3
Modelagem Textual

> Everything should be made as simple as possible, but not simpler.
>
> Albert Einstein

O primeiro passo em classificação de textos é transformar documentos numa representação adequada para o algoritmo de aprendizado. Adotamos uma representação vetorial para representar os documentos tal que cada elemento do vetor é uma variável que indica uma determinada característica do texto.

O modelo textual pode ser interpretado como uma hipótese do processo de geração de documentos. Tipicamente, tais hipóteses são simplórias e obviamente não refletem o processo de geração de uma opinião uma vez que, dada a complexidade da linguagem natural, é muito difícil modelar matematicamente o processo de criação de um autor. A linha de pesquisa que adota este tipo de abordagem é denominada Intelligent Language Processing (ILP) e tipicamente busca modelar relações semânticas da linguagem. Uma das limitações desta técnica é o fato dos modelos serem fortemente dependentes de idioma e domínio.

Modelos estatísticos sugerem que é possível aprender estruturas complexas de linguagem ao estudar a distribuição estatística presente num *corpus* de texto.

Impulsionados pelo massivo volume de dados disponível para treinamento e um alto poder computacional, técnicas baseadas em aprendizado de máquina vem obtendo bons resultados através de representações textuais simples como saco-de-palavras (BOW[1]).

Nesta dissertação adotamos "representações simples" de documentos e testamos suas eficácias com os classificadores probabilísticos Naive Bayes e Support Vector Machines (SVM).

[1] bag of words

É importante ressaltar que os modelos textuais apresentados neste capítulo consistem apenas em modos de representar documentos e portanto são independentes de classificadores.

3.1 Saco de Palavras

Saco de palavras é o modelo mais utilizado em aplicações de categorização de texto. Neste modelo, cada elemento do vetor de *features* indica se uma determinada palavra ocorre ou não no documento. Propriedades básicas do texto como a ordem em que as palavras ocorrem e dependências condicionais são ignoradas.

Existe uma divergência na literatura quanto ao uso do termo *bag-of-words* (saco de palavras). Em algumas referências, o termo é utilizado para descrever palavras propriamente ditas e em outras o mesmo termo tem uma conotação mais abrangente onde um *saco de palavras* representa um conjunto de eventos tal que a ordem e as dependências condicionais são ignorados. O termo é utilizado em áreas como Biologia Computacional e Visão Computacional com este significado. Adotamos neste trabalho a primeira definição. As matrizes abaixo ilustram o modelo de saco de palavras:

	w_1	w_2	w_3	...	w_n
d_1	1	0	0	...	1
d_2	0	0	1	...	1
d_3	1	1	0	...	0
⋮			⋱		⋮
d_m	0	0	1	...	0

	w_1	w_2	w_3	...	w_n
d_1	12	0	1	...	5
d_2	7	3	2	...	0
d_3	4	15	1	...	0
⋮			⋱		⋮
d_m	0	0	3	...	2

onde w_i representa uma palavra e d_i representa um documento.

3.2 N-gramas

Um N-grama é um conjunto de N palavras consecutivas extraídas de uma frase. O modelo de representação em N-grama tenta amenizar o problema da não-captura da ordem das palavras no modelo em saco de palavras. Neste modelo, as *features* representam a ocorrência de N palavras consecutivas. No modelo saco de palavras, um texto contendo a expressão "grande homem" e outro contendo "homem grande" geram exatamente a

mesma distribuição. Porém, ao adotarmos o modelo 2-grama (bigrama), evitamos que isto ocorra.

3.3 Part of Speech Tagging

Uma mesma palavra pode ter significados diferentes dependendo do contexto em que ela é usada. É comum que sistemas de Mineração de Texto tratem estas ambigüidades para melhorar a qualidade da informação extraída. No caso de *Sentiment Classification* tal tipo de ambigüidade pode gerar um impacto negativo no desempenho do classificador uma vez que dois sentimentos diferentes podem estar mapeados par uma mesma *feature*. Por exemplo, a palavra *love* na frase *"I love this movie"* tem um significado diferente de quando usada na frase *"It's a love story"*. Ao extrairmos a classe gramatical das palavras (adjetivo e substantivo respectivamente) reduzimos este conflito.

