

# Análise de Sentimentos Utilizando Diferentes Técnicas de Aprendizagem de Máquina com Processamento Paralelo

Guilherme Alberto Sousa Ribeiro

Instituto de Informática - INF - UFG

25 de Junho de 2019

# Roteiro

- 1 Introdução
  - Introdução
- 2 Aprendizagem de Máquina
  - SVM
  - Naive Bayes
  - Redes Neurais
  - Florestas Randômicas
- 3 Transformações
  - Pré-processamento
  - Vetorização
- 4 Organização e Resultados do Trabalho
  - Metodologia do Trabalho
  - Corpus
  - Plano de Desenvolvimento e Testes
  - Resultados Esperados e Obtidos

# Introdução

- Popularização das Redes Sociais e aumento do volume de dados;
- Desenvolvimento de técnicas para extração de conhecimento dessas bases;
- NLP tenta interpretar a linguagem utilizada por seres humanos, denominada linguagem natural;
- Os computadores não são capazes de compreender as palavras diretamente, para isso, são necessárias transformações.

# Introdução

- Um dos problemas estudados dentro do NLP é a análise de sentimentos;
- Tenta detectar o que uma pessoa quis dizer ou qual sentimento quis emitir com determinada opinião textual;
- Sentimento possui polaridade: negativo, positivo ou neutro.

# SVM

- Divide conjunto de dados em hiperplanos;
- As classes que compõem o conjunto de dados devem ser linearmente separáveis;
- Para que as classes sejam definidas, o SVM trabalha com a minimização do erro.

# Máquina de Vetor de Suporte (SVM)

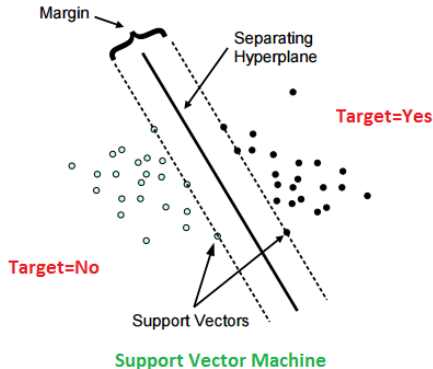


Figura 1: Demonstração de como funciona um algoritmo SVM de *kernel* linear.

# Naive Bayes

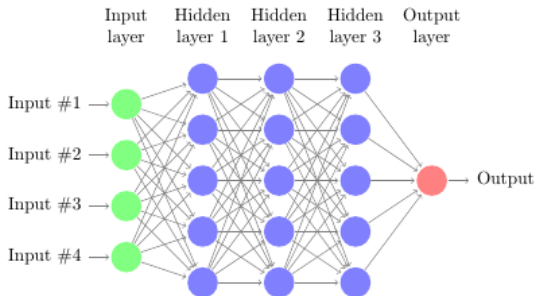
- Amplamente utilizado para classificação de textos;
- Trabalha com o cálculo de probabilidades posteriores;
- Calcula as probabilidades do texto pertencer a uma classe posterior e atribuí a ele a classe de maior probabilidade;
- Utilização do modelo multinomial.

# Redes Neurais

- Baseados no modelo biológico de um neurônio;
- Otimização de pesos através da multiplicação deles, pelo vetor de características;
- Modelo mais tradicional é a MLP (*Multi-layer Perceptron*);
- Rede Neural x Rede Neural Profunda (*Deep Learning*).



# Exemplo de uma Rede Neural Profunda



**Figura 2:** Modelo de uma rede neural profunda. Essas redes se caracterizam por ter vários *layers* na camada escondida e exatamente por isso recebem essa denominação.

# Florestas Randômicas

- Combinação de diversas árvores de decisão;
- Cada árvore é treinada individualmente utilizando todo o conjunto ou subconjuntos dos vetores de características;
- Após o treino de cada árvore, um outro algoritmo é utilizado para fazer a votação entre os resultados;
- O resultado da votação indica a classe predita.

