



**Curso:** Engenharia Mecânica/Computação

**Disciplina:** Machine Learning

**Profº:** FÁBIO JOSÉ AYRES

## **RELATÓRIO**

Projeto Final – Análise de Tweets de empresas de notícias para previsão  
de mercado

Grupo:

Eduardo Magalhães – [eduardofm3@al.insper.edu.br](mailto:eduardofm3@al.insper.edu.br)

Bruno Bitelli

Hugo Mendez

## Sumário

1	Introdução .....	1
2	Revisão Bibliográfica.....	1
3	Metodologia.....	10
3.1	Coleta e filtragem dos Dados .....	11
3.2	Latent dirichlet allocation (LDA).....	12
3.2.1	Testes de Sanidade.....	12
3.2.2	Media de Tópicos .....	13
3.2.3	Media de Tópicos + Variação da Bolsa para escândalos ...	14
3.2.4	Tópicos com avaliação individual .....	15
3.2.5	LDA + Word2Vec.....	15
3.3	Redes Neurais.....	16
3.3.1	LSTM prediction.....	16
3.3.2	LSTM Autoencoder .....	18
4	Resultados .....	18
4.1	Resultados Filtragens.....	19
4.2	LDA .....	21
4.2.1	Teste de sanidade .....	22
4.2.2	Media dos tópicos.....	23
4.2.3	Media dos tópicos + Variação da bolsa para escândalos ...	24
4.2.4	Tópicos com avaliação individual (Comparação de fontes) 27	
4.2.5	Tópicos com avaliação individual .....	30
4.2.6	LDA2Vec.....	32
4.3	Redes Neurais.....	34
5	Conclusões .....	34
6	Referências Bibliográficas.....	36

## 1 Introdução

Entender o mercado financeiro e especialmente modelar seu comportamento é um desafio tão velho quanto o próprio mercado, os primeiros testes com modelos de finanças quantitativa começaram desde 1900 com a tese de doutorado de Louis Bachelier, *Theory of Speculation*, e apesar das incontáveis tentativas existentes, poucos métodos possuem uma alta taxa de retorno com um baixo risco. Dessa forma, a motivação para encontrar novos métodos de modelagem de mercado é muito forte, especialmente pela promessa de lucro e de reconhecimento.

Recentemente, com a divulgação quase instantânea de notícias em mídias sociais, um novo modo de modelagem começou a ser testado por grandes empresas, utilizar técnicas de reconhecimento de linguagem natural para ler notícias, ou comentários de pessoas importantes e estimar a possível consequência dessas informações nos próximos dias do mercado.

Nesse sentido, o objetivo desse projeto é tentar utilizar as técnicas de interpretação de linguagem natural, junto a modelos clássicos *de Machine Learning* e modelos de *Deep Learning*, para tentar criar um modelo de previsão de mercado eficiente.

## 2 Revisão Bibliográfica

Antes de discutir os métodos e resultados apresentados nesse relatório é necessário entender um pouco sobre os métodos que serão adotados nas análises e pesquisas que já foram realizadas nesse sentido, dessa forma, essa seção trata de um pouco da história da interpretação de linguagem natural, modelos de análises de sentenças e referências bibliográficas relevantes.

O processo de análise de linguagem natural não é um desafio novo, desde a invenção dos primeiros computadores a capacidade de ler frases era um desafio, especialmente para tradução entre línguas. A IBM no Georgetown experimente em 1954 já realizava testes nesse sentido, tendo conseguido traduzir cerca de 60 frases de russo para inglês com

algoritmos complexos baseados em regras pré-determinadas de interpretação de frases.