Pré-processamos os textos de modo que cada palavra seja "carimbada" com sua classe gramatical (*part of speech*). Por exemplo, a frase "I love this movie" é transformada em "I/PRP love/VBR this/PP movie/NN" onde PRP = pronome, VBR = verbo, PP = preposição e NN = substantivo. Os classificadores simplesmente passam a tratar o par (palavra/*pos*) como uma palavra. Utilizamos o *Stanford Tagger* (23) para fazer a classificação morfossintática do *corpus*. Este classificador morfossintático implementa um algoritmo de máxima entropia e reporta uma precisão de 97%.

3.4 Filtro de Subjetividade

Um dos problemas identificados por Pang et al. em (17) é o fato de que uma classe de frases (aquelas que não emitem opiniões) geram um ruído indesejável para o classificador. Obviamente não faz sentido falar em classificação de sentimento sem sabermos *onde* as opiniões são emitidas. Portanto, um primeiro passo da tarefa de *Sentiment Classification* é fazer um *Subjectivity Mining*.

Para estimar o *sentiment* relacionado a uma empresa é necessário antes identificar quais textos emitem opinião e quais simplesmente apresentam dados factuais. Assim, *Subjectivity Mining* é considerado um problema complementar à *Sentiment Analysis*. Alguns dos principais trabalhos neste tema são (21, 29).

Em geral, a classificação de subjetividade trabalha numa granularidade de frases ou parágrafos, isto é, a tarefa consiste em classificar se um parágrafo

ou frase é subjetiva ou objetiva. É obvio que esta é uma premissa simplificadora pois uma frase pode conter tanto dados factuais como opiniões. Até mesmo a existência desta fronteira entre subjetivo e objetivo é questionada por lingüistas, porém esta é uma questão filosófica e fora do escopo deste trabalho.

Um dos problemas descritos em (17) é a presença de frases que "contam a estória" do filme nas avaliações (e.g. "The protagonist tries to protect her good name"). Tais frases não são informativas quanto ao sentimento do avaliador e geram um ruído indesejado para o classificador impactando negativamente no seu desempenho. Fizemos um pré-processamento no texto com o objetivo de filtrar estas frases que não expressam opiniões. Uma descrição de como aplicamos o filtro de subjetividade está no Capítulo 6.

3.5 Seleção de Features

O objetivo em Seleção de *Features* é dado um conjunto de *features* encontrar o subconjunto mais útil para a tarefa de classificação.

Podemos estimar a distribuição sobre as ocorrências das *features* razoavelmente bem quando a quantidade de instâncias é consideravelmente maior do que o número de *features*. Infelizmente, isto raramente ocorre em problemas de categorização de texto. Considere o modelo de documento binário onde o número de ocorrências de um termo no documento é ignorado. Dado um léxico W, existem 2^W possíveis representações de documentos. Considerando o nosso *corpus*, este número seria $2^{30000} \approx 10^{10000}$, enquanto que nossa amostra é composta por 2000 documentos. Portanto, qualquer estimativa da distribuição conjunta sobre todos os termos será sempre uma aproximação grosseira e crua.

Os classificadores textuais ao invés de tentarem estimar a distribuição conjunta de todos os termos, se restringem a estimar a distribuição *marginal* de cada termo em cada classe. Os classificadores baseiam-se na premissa de que se um termo t é visto mais vezes nos documentos da classe c_1 do que nos da classe c_2, um documento contendo um termo t tem mais chances de pertencer a classe c_1. Porém, inferir a partir de um conjunto de treinamento limitado a classe em que um termo ocorre com maior freqüência não é trivial.

Ilustramos esta questão considerando o seguinte exemplo: Considere D documentos amostrais de cada classe e um termo t. Abrir um documento e verificar se este contém t é como lançar uma moeda. Para as 2 classes, imaginemos 2 moedas, com $\Pr(\text{"}cara\text{"}) = \phi_1$ e ϕ_2, cada uma das quais foi lançada N vezes e produziu-se k_1 e k_2 "caras" respectivamente. Note que é

possível que $\phi_1 < \phi_2$ apesar de $k_1 > k_2$, principalmente se N é pequeno. Se N é pequeno demais para crermos que $\phi_1 < \phi_2$ ou $\phi_1 > \phi_2$ com confiança suficiente, é melhor não considerarmos t na classificação ao invés de criar um modelo não-confiável que leva a erros de classificação. A construção de um modelo não confiável que se adéqua bem ao conjunto de treinamento porém falha em generalizar para dados de teste não vistos é chamado de *overfitting*.

Seleção de *features* pode ser heurística, guiada por conhecimento lingüístico e de domínio ou estatístico. Alguns classificadores eliminam as chamadas "stopwords" (e.g. *a*, *an*, *the*). Testamos este método usando alguns dicionários de *stopwords* e verificamos que para o nosso problema o impacto mostrou-se desprezível (mínima melhoria quando usados em alguns modelos e mínima piora quando usado em outros). Alguns classificadores também fazem aproximações de seleção de *features* ao ignorar termos que são muito freqüentes ou muito raros de acordo com limiares empiricamente selecionados (04).

A medida que conjuntos de dados tornam-se maiores e mais complexos, estas heurísticas simples passam a não ser suficientes. Seleção de *features* é desejável não somente para evitar *overfitting* e melhorar a precisão mas também visando manter a precisão enquanto descartamos o maior número de *features* possível, diminuindo assim os tempos de treinamento e classificação além de ficarmos com um modelo mais "enxuto" tornando a interpretação dos resultados por especialistas mais gerenciável.

Um algoritmo ótimo para seleção de *features* seria o seguinte: seleciona todos os possíveis subconjuntos de *features* e para cada subconjunto treina e testa o classificador. O subconjunto com mais alta precisão passaria a ser o conjunto de *features*. Este algoritmo é obviamente computacionalmente inviável para problemas de categorização de texto. Portanto, a busca por este subconjunto precisa ser limitada.

Métodos estatísticos de seleção de *features* em geral analisam a correlações entre as *features* e as classes. Alguns métodos de seleção de *features* são Chi-Quadrado de Pearson (também conhecido como *teste de* X_2), Índice de Discriminação de Fischer e Informação Mútua Média) (04). Uma outra técnica simples e bastante usada consiste em selecionar as *features* de maior ocorrência absoluta.

Utilizamos em nossos experimentos o método que consiste em selecionar as *features* com valores IMM (Informação Mútua Média) mais altos.

3.5.1

Informação Mútua Média

Informação Mútua Média é calculada como sendo a diferença entre a entropia da variável de classe, $H(C)$, e a entropia da variável se classe condicionada sobre a presença ou ausência de uma *feature* de interesse, $H(C, F_i)$:

$$\begin{aligned} I(C; F_i) &= H(C) - H(C|F_i) \\ &= -\sum_{c \in C} \Pr(c) log(\Pr(c)) + \sum_{p_i \in \{0,1\}} \Pr(p_i) \sum_{c \in C} \Pr(c|p_i) \log(\Pr(c|p_i)) \\ &= -\sum_{c \in C} \sum_{p_i \in \{0,1\}} \Pr(c|p_i) \log \left(\frac{\Pr(c, p_i)}{\Pr(c) \Pr(p_i)} \right) \end{aligned} \quad (3\text{-}1)$$

$\Pr(c)$, $\Pr(p_i)$, e $\Pr(c, p_i)$ são calculados pela soma sobre todos os documentos, isto é, $\Pr(c)$ é o número de documentos da classe c dividido pelo número total de documentos, $\Pr(p_i)$ é o número de documentos que contém a *feature* f_i dividido pelo número total de documentos, e $P(c, p_i)$ é o número de documentos da classe c que contém a *feature* f_i dividido pelo número total de documentos. Se $\Pr(c, p_i) = 0$, substituímos este termo por $\epsilon = 0.000001$ evitando assim a computação do logaritmo de zero na Equação (3-1).

Uma vez computada a Informação Mútua Média para todas as *features* do *corpus* de treinamento selecionamos as M *features* de maior valor $I(C; F)$. Note que para selecionar estas M features não é necessário ordenar todo o conjunto.

Uma vez calculadas as IMM, executamos o algoritmo *K-maiores* para fazer esta seleção que tem complexidade linear em função do número de *features*.

A Tabela (3.5.1) mostra as 15 *features* de maior Informação Mútua Média extraídas de um dos experimentos. A tabela também mostra o número de documentos positivos e negativos em que cada *features* ocorre.

Feature	+	-	$I(C;F_i)$
bad	230	464	1.45871
worst	41	170	1.43918
stupid	34	138	1.42744
boring	42	151	1.4267
ridiculous	19	98	1.42079
waste	18	94	1.41959
wasted	16	88	1.41848
outstanding	61	5	1.41846
awful	19	94	1.41831
lame	14	78	1.41489
life	443	297	1.41361
wonderfully	63	9	1.41254
perfect	167	70	1.41236
supposed	72	167	1.41108
mess	29	98	1.40987

Tabela 3.1: 15 features de IMM mais alto