## Exemplo de uma Floresta Randômica

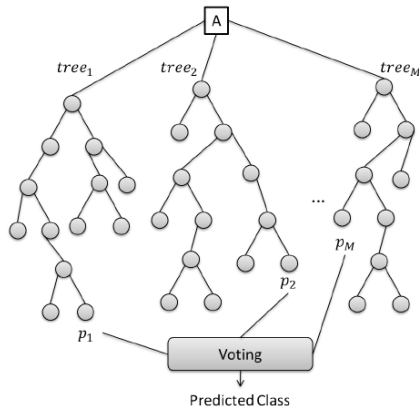


Figura 3: Exemplo de uma Floresta Randômica. São utilizadas diversas abordagens baseadas em árvore.

# Transformações dos Dados

- Métodos de aprendizagem de máquina possuem limitações para lidar com texto;
- Para tornar o processo de aprendizado menos custoso, os dados devem sofrer algumas transformações antes de serem imputados nos algoritmos;
- Ex.: transformar sentenças em um vetor de características ou em números que representem as palavras.

# Pré-processamento

- Tem como principal tarefa deixar o texto mais limpo, o que o torna mais simples de ser processado;
- Técnicas de Pré-processamento:
  - 1 *Lower Case*  
Ex.: FLY → fly;
  - 2 Remoção de Caracteres HTML  
Ex.: <p>Sentence</p> → Sentence;
  - 3 Remoção de Pontuação;  
Ex.: John, how are you? → John how are you;
  - 4 Remoção das Stop-Words  
Ex.: The big house → big house;

# Vetorização

- Considerada uma das etapas mais importantes do processo de análise de texto;
- Os textos são transformados em sequências de palavras;
- Sequências de palavras são transformadas em características numéricas;
- Após essa última transformação, os dados são submetidos as técnicas de aprendizagem de máquina.

# Vetorização - *Bag of Words*

- Forma mais simples de vetorização de uma sentença;
- Baseado na contagem de ocorrências de uma palavra dentro de um texto ou documento, gerando quantidades para cada palavra;
- Ex.: *Im eating*  
Vetor  $\rightarrow$  [*Im*: 1, *eating*: 1, *with*: 0, *my*: 0, *brother*: 0]  
Vetor  $\rightarrow$  [1, 1, 0, 0, 0]

# Vetorização - Frequência de Vezes e Inversa no Documento

- Também representa os textos através de vetores, porém, ao invés das quantidades, ele irá conter um *score* para cada uma das palavras:

$$TF = \frac{\text{Num de vezes que a palavra aparece}}{\text{Num de palavras no texto}} \quad (1)$$

$$IDF = \log_e \left( 1 + \frac{\text{Num de textos}}{\text{Num de palavras no texto}} \right) \quad (2)$$

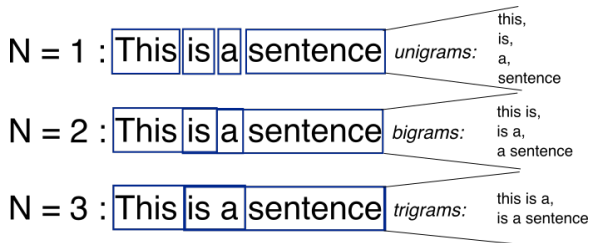
$$TF - IDF = TF * IDF \quad (3)$$



# Vetorização - N-gramas

- Coleções de palavras que podem ser agrupadas N por N;
- As palavras vão sendo agrupadas conforme o valor de N e de acordo com a sequência do texto;
- Bastante utilizada na classificação de texto devido, pelo fato de algumas palavras isoladas não possuírem um significado, ou sozinhas possuem um e agrupadas possuem outro.

## Vetorização - N-gramas



**Figura 4:** Demonstração de N-gramas. Na imagem, temos um exemplo simples de um unigrama, um bigrama e um trigrama.

# Metodologia

- Organizar os testes de forma a evitar o *overfitting*;
- Utilização dos dados de treino e teste separadamente conforme fornecido pelas instituições que disponibilizaram os dados;
- Uso da técnica de validação cruzada em 10 pedaços (*10-fold-cross-validation*).

