UMA ABORDAGEM BAYESIANA PARA A CLASSIFICAÇÃO DE TEXTOS

Cloves Adriano
Michael Sieben

Resumo

O objetivo do trabalho é apresentar um novo algoritmo de Processamento de Linguagem Natural e, além disso, dar uma amostra da capacidade do mesmo resolvendo um problema real. Treinaremos nosso modelo de modo que ele consiga distinguir comentários de eleitores e não eleitores do 52º Prefeito da cidade de São Paulo, João Doria. A metodologia usada consistiu em extrair comentários de leitores realizados em notícias de alguns portais na web, sobre o Pref. João Doria. Para isso, usou-se técnicas de raspagem de dados web tendo como ferramenta a linguagem de programação Python. Obtidos os comentários, forneceremos alguns exemplos de treinamento para que nosso modelo consiga aprender e assim conseguir classificar o restante dos comentários como sendo ou não de um eleitor de Doria. O algoritmo é inspirado em técnicas de Aprendizagem Estatística Supervisionada, foi desenvolvido em cima do Teorema de Bayes e consiste basicamente no uso de uma matriz de probabilidades que é gerada pelas palavras dos comentários que foram classificados manualmente. Para diminuir a dimensionalidade de tal matriz, fez-se um tratamento dos comentários removendo palavras que não acrescentam informações relevantes, como artigos e conjunções, e extraindo apenas os radicais. Encerra-se concluindo sobre o objetivo proposto e discutindo os pontos fortes e fracos do algoritmo usado.

Palavras chaves: ***Estatística, raspagem de dados, Naive Bayes***

1. Introdução

Com tantos dados sendo produzidos atualmente pelos usuários da internet, o grande desafio atual não é mais como armazená-los, mas sim como extrair informações que agreguem alguma forma de valor dessa massa desorganizada de dados [1]. Pelos avanços tecnológicos da atual era digital, aumentou-se imensamente a facilidade de se comunicar. As redes sociais e fóruns online representam uma grande fonte de dados em formato de imagens, textos, áudios e vídeos, que são produzidos pela população em geral e que expressam suas mais diversas opiniões e sentimentos. Tais dados representam um grande potencial para pesquisas nas mais diversas áreas como em política [2], saúde [3], marketing [4], segurança [5] e economia [6]. Particularmente os textos, que são um tipo comum de dados não estruturados [7], são extremamente informativos e a compreensão automática dos mesmos trazem aplicações muito úteis.

A área que se ocupa com a compreensão automática da linguagem, tanto falada quanto escrita, é um subcampo da Aprendizagem de Máquina chamado Processamento de Linguagem Natural (PLN) [8]. Atualmente, pesquisadores no campo de PLN já vêm produzindo resultados convenientes e estão desenvolvendo técnicas novas e relevantes como a Mineração de Textos [7], a Mineração de Sentimentos [9] e a Mineração de Opiniões [9].

No presente artigo, pretendemos apresentar um novo algoritmo de classificação, baseado no Teorema de Bayes, que se mostra muito eficaz na compreensão de textos. Além da apresentação teórica, faremos uma aplicação real: treinaremos nosso modelo de modo que ele consiga distinguir comentários de eleitores e não eleitores do 52º Prefeito da cidade de São Paulo, João Doria.

2. O Teorema de Bayes

O *Teorema de Bayes* descreve a probabilidade de um evento E_1, na base do conhecimento "a priori" de um outro evento E_2 que se pode relacionar ao primeiro. Essa probabilidade também é chamado "a posteriori". Em termos matemáticos isso é escrito assim:

Para dois eventos E_1 e E_2, que ocorrem com as probabilidades $\mathbb{P}(E_1)$ e $\mathbb{P}(E_2)$ com $\mathbb{P}(E_2) \neq 0$, a probabilidade de E_1 acontecer, dado que E_2 aconteceu, é dado por o seguinte formalismo:

$$\mathbb{P}(E_1|E_2) = \frac{\mathbb{P}(E_1 \cap E_2)}{\mathbb{P}(E_2)}, \tag{1}$$

com a probabilidade $\mathbb{P}(E_1 \cup E_2)$ de E_1 e E_2 acontecer ao mesmo tempo. O Teorema de Bayes fornece uma forma de calcular a probabilidade posterior $\mathbb{P}(E_1|E_2)$ a partir de $\mathbb{P}(E_1)$, $\mathbb{P}(E_2)$ e $\mathbb{P}(E_2|E_1)$. Veja a equação abaixo:

$$\mathbb{P}(E_1|E_2) = \frac{\mathbb{P}(E_2|E_1)\mathbb{P}(E_1)}{\mathbb{P}(E_2)}. \tag{2}$$