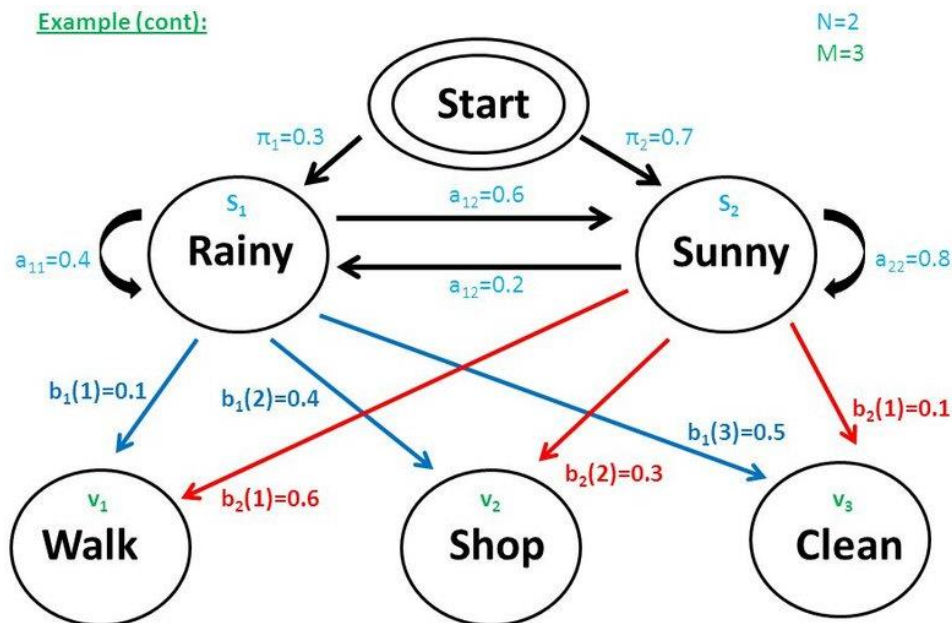
Apesar desse sucesso inicial, traduções e interpretações de frases tiveram um progresso muito lento, apesar das expectativas iniciais dos pesquisadores da IBM, devido à complexidade que sentenças não literais podem ter. Em 1966 no ALPAC-report (*Automatic Language Processing Advisory Committee*) foi relatado o progresso decepcionante dos algoritmos detalhados por regras de tradução. Com essa redução de expectativa, muitos pesquisadores passaram a tentar traduzir as palavras para dados que o computador pudesse interpretar ao invés de focar em um conjunto de regras tão detalhado dando mais flexibilidade a interpretação da máquina sobre o texto. Foi nesse período ocorreu o surgimento dos primeiros “chatbots” (Programas de computador que imitam conversações humanas) sendo “*PARRY*”, “*Racter*”, e “*Jabberwacky*” alguns exemplos deles.

Porém só em 1980 com os primeiros algoritmos de *Machine Learning* que progressos na interpretação de sentenças começaram a ser notados. Isso foi possível devido ao avanço constante do poder computacional disponível (lei de Moore) junto à redução da crença na teoria de linguagem de Chomskyan que pressupunha um conhecimento inato para domínio da linguagem <sup>[1]</sup>.

As primeiras versões de algoritmos foram baseadas em *Decision Trees* que permitiam criar um set de regras de interpretação e tradução muito mais completo e robusto que os criados manualmente por pesquisadores. Porém da mesma forma que as regras escritas a mão, esses modelos possuíam algumas falhas, e foi só com a introdução de modelos estatísticos associados ao uso de “*Hidden Markov Chains*” que grandes progressos voltaram a ocorrer. Essa combinação de modelos permitia criar uma análise de probabilidade que poderiam ser escolhidas após um estado atual e permitiam que pesos fossem associadas a cada probabilidade permitindo variar como o processo de decisão do computador fosse modulado (figura 1). Com essa combinação de algoritmos os primeiros modelos de análise de linguagem natural foram

criados, permitindo que máquinas escrevessem e traduzissem sentenças com um sucesso considerável, mas ainda não permitiam interpretar sentenças e associá-las a outros dados como assunto tratado ou sentimento presente por exemplo.

Figura 1: Exemplo de "Hidden Markov Model"



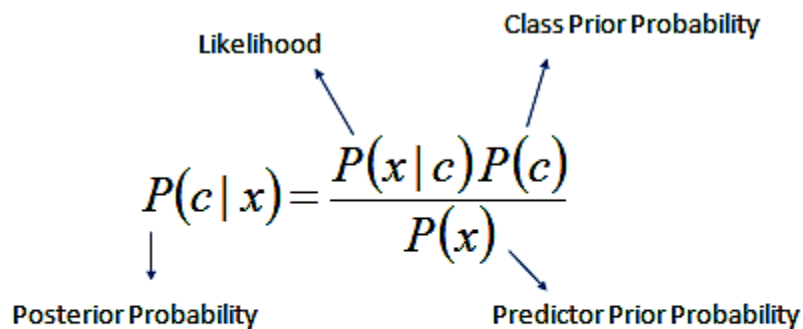
Fonte: Martin R. Beel J. Tkaczyk D. - "Citation Data-set for Machine Learning Citation Styles and Entity Extraction from Citation Strings" - 2018

Com a capacidade de formar frases e traduzir sentenças muitos pesquisadores optaram por tentar entender de fato o que está sendo passado por cada sentença, e nesse ponto algoritmos de categorização de palavras e de sentenças começaram a ser desenvolvidos com técnicas de *Machine Learning*.