# Corpus

- IMDB:
  - Revisões de filmes: positivas e negativas;
  - Base de treino e teste com 25.000 registros;
  - Base de dados balanceada;

# Plano de Desenvolvimento e Avaliação

- Algoritmos implementados em Python, na versão 3.6;
- O conjunto de dados foi treinado com uma base e testado com outra, conforme separação das entidades que geraram os dados;
- Os dados também foram submetidos a testes no formato *10-folds-cross-validation*;
- Os algoritmos serão avaliados por quatro métricas: *Accuracy*, *Precision*, *F1-Score* e *Recall*.
- A base de dados será submetida aos três algoritmos tanto no modelo de programação serial como paralelo, de forma que esses resultados possam ser comparados.

# Arquitetura Serial

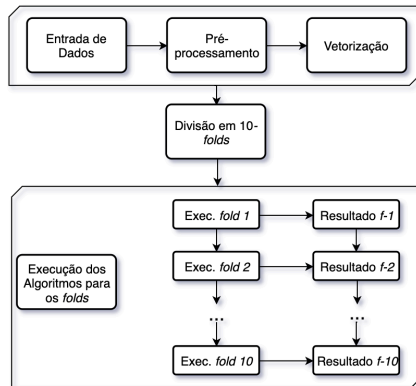
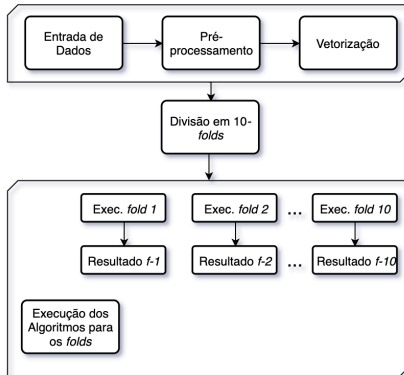


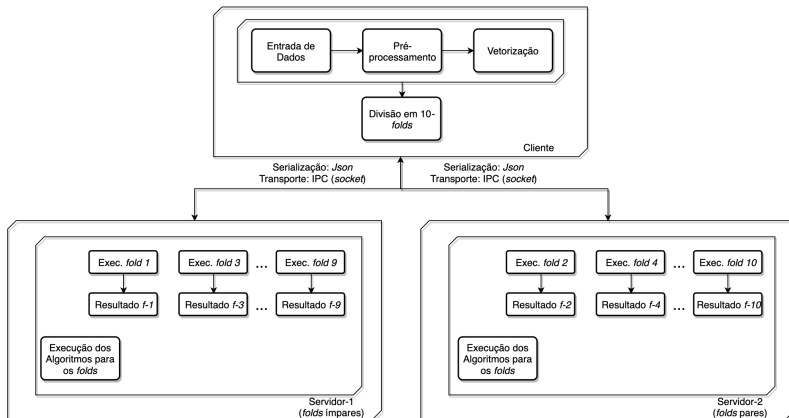
Figura 5: Desenho da arquitetura para o modelo de programação serial. Cada *fold* será executado sequencialmente.

# Arquitetura Paralela



**Figura 6:** Desenho de arquitetura para o modelo programação paralela. Cada um dos *folds* será executado executado simultaneamente por *threads* diferentes.

# Arquitetura Paralela Distribuída



**Figura 7:** Desenho de arquitetura para o modelo programação paralela. Cada um dos *folds* será executado executado simultaneamente por *threads* diferentes.



## Resultados Esperados

- Realizar a implementação das transformações e dos algoritmos descritos e aplicá-los a base de dados IMDB;
- Realizar uma comparação entre os algoritmos dentro da metodologia de implementação serial e paralela, de forma a mostrar que o paralelismo pode trazer ganhos em relação ao tempo de processamento sem que haja perda nos resultados das métricas avaliativas;
- Apresentar os resultados obtidos;
- Elaborar um ponto de partida para uma possível linha de pesquisa do doutorado dentro da disciplina.

# Detalhamento das métricas

- Sobre as métricas:
  - ① **Acurácia:** Verdadeiros positivos e negativos no geral, ou seja, qual a frequência geral de acertos do algoritmo;
  - ② **Precisão:** dentre todas as classificações de classe Positivo que o modelo fez, quantas estão corretas;
  - ③ **Recall:** Frequência com que o classificador encontra exemplos de uma classe;
  - ④ **F1-Score:** Combinação de precisão e *recall* (média harmônica) de forma a trazer um valor único que indique a qualidade geral do modelo.

## Resultados Obtidos

acImDB - Acurácia			
Algoritmos	BoW	TF-IDF	TF-IDF - Bigrams
SVM	83,53%	86,74%	<b>88,32%</b>
Forest	77,02%	77,50%	77,97%
Naive Bayes	83,39%	83,01%	85,15%
Rede Neural	<b>87,16%</b>	<b>87,49%</b>	<b>88,32%</b>

Tabela 1: Resultados parciais em termos de acurácia.

acImDB - Precisão			
Algoritmos	BoW	TF-IDF	TF-IDF - Bigrams
SVM	83,54%	86,76%	<b>88,32%</b>
Forest	77,61%	78,14%	78,58%
Naive Bayes	83,58%	83,34%	85,15%
Rede Neural	<b>87,29%</b>	<b>87,51%</b>	88,31%

Tabela 2: Resultados em termos de precisão.

## Resultados Obtidos

<i>acImDB - Recall</i>			
Algoritmos	BoW	TF-IDF	TF-IDF - Bigrams
SVM	83,53%	86,73%	<b>88,32%</b>
Forest	77,02%	77,50%	77,97%
Naive Bayes	83,49%	83,01%	85,15%
Rede Neural	<b>87,17%</b>	<b>87,49%</b>	88,31%

Tabela 3: Resultados parciais em termos de *recall*.

<i>acImDB - F1-Score</i>			
Algoritmos	BoW	TF-IDF	TF-IDF - Bigrams
SVM	83,52%	86,73%	<b>88,31%</b>
Forest	76,89%	77,36%	77,85%
Naive Bayes	83,47%	82,97%	85,13%
Rede Neural	<b>87,15%</b>	<b>87,48%</b>	<b>88,31%</b>

Tabela 4: Resultados em termos de *F1-Score*.

## Resultados Obtidos


acImDB - Tempo de Execução			
Modelo de Programação	BoW	TF-IDF	TF-IDF - Bigrams
SVM - Serial	19,74	2,30	2,31
SVM - Paralelo	<b>5,88</b>	<b>0,78</b>	<b>0,81</b>
ANN - Serial	286,88	289,12	294,56
ANN - Paralelo	<b>28,39</b>	<b>28,77</b>	<b>32,74</b>
RF - Serial	42,22	39,52	40,14
RF - Paralelo	<b>9,32</b>	<b>8,87</b>	<b>8,97</b>
Bayes - Serial	0,28	0,26	0,26
Bayes - Paralelo	<b>0,16</b>	<b>0,12</b>	<b>0,12</b>


**Tabela 5:** Resultados para os modelos de programação serial e paralela de todos os algoritmos para as três técnicas de vetorização utilizadas no decorrer do trabalho.


## Discussões

- Para praticamente todos os resultados das quatro métricas, utilizando as três técnicas de vetorização, a Rede Neural obteve um melhor desempenho que os demais algoritmos;
- O algoritmo *Random Forest* foi o que obteve o pior desempenho;
- Em relação ao objetivo do trabalho, foi provado que é possível obter os mesmos resultados com um tempo de resposta até 90% mais rápido se utilizado o modelo de programação paralelo;
- Em trabalhos futuros a ideia é utilizar bases de dados cada vez maiores e aplicá-las ao modelo de programação paralela em GPU's.


## Referências


 Das, B. and Chakraborty, S. (2018). An improved text sentiment classification model using tf-idf and next word negation.


 Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. In Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

 George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. 2011. Distributed Systems: Concepts and Design (5th ed.). Addison-Wesley Publishing Company, USA.

## Referências


 Shi, Y., Tian, Y., Kou, G., Peng, Y., and Li, J. (2011). Optimization based data mining: Theory and applications. In Advanced Information and Knowledge Processing.

 Llombart, O. R. (2017). Using machine learning techniques for sentiment analysis. Technical report, Universitat Autònoma de Barcelona, UAB, ESP, Final Project.

 Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., and Zheng, X. (2015). Tensorflow : Large-scale machine learning on heterogeneous distributed systems



## Referências

 Pradhan, V. M., Vala, J., and Balani, P. (2016). A survey on sentiment analysis algorithms for opinion mining. International Journal of Computer Applications, 133(9):7–11. Published by Foundation of Computer Science (FCS), NY, USA.