A prova do teorema se conclui do seguinte fato:

$$\mathbb{P}(E_1|E_2) = \frac{\mathbb{P}(E_1 \cap E_2)}{\mathbb{P}(E_2)} = \frac{\frac{\mathbb{P}(E_1 \cap E_2)}{\mathbb{P}(E_1)} \cdot \mathbb{P}(E_1)}{\mathbb{P}(E_2)} \tag{3}$$

$$= \frac{\mathbb{P}(E_2|E_1) \cdot \mathbb{P}(E_1)}{\mathbb{P}(E_2)}. \tag{4}$$

3. O Classificador Naive Bayes

Na teoria matemática de probabilidade existem funções ue atribuem probabilidades a certos eventos. Essas funções se chamam classificadores, porque elas "classificam" esses eventos. Apresentamos a seguir uma nova forma de classificação baseada no Teorema de Bayes que se mostrará muito útil na classificação de textos escritos.

Supondo que X é um conjunto de eventos x_i, $n \in \mathbb{N}$ e o vetor $x = (x_1, \ldots, x_n) \in X$ seja um evento que se compõe de n elementos independentes. Além disso a classe $C_k \in C$ representa um resultado que pode depender de x. A probabilidade de C_k acontecer se x ocorrer é dado pelo termo

$$\mathbb{P}(C_k|x_1, \ldots, x_n). \tag{5}$$

(5) pode ser escrita como:

$$\mathbb{P}(C_k|x) = \frac{\mathbb{P}(C_k, x_1, \ldots, x_n)}{\mathbb{P}(x_1, \ldots, x_n)}. \tag{6}$$

Agora caso (6) seja considerado:

$$\mathbb{P}(C_k, x_1, \ldots, x_n) \tag{7}$$

$$= \mathbb{P}(x_1, \ldots, x_n, C_k) \tag{8}$$

$$= \mathbb{P}(x_1|x_2, \ldots, x_n, C_k) \cdot \mathbb{P}(x_2, \ldots, x_n, C_k) \tag{9}$$

$$= \ldots \tag{10}$$

$$= \mathbb{P}(x_1|x_2, \ldots, x_n, C_k) \cdot \mathbb{P}(x_2|x_3, \ldots, x_n, C_k) \tag{11}$$

$$\cdot \ldots \cdot \mathbb{P}(x_{n-1}|x_n, C_k) \cdot \mathbb{P}(x_n|C_k) \cdot \mathbb{P}(C_k) \tag{12}$$

E porque x_i e x_j são independente para $i \neq j$, o último termo vira:

$$= \mathbb{P}(x_1|C_k) \cdot \mathbb{P}(x_2|C_k) \cdot \ldots \cdot \mathbb{P}(x_n|C_k) \cdot \mathbb{P}(C_k) \tag{13}$$

$$= \mathbb{P}(C_k) \cdot \prod_{i=1}^{n} \mathbb{P}(x_i|C_k). \tag{14}$$

Com isso o seguinte é verdade:

$$\mathbb{P}(C_k|x) = \frac{\mathbb{P}(C_k, x_1, \ldots, x_n)}{\mathbb{P}(x_1, \ldots, x_n)} \tag{15}$$

$$= \frac{\mathbb{P}(C_k) \cdot \prod_{i=1}^{n} \mathbb{P}(x_i|C_k)}{\mathbb{P}(x)} \tag{16}$$

$$= \frac{\mathbb{P}(C_k)}{P(x)} \cdot \prod_{i=1}^{n} \mathbb{P}(x_i|C_k). \tag{17}$$

Agora um classificador é uma função da seguinte aparência:

$$\hat{c} = \underset{C_k \in C}{\operatorname{argmax}}\, \mathbb{P}(C_k|x) \tag{18}$$

$$= \underset{C_k \in C}{\operatorname{argmax}}\, \frac{\mathbb{P}(C_k)}{P(x)} \cdot \prod_{i=1}^{n} \mathbb{P}(x_i|C_k). \tag{19}$$