Uma das técnicas mais antigas de classificação eram modelos "Naive Bayes" que começaram a ser estudados em paralelo as primeiras técnicas de tradução com a intenção de classificar textos e encontrar seu contexto ou sentimento. Esse modelo se baseia na distribuição probabilística das palavras em cada classe definida (Figura 2) o que faz com que seus resultados dependam fortemente de como essas probabilidades são construídas. Por isso, apesar de ter o início de seu desenvolvimento ainda na década de 60, esse modelo dependeu de avanços significativos na capacidade computacional dos computadores e de algoritmos de

treinamento e teste semelhantes a algoritmos de “*Machine Learning*” da década de 80 [2].

Figura 2: Equação de probabilidade Naive Bayes

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$


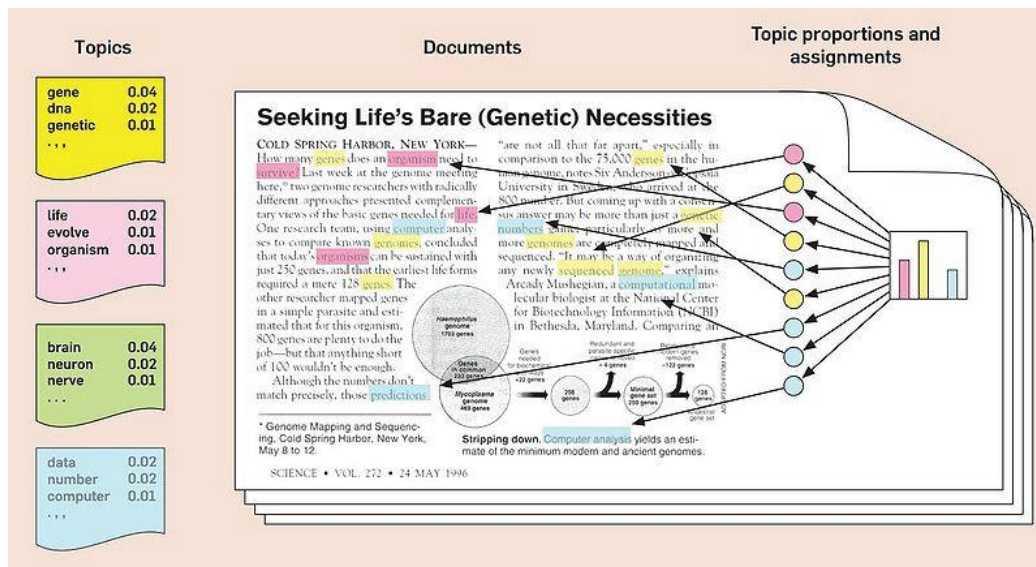
$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Fonte: [http://uc-r.github.io/naive\\_bayes](http://uc-r.github.io/naive_bayes)

Outro algoritmo interessante para classificação de texto é o LDA (*Latent Dirichlet allocation*), sendo consideravelmente mais novo que outros modelos com seu desenvolvimento proposto nos anos 2000 por J. Pritchard, M. Stephens and P. Donnelly em “*Inference of population structure using multilocus genotype data*” com uma aplicação completamente diferente.

Esse método parte do princípio de desconstruir as amostras fornecidas assumindo que essas foram criadas por um modelo generativo que poderia ser revertido. Esse modelo generativo teria utilizado tópicos de textos com probabilidades diferentes para um dado vocabulário de palavras e teria criado textos com distribuições diferentes de cada tópico, dessa forma a LDA utiliza um método não supervisionado para tentar recriar os tópicos de palavras e as respectivas probabilidades de cada palavra permitindo classificar todas as amostras de texto em porcentagem de cada tópico encontrado (Figura 3).

Figura 3: Ilustração de modelo LDA



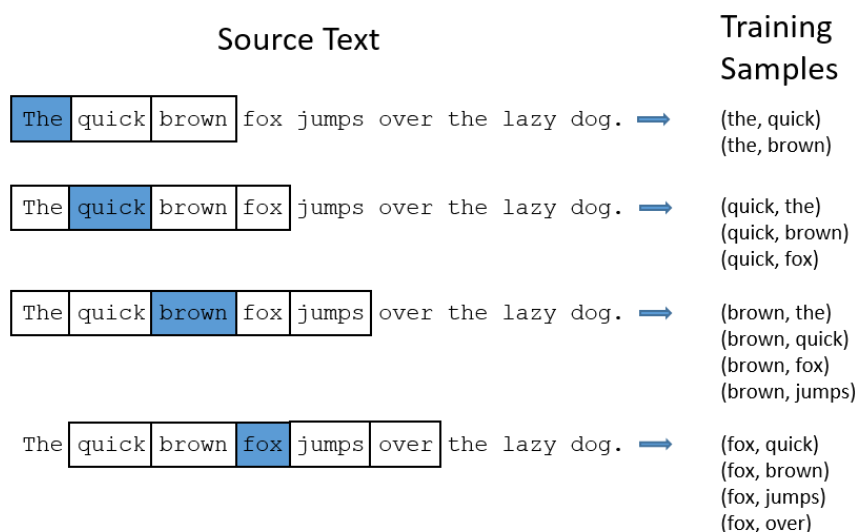
Fonte: Communications of ACM, Probabilistic Topic Models (2012)

Além da possibilidade de classificar os textos como um todo existem algoritmos que permitem analisar as palavras contidas em uma sentença e interpreta-las com o computador, esses algoritmos na verdade são processos de “*embedding*”. Um dos métodos mais interessantes nessa categoria é o Word2Vec por trazer propriedades surpreendentes a sua vetorização das palavras.

Esse algoritmo torna as palavras e m vetores baseando-se nas palavras próximas à palavra que se deseja transformar, dessa forma o modelo utiliza os “vizinhos” de cada palavra para transforma-las em vetor (figura 4). Esse método de *embedding* pelos vizinhos por sua vez que traz as propriedades interessantes desse algoritmo uma vez que palavras comumente utilizadas próximas de outras ficam com vetores semelhantes.

Por exemplo, é possível vetorizar palavras baseado nas letras que a compõe, porém palavras com as mesmas letras como “casa”, “casal”, “asa” e “nasal” apesar de terem as mesmas letras não possuem relação no seu significado, mas teriam vetores muito parecidos. Agora palavras que normalmente estão na mesma posição em uma sentença podem ter, uma vez que de maneira geral sentenças são construídas em uma ordem muito semelhante: sujeito, verbo, ação ou sujeito, verbo, adjetivo.

Figura 4: Exemplo de Embedding com Word2Vec

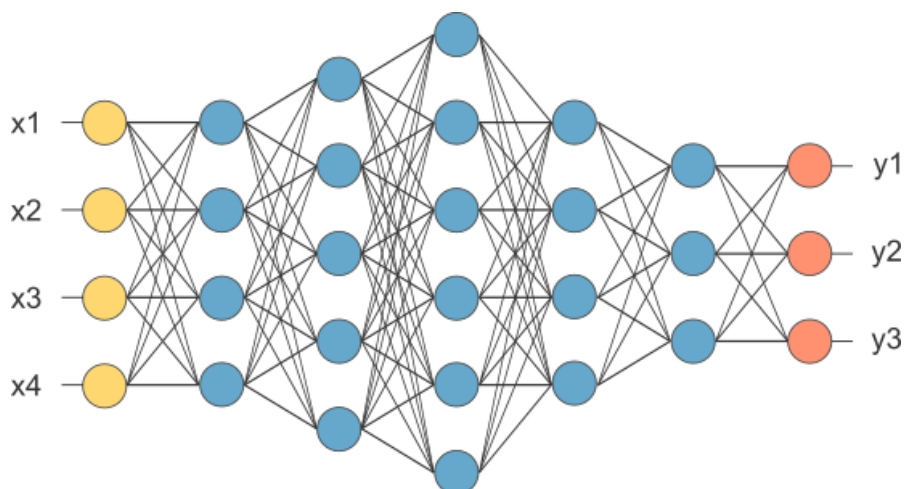


Fonte: <https://mubaris.com/posts/word2vec/>

Dessa forma, esse algoritmo pode ser utilizado para reduzir a dimensionalidade de textos ao mesmo tempo que permite agregar um novo significado as palavras.

Por fim, outros métodos de análise que ganharam muita força nos últimos anos após serem deixados de lado na década de 80 são as redes neurais (figura 5). Esses algoritmos por sua enorme flexibilidade permitem interpretar textos em quase qualquer forma e poderiam ser usados para classificar sentimento ou contexto de forma mais precisa apesar de menos intuitiva e eficiente.

Figura 5: Ilustração Rede Neural

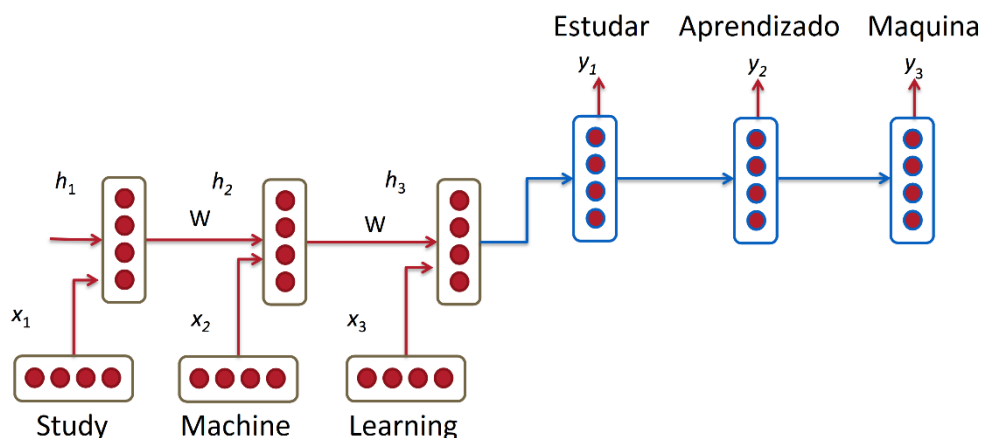


Fonte: <http://www.decom.ufop.br/imobilis/fundamentos-de-redes-neurais/>



Em especial, redes neurais recorrentes tendem a ter um maior sucesso nesse quesito por terem sua estrutura associada a sequencias de dados (figura 6) o que permite interpretar o texto como uma sequência de palavras e não apenas um aglomerado de palavras.

Figura 6: Ilustração de uma RNN para Tradução de Texto

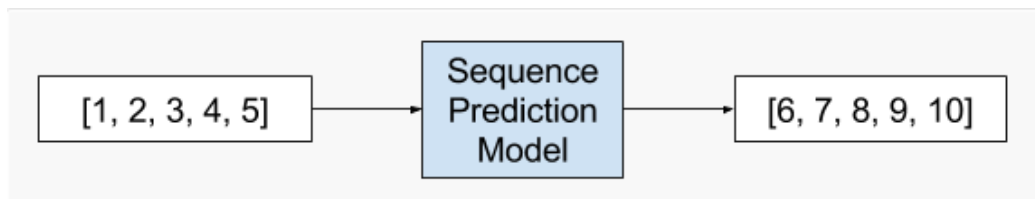


Adaptado de: [www.wildml.com/2015/09/recurrent-neural-networks-tutorial-introduction-to-rnns/](http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-introduction-to-rnns/)

Essa estrutura combinada com células LSTM (Long-Short Term Memory) tem mostrado resultados ainda mais precisos com essas células ajudando a rede a captar a essencial sequência de textos. Estas estruturas são muito comuns à problemas de sequências, mas antes de entender sua aplicação, é necessário um correto entendimento de um sistema de sequências.

Comumente em *machine learning* as amostras treinadas e testadas não possuem ordem específicas, com algum significado dimensional em sua ordenação, mas para sequências isso não é verdade. Nelas é necessário exprimir a correta ordem de observações, esta ordem é importante e precisa ser recebida por um modelo capaz de entendê-la tanto na entrada como na saída. Um exemplo de modelo de predição de sequência pode ser uma simples sequência numérica (figura 7), uma outra aplicação possível, seria um documento em texto com suas palavras ou caracteres vetorizados, tornando a combinação linear de vetores a nova sequência a ser analisada pelo sistema.

Figura 7: exemplo de predição de sequências



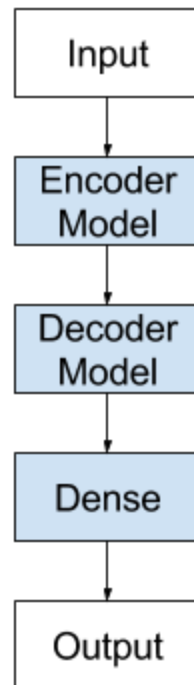
Adaptado de: <https://machinelearningmastery.com/sequence-prediction/>

Por sua característica temporal, as redes recorrentes se apresentam como um bom modelo para atacar problemas sequenciais. Mais especificamente a LSTM possui uma característica de armazenamento do erro propagado, de onde vêm o “*long*” em seu nome, mantendo um erro mais constante. Isso permite que impactos por “decisões” feitas a “mais tempo atrás” possam ser sentidos de maneira mais distribuída pela rede, impedindo um problema comum de gradientes que desaparecem [6]. As RNNs possuem um problema corriqueiro de desaparecimento ou explosão de gradientes pois ao longo de suas sequências temporais é possível obter pesos absurdamente grandes ou pequenos para passos temporais específicos, isso é encarado como um *overfit* e, também, como um impedimento prático para o avanço do treinamento. Ao distribuir temporariamente o erro, ou seja, ao distribuir o erro no tempo, guardando parte de seu valor, e assim, desenvolvendo uma parcela de *long-term-memory*, reduz-se a chance de um peso crescer ou decrescer absurdamente em um dado passo temporal, permitindo que o modelo treine de maneira mais segura [7].

Outra aplicação possível para uma rede recorrente no mundo de processamento de linguagem natural é a o uso de *autoencoders*. Resumidamente um *autoencoder* é uma rede neural que busca aprender uma representação comprimida de um determinado input. Se trata de um método de aprendizagem auto supervisionado, que utiliza métodos comumente supervisionados, para fornecer feedback para si mesmo sem acesso direto à resposta correta. A estrutura pode ser resumida a uma sequência de um *encoder*, onde os dados são “afunilados” seguidos de um *decoder*, onde os pesos são expandidos novamente obtendo-se a mesma distribuição do input inicial. Essa construção, e, especialmente a concordância do output com o input permite que o próprio input seja usado

de target, permitindo o treinamento do modelo, assim o encoder-decoder se torna um autoencoder (figura 8).

Figura 8: Arquitetura Encoder-Decoder



Entre suas aplicações mais comuns, usa-se para redução de dimensionalidade obtendo-se, ao final do *encoder* um resumo das features importantes.

Em 2016 foi introduzida a lógica de redes neurais para caracteres no mundo de processamento de linguagem natural (*character-level neural networks*<sup>[5]</sup>), onde o *embedding*, no mesmo processo de vetorização como o descrito acima, é feito a cada carácter ao invés de cada palavra, como normalmente é feito no word2vec. Primeiramente, ter *embedding* por carácter permite que todas as palavras podem ser reconstruídas como vetores, mesmo se houver uma palavra fora do vocabulário da amostra de treinamento (*out-of-vocabulary words*). Em segundo lugar palavras escritas erradas, emojis e emoticons e novas palavras (novos nomes ou anglicanismos) podem todas ser corretamente convertidas por vetores através da combinação dos vetores dos diferentes carácter que as compõe. Vale ressaltar que os casos listados acima são casos muito recorrentes para textos de tweets, analisados no projeto.

Por fim a última vantagem do *character-level embedding* é que o *embedding* pode ser feito com um menor número de vetores, pois há mais palavras do que caracteres para cada amostra. Isso reduz a complexidade do modelo, e, assim, melhora performance.

Com esse arsenal de modelos descrito seria possível agora encolher alguns métodos para a análise em questão do relatório, as escolhas justificativas e resultados são apresentadas a seguir nas seções Metodologia e Resultados

### **3 Metodologia**

Nessa seção serão detalhados os procedimentos adotados para realizar 3 testes distintos, todos realizados em módulos de python, para criar o modelo desejado de previsão de mercado

A premissa acerca da metodologia utilizada se baseia na percepção de que o que pessoas e empresas falam sobre determinados assuntos influencia diretamente nas ações listadas na bolsa. Isso significa que notícias sobre delações, prisões, escândalos, acidentes ou novos investimentos poderiam, ao chegar em alguma mídia, dar indícios sobre a movimentação do mercado para um período subsequente.

Nesse sentido pode-se pensar que existe duas formas desse fenômeno ocorrer, dado um evento, uma grande massa populacional poderia expressar sua opinião sobre o mesmo, permitindo avaliar o acontecimento como bom ou ruim de uma perspectiva político-econômica, ou, uma pessoa de grande influência em determinado setor (política, economia, esporte) acaba protagonizando um evento relevante, que poderia ser associado as variações de ações subsequentes.

Outro fenômeno que poderia ocorrer seriam os dois cenários de forma sinérgica, em que um político por exemplo realiza uma ação e a massa, rapidamente, expressa sua opinião sobre determinada ação.

### 3.1 Coleta e filtragem dos Dados

Trabalhando com *machine learning*, um dos principais fatores para o sucesso de um modelo é uma grande quantidade de dados. A solução para minerar esses dados contendo opinião das pessoas foi utilizar-se das redes sociais, mais especificamente o Twitter, ambiente virtual em ascensão e com grande adesão de figuras políticas, econômicas, esportistas cuja opinião seria de interesse para o projeto.

As vantagens de utilizar o Twitter além da grande adesão de figuras político-econômicas relevantes seria a velocidade dos posts, extremamente rápida por parte dos usuários, o que normalmente demora mais para acontecer em outras mídias.

Em resumo, a ideia do estudo convergiu para analisar posts no Twitter de pessoas, ou veículos de notícias, que foram consideradas pelo grupo como figuras com grande capacidade de influência afim de analisar se existe algum padrão entre o que está sendo dito e o que acontece com os valores das ações das empresas no mercado.

Antes de implementar os modelos, era necessário primeiro construir os *datasets*. Para isso a API do Twitter foi utilizada fornecendo cerca de 3200 (limite de solicitação por usuário) de cada fonte de notícia considerada relevante, essas foram: OGloboPolítica, folha\_poder, GloboNews, EstadaoPolítica, RevistaEpoca, valoreconomico, g1politica, conexaopolitica, EstadaoEconomia, UOLEconomia, folha\_mercado, g1economia, OGlobo\_Economia, valoreconomico. Outras fontes como VEJA e UOL foram coletadas também, mas resultados preliminares mostraram que ficar restrito a notícias de fontes vinculadas a política e economia produziam melhores resultados.

Após a coleta dos dados foi possível realizar um tratamento em todo o dataset para filtrar pontuação e letras maiúsculas dos textos além de retirar hyperlinks dos posts, adicionando uma variável booleana que indicaria se havia ou não um site, foto ou vídeo atrelado ao post. Com essa filtragem era possível garantir que de maneira geral os tweets seriam compostos apenas por palavras reconhecíveis, esse processo resultou

em um dataset de aproximadamente 44000 tweets para um total de aproximadamente 130 dias de transação na bolsa.

Ao mesmo tempo terminais da Bloomberg foram utilizados para conseguir os dados respectivos à Bovespa sendo que as principais ações (PETR4, VALE3, ITUB4 entre outras) e o próprio índice Bovespa foram coletados.

Com isso foi possível criar os *datasets* de textos e respostas na bolsa necessário para treinar os modelos de classificação de texto e causalidade. Vale destacar que para o primeiro teste abordado nesse relatório, os dados só foram correlacionados com dados do Índice Ibovespa testes formais com outras ações não foram realizados.

### **3.2 Latent dirichlet allocation (LDA)**

Nessa seção são discutidos os procedimentos adotados para se realizar os testes de previsão de bolsa com os modelos de LDA das bibliotecas scikit-learn e ginsin do python.

Para a preparação dos dados, além das etapas de filtragem dos dados, por ser um modelo *Bag-of-Words* exige que antes os dados passem por um *Count Vectorizer* que lista as palavras atribuindo um número para cada palavra do vocabulário encontrada, sendo que todas as palavras, após a filtragem, passaram por esse processo.

É importante destacar que para todