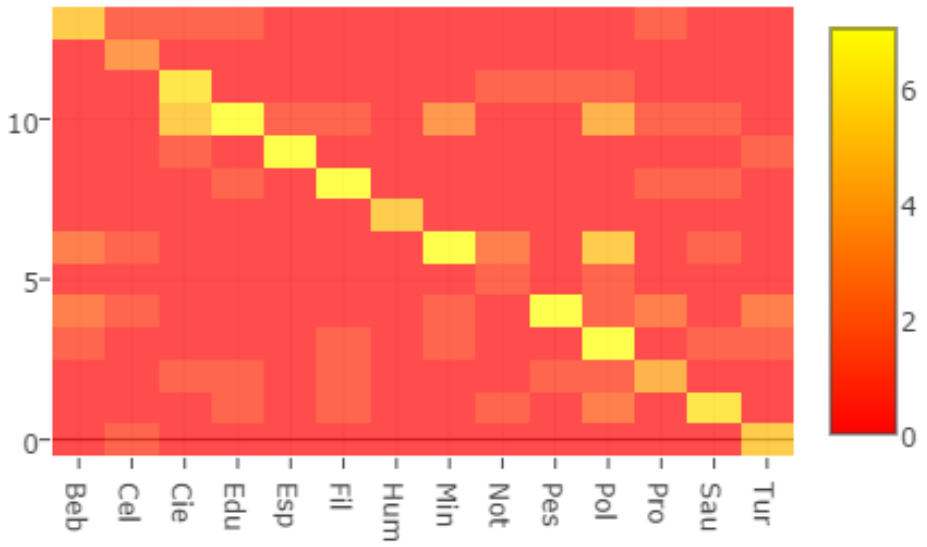

Figura 25: Heatmap da matriz de confusão da Tabela 9

Como pode ser visto na Figura 25 o resultado obtido melhorou consideravelmente. Agora ao Heatmap possui uma diagonal principal com uma tonalidade forte

de amarelo e algumas outras células com alguns erros. A confusão entre Minorias e Política ainda acontece, porém numa intensidade bem menor que anteriormente.

Outras estatísticas analisadas são as precisões, as abrangências e os f1scores para cada uma das classes, conforme relacionado na Tabela 10.

Classe	Precisao	Abrangencia	F1score
Bebes	0.555556	0.500000	0.526316
Celebridade	1.000000	0.428571	0.600000
Ciencia	0.666667	0.428571	0.521739
Educacao	0.304348	0.636364	0.411765
Esporte	0.777778	0.875000	0.823529
Filmes	0.700000	0.636364	0.666667
Humor	1.000000	1.000000	1.000000
Minorias	0.560000	0.736842	0.636364
Noticias	0.500000	0.200000	0.285714
Pessoal	0.727273	0.923077	0.813559
Politica	0.857143	0.666667	0.750000
Propaganda	0.444444	0.444444	0.444444
Saude	0.545455	0.600000	0.571429
Turismo	0.833333	0.555556	0.666667
Media Micro	0.656085	0.656085	0.656085
Media Macro	0.676571	0.616533	0.645158

Tabela 10: Precisão e abrangencia para Weighted NB com features extras

6.6 Utilização dos links

6.7 Concatenação do texto dos links

6.8 Fusão de classes pequenas

6.9 Extensão final desenvolvida

6.10 Teste de usabilidade

7 Conclusões e Trabalhos Futuros

Referências

1 Gianmario Bollano, Donato Ettorre, and Antonio Esiliato. Method and system to improve automated emotional recognition, January 31 2007. US Patent App. 12/449,298.

2 Juan Carlos Fernández Caballero, Francisco José Martínez, César Hervás, and Pedro Antonio Gutiérrez. Sensitivity versus accuracy in multiclass problems using memetic pareto evolutionary neural networks. *Neural Networks, IEEE Transactions on*, 21(5):750–770, 2010.

3 Edwin Chen. What are the advantages of different classification algorithms? `https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms`. [Online; acessado em 08 de novembro de 2015].

4 G. F. Cooper. In *The computational complexity of probabilistic inference using Bayesian belief networks*, pages 393–405, 1990.

5 Google. Chrome extensions - overview. `https://developer.chrome.com/extensions/overview`. [Online; acessado em 11 de novembro de 2015].

6 Dan Jurafsky and Christopher Manning. Practical Issues in Text Classification. `https://class.coursera.org/nlp/lecture`. [Online; acessado em 22 de setembro de 2015].

7 Dan Jurafsky and Christopher Manning. What is Text Classification? `https://class.coursera.org/nlp/lecture`. [Online; acessado em 20 de setembro de 2015].

8 Daphne Koller. Bayesian Network Fundamentals - Semantics And Factorization. `https://class.coursera.org/pgm/lecture`. [Online; acessado em 10 de outubro de 2015].

9 J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

10 Chang-Hwan Lee, Fernando Gutierrez, and Dejing Dou. Calculating Feature Weights in Naive Bayes with Kullback-Leibler Measure. In *Data Mining (ICDM), 2011 IEEE International Conference on Data Mining*, pages 1146–1151. IEEE, 2011.

11 P. Kevin Murphyk. A Brief Introduction to Graphical Models and Bayesian Networks. `http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html`, 1998. [Online; acessado em 04 de novembro de 2015].

12 rbx. Kappa statistic in plain english. http://stats.stackexchange.com/questions/82162/kappa-statistic-in-plain-english, 2013. [Online; acessado em 10 de novembro de 2015].

13 Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

14 Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Introduction to Data Mining. In *Introduction to Data Mining*, pages 145–146, 2006.

15 Anthony J Viera, Joanne M Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363, 2005.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. REGISTRO N°	4. N° DE PÁGINAS
TC			N PAGINAS

5. TÍTULO E SUBTÍTULO:

Desenvolvimento de um classificador de postagens em rede social utilizando Processamento de Linguagem Natural

6. AUTOR(ES):

Luiz Filipe Martins Ramos
Bernardo Monteiro Rufino

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica - ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Inteligência Artificial, Processamento de Linguagem Natural, Redes Bayesianas

9.PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Solicite preenchimento dos campos 2, 3 e 9 – envie este formulário para doc.pt@ita.br

10. APRESENTAÇÃO: **X Nacional** **Internacional**

ITA, São José dos Campos. Curso de Graduação em Engenharia de Computação. Orientador(es): Prof. Dr. Paulo André Lima de Castro. Publicado em 2015

11. RESUMO:

12. GRAU DE SIGILO:

(X) OSTENSIVO **( ) RESERVADO** **( ) SECRETO**