3.1. Treinando o Classificador Bayes

Como podemos aprender as probabilidades $\mathbb{P}(C_k)$ e $\mathbb{P}(x_i|C_k)$? Vamos primeiramente considerar a maximoverosimilhança. Nós simplesmente usaremos as frequências dos eventos, em nosso caso cada palavra dos comentários dos leitores é um evento, nos dados. Perguntamos qual é a percentagem dos comentários que estão em nosso conjunto de treinamento [1] estão na classe C_k, e assim obtemos uma estimativa de C_k, e assim obtemos uma estimativa de $\mathbb{P}(C_k)$. Seja N_{C_k} o número de comentários em nossos dados de treinamento com classe C_k e N_coment seja o número total de comentários. Então nós obtemos o seguinte estimador:

$$\hat{\mathbb{P}}(C_k) = \frac{N_{C_k}}{N_{coment}}. \tag{20}$$

Para aprender a probabilidade $\mathbb{P}(x_i|C_k)$, assumimos que é a probabilidade de existência da palavra x_i nos comentários da classe C_k. Calculamos a probabilidade como a fração de vezes que a palavra x_i aparece entre todas as palavras em todos os comentários da classe C_k. Primeiro concatenamos todos os comentários com categoria C_k em uma grande categoria de "texto" do tipo C_k. Então usamos a frequência de x_i neste "texto" concatenado para dar uma estimativa de maximoverossimilhança da probabilidade:

$$\hat{\mathbb{P}}(x_i|C_k) = \frac{\text{count}(x_i, C_k)}{\sum_{x \in \overline{X}} \text{count}(x, C)}. \tag{21}$$

Aqui $\overline{X}$ representa a união de todos os elementos que existem em todos os $x \in X$. Em nosso estudo exemplo, o conjunto de todos os comentários feitos pelos leitores em notícias sobre o Prefeito Jão Dória é representado por X, as palavras x_i, quais nos supomos como independente uns dos outros. As classes C dos comentários são:

C_1 : **Não eleitor do Prefeito** - representado por 0,

C_2 : **Eleitor do Prefeito** - representado por 1.

Após processar todos os comentários do conjunto de treino, o algoritmo irá gerar uma matriz de probabilidades em que as linhas são as classes mencionadas anteriormente, portanto duas linhas, e as colunas serão todos os eventos, ou seja, todas as palavras de todos os comentários do conjunto de treino. Supondo que a cardinalidade da união de todas as palavras x_j de todos os comentários seja $|\hat{X}| = m$, teremos a seguinte representação esquemática de tal matriz de probabilidades:

	x_1	x_2	...	x_m	$\hat{\mathbb{P}}(C_k)$
C_1	$\hat{\mathbb{P}}(x_1\|C_1)$	$\hat{\mathbb{P}}(x_2\|C_1)$	...	$\hat{\mathbb{P}}(x_m\|C_1)$	$\hat{\mathbb{P}}(C_1)$
C_2	$\hat{\mathbb{P}}(x_1\|C_2)$	$\hat{\mathbb{P}}(x_2\|C_2)$	...	$\hat{\mathbb{P}}(x_m\|C_2)$	$\hat{\mathbb{P}}(C_2)$
$\hat{\mathbb{P}}(x_j)$	$\hat{\mathbb{P}}(x_1)$	$\hat{\mathbb{P}}(x_j)$	...	$\hat{\mathbb{P}}(x_n)$	1

[1] O conjunto de treinamento todos os comentários são classificados manualmente, ou seja, cada comentário deste conjunto contém a informação de qual classe(eleitor ou não do Pref. João Dória) ele pertence.

Para obter a classificação de um comentário x, é preciso calcular as as probabilidades $\mathbb{P}(C_1|x)$ e $\mathbb{P}(C_2|x)$. A $\max(\mathbb{P}(C_1|x), \mathbb{P}(C_2|x))$ vai determinar a classificação de um comentário. Em pseudocódigo, o algoritmo procede da seguinte maneira:

1. Pegue um comentário no conjunto de comentários,
2. Para cada palavra do comentário, use a matriz de probabilidades para obter a probabilidade dessa palavra pertencer a classe 0,
3. Multiplicar as probabilidades obtidas para cada palavra do comentário entre si e também multiplicar pela probabilidade da classe 0 acontecer. Guardar esse valor,
4. Repita os passos 2 e 3 tendo como referência a classe 1,
5. Classificar o comentário como pertencente à classe que teve o maior valor da multiplicação das probabilidades $(\max(\mathbb{P}(C_1|x), \mathbb{P}(C_2|x)))$,
6. Repita os passos anteriores para os comentários restantes.

4. Análise dos dados

4.1. Obtendo os comentários

Para obter os comentários, foi usado a técnica de *Raspagem de Dados*. Usou-se a linguagem de programação Python e as suas respectivas bibliotecas Request, para se conectar a páginas da web, e Beautifulsoup para extrair o conteúdo desejado no HTML das páginas. Em algumas páginas web, os comentários dos leitores estavam sendo renderizados por JavaScript, de modo que também foi necessário o uso da biblioteca Selenium para conseguir capturá-los.

Os Dados obtidos foram: data de publicação da matéria, título da matéria, URL da metéria, comentários dos leitores feitos na matéria. Estes dados foram armazenados numa planilha excel conforme figura 1:

	A	B	C	D
1	data	titulo	url	comentario
2	23/08/20	João Doria tem título de cidadão honorário negado na Assembleia...	http://g1.	população são fáceis de serem manipuladas, tá arriscado mesmo esse politiquinho de pulôver ser eleito presidente.
3	23/08/20	João Doria tem título de cidadão honorário negado na Assembleia...	http://g1.	voto dos desavisados e manter os corrompidos no poder!
4	23/08/20	João Doria tem título de cidadão honorário negado na Assembleia...	http://g1.	Eu não entro nessa arapuca não! Sou BOLSONARO2018!
5	23/08/20	João Doria tem título de cidadão honorário negado na Assembleia...	http://g1.	sou mais dar esse premio para o newton cardoso!!
6	23/08/20	João Doria tem título de cidadão honorário negado na Assembleia...	http://g1.	que fazer, Zé Mané.....
7	23/08/20	João Doria tem título de cidadão honorário negado na Assembleia...	http://g1.	PELO BRASIL ATE AGORA..PRINCIPALMENTE POR SAO PAULO...QUE JOGADO AS MINGUAS..
				políticos. Os honrados são os mais esquecidos.

Figura 1: Exemplo da planilha excel.

O código usado para a captura destes dados pode ser visto e baixado no GitHub pelo seguinte link:

https://github.com/SCloves/popularidade_do_prefeito_doria

Ao todo, foram obtidos 10557 comentários que foram realizados entre 03/10/2016 e 31/08/2017 em matérias sobre o Pref. João Dória no G1.

4.2. Pré-processamento dos comentários

Como exemplo, iremos fazer o pré-processamento dos comentários das figuras 2 e 4

Figura 2: Comentário de um leitor pertencente a classe 0.

Figura 3: Comentário de um leitor pertencente a classe 1.

Estes comentários receberam as classificações manualmente pois farão parte do conjunto de comentários que servirão para treinar o algoritmo. São removidas

todas as palavras pouco relevantes como conjunções, artigos definidos e indefinidos etc. Tecnicamente essas palavras pouco informativas são chamadas de *StopWords*. Após a remoção das StopWords, cada comentário do conjunto de treino assumirá o formato `([vetor de palavras relevantes do comentário],classe do comentário)`. A seguir, temos o exemplo de como os comentários anteriores ficaram:

1. O vetor do comentário da figura 2:
   `(['FORA', 'DORIA', 'SAO', 'PAULO', 'NAO', 'PRECISA', 'VOCE', 'VIU', 'MALDADE', 'JOVENS'], 0)`
2. O vetor do comentário da figura 4:
   `(['DORIA', 'PRESIDENTE', '2018', 'BRASIL', 'MUDAR', 'REALMENTE'], 1)`

Além disso, para fins de comparação, todas as palavras devem ficar em letras minúsculas, caso contrário palavras como "Presidente" e "presidente" seriam entendidas como sendo diferentes pelo algoritmo. Buscou-se também extrair apenas os radicais das palavras com a intenção de diminuir ainda mais a dimensão da matriz probabilística. Os comentários dos exemplos anteriores ficaram no seguinte formato:

1. O vetor alterado do comentário em figura 2:
   `(['for', 'doria', 'sao', 'pau', 'nao', 'precis', 'voc', 'viu', 'maldad', 'jov'], 0)`
2. O vetor alterado do comentário em figura 4:
   `(['doria', 'presid', '2018', 'brasil', 'mud', 'real'], 1)`

5. Resultados

O modelo foi treinado com 587 comentários classificados a mão. Com esses comentários, nosso modelo aprendeu algumas características relevantes na hora de se classificar um comentário. Na Figura 4, podemos ver quais radicais que o modelo acha significativos:

```
classificador.show_most_informative_features(5)
Most Informative Features
                parabém = True              1 : 0      =     24.1 : 1.0
                  2018. = True              1 : 0      =      9.1 : 1.0
                  parab = True              1 : 0      =      7.8 : 1.0
                 melhor = True              1 : 0      =      7.3 : 1.0
                 negóci = True              0 : 1      =      6.0 : 1.0
```

Figura 4: Características mais informativas encontradas pelo modelo.

Portanto, um comentário com o radical "parab" tem 7.8 vezes mais chance de pertencer a classe 1 (eleitor do Doria) do que a classe 0 (não eleitor do Doria). A seguir, nas figuras 5 e 6, podemos ver algumas das palavras mais usadas pelos leitores pertencentes às duas classes.

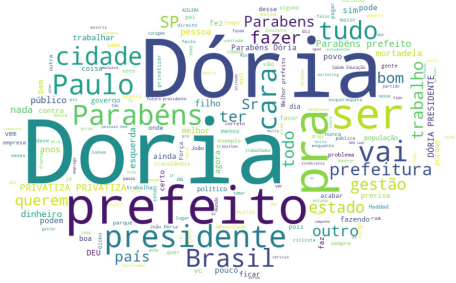

Figura 5: Nuvem de palavras mais usadas pelos eleitores de Doria.

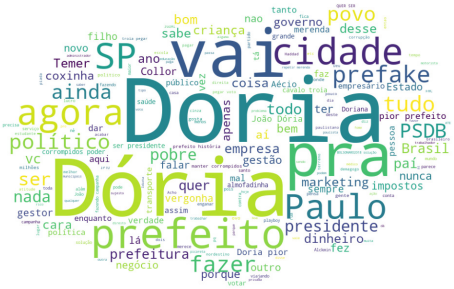

Figura 6: Nuvem de palavras mais usadas pelos não eleitores de Doria.

No grupo dos eleitores de Doria vemos que as palavras "Doria", "Prefeito", "Parabéns", "Presidente" e "Brasil" estão em destaque. Já no grupo dos não eleitores de Doria, além das palavras "Doria" e "Prefeito", podemos notar uma presença significativa do trocadilho "prefake". Cada uma dessas palavras darão seus respectivos pesos na hora de nosso modelo classificar um comentário.

Na Figura 7 e Figura 8 temos alguns exemplos de comentários que foram classificados corretamente pelo nosso modelo.

```
In [89]: comentario_teste4 = u'Imaginar que esse almofadinha é um potencial presidente do país mostra bem que futuro nos espera'

In [90]: preditor(comentario_teste4, classificador)
         O comentário não é de um eleitor do Dória

         0: 0.984830
         1: 0.015170

In [91]: comentario_teste5 = u'Globo já fazendo campanha antecipada pra esse novo Collor se lançar a presidência.'

In [92]: preditor(comentario_teste5, classificador)
         O comentário não é de um eleitor do Dória

         0: 0.971604
         1: 0.028396
```

Figura 7: Comentários reais que foram classificados pelo modelo como pertencentes a não eleitores de Doria.

```
In [83]: comentario_teste2 = 'Acelera Sampa!!!! Doria o terror dos mortadelas'

In [84]: preditor(comentario_teste2, classificador)
         O comentário é de um eleitor do Dória

         0: 0.307913
         1: 0.692087
```

Figura 8: Comentários reais que foram classificados pelo modelo como pertencentes a eleitores de Doria.

A Figura 9 apresenta um exemplo de comentário que foi classificado de maneira incorreta pelo nosso modelo.

```
In [174]: c41=u'Dória, em 3 meses de mandato já fez muita coisa por S.Paulo, \
              quer na campanha Cidade Limpa, quanto atendimento médico, \
              quanto a creches, os marginalizados, etc. TUDO ISSO EM 3 MESES. \
              OS MORTADELAS QUEREM QUE RESOLVAM O QUE NÃO FIZERAM NESTE TEMPO? \
              Aí ficam tentando denegrir a imagem de quem faz. Típico de difamador.'

In [175]: preditor(c41, classificador)
          O comentário não é de um eleitor do Dória

          0: 0.995238
          1: 0.004762
```

Figura 9: Comentário classificado de maneira incorreta pelo modelo.

Como teste, foram usados um total de 246 comentários, destes, 182 foram classificados corretamente pelo nosso modelo, ou seja, 74% dos comentários. Conforme a Matriz de Confusão na Figura 10, o modelo classificou 80 comentários de não eleitores do Doria de forma correta, 102 comentários de eleitores do Doria de maneira correta, 21 comentários de não eleitores do Doria de maneira incorreta e 43 comentários de eleitores do Doria de maneira incorreta.

```
  |   0   1 |
--+---------+
0 | <80> 43 |
1 |  21<102>|
--+---------+
(row = reference; col = test)
```

Figura 10: Matriz de Confusão do modelo.

6. Conclusão

Com os resultados obtidos, pode-se afirmar que o algoritmo já mostra uma acurácia significativa, embora tenha sido treinado com um conjunto relativamente pequeno de exemplos. Para uma pesquisa, seria recomendável considerar um conjunto maior de treinamento que também deveria ser mais diversificado. No presente trabalho, por exemplo, foram considerados comentários de apenas um portal de notícias na web, que foram escolhidas de uma maneira aleatória. Para garantir um viés mais baixo, os dados deveriam ser escolhidos de maneira uniforme levando em conta vários fatores como, variedade no tempo (antes, durante, depois da eleição), diferentes classes sociais, níveis de ensino e páginas de

notícias com diferentes posicionamentos políticos (esquerda, centro e direita).

No conjunto de treinamento, havia mais comentários desfavoráveis ao Prefeito João Doria, isso faz com que a probabilidade a priori tenha um peso maior para a classe 0, ou seja, na dúvida o nosso modelo classificará o comentário como sendo de um não eleitor do Doria. Isso faz com que comentários como o da Figura-9 por exemplo, que contém a palavra "mortadelas" usada quase exclusivamente pela classe 1, seja classificado como pertencente à classe 0, pois o mesmo contém muitas outras palavras que estão presentes em ambos os grupos, levando nosso modelo a ceder ao peso da probabilidade a priori da classe 0.

Além disso, a abordagem no algoritmo Naive Bayes leva em conta a suposição de independência de eventos, ou seja, cada palavra surge de forma independente nos comentários. Mas sabemos que isso não é de todo verdadeiro, já que a presença do trocadilho "prefake" diminui a chance da palavra "mortadela" aparecer no comentário, ou seja, há uma dependência negativa entre algumas palavras, se uma aparece a outra não aparece, e também dependências positivas, se uma aparece é bem provável que outra também irá aparecer, como no caso entre as palavras "Presidente" e "2018". Isso pode influenciar de modo negativo a aprendizagem do modelo, fazendo com que ele atribua probabilidade incorretas a algumas palavras. Apesar dessa suposição não se mostrar verdadeira para os comentários dos leitores, nosso modelo se mostrou razoavelmente acurado mesmo com uma quantidade pequena de exemplos.

Após todas essas observações, podemos concluir que nosso classificador possui as seguintes vantagens:

- Simplicidade: fácil de entender e implementar,
- Leve para treinar: não é necessária uma otimização complicada,
- Facilmente atualizável se novos dados de treinamento forem recebidos,
- Pequeno uso de memória,
- Embora a suposição de independência possa parecer às vezes irracional, seu desempenho geralmente é bom, mesmo para esses casos.

Referências

[1] http://www.b-eye-network.com/view/6311, 12/11/2017

[2] A. Ceron, L. Curini, S. M. Iacus, G. Porro: *Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens political preferences with an application to Italy and France*. new media & society, 2014, Vol. 16(2) p. 340–358

[3] C. Meaneya, R. Moineddina, T. Vorugantib, M. A. O'Briena, P. Kruegera, F. Sullivana,c: *Text mining describes the use of statistical and epidemiological methods in published medical research.* Journal of Clinical Epidemiology 74 (2016) p. 124 e 132

[4] Jacopo Soriano, Timothy Au and David Banks: *Text Mining in Computational Advertising.* Wiley Online Library 2013, Bd.:6 iss:4 p. 273 - 285

[5] P. Thompson: *Text Mining, Names and Security.* Idea Group Publishing 2005

[6] O. Netzer, R. Feldman, J. Goldenberg, M. Fresko: *Mine Your Own Business: Market-Structure Surveillance Through Text Mining.* Marketing Science, Vol. 31, No. 3, May–June 2012, p. 521–543

[7] S. M. Weiss, N. Indurkhya, T. Zhang: *Fundamentals of Predictive Text Mining.* Springer 2nd edition, London 2015, p. 11

[8] Angel R. Martinez: *Natural language processing.* WIREs Computational Statistics 2010, p. 352-357.

[9] Bing Liu: Sentiment Analysis. Cambridge University Press, Cambridge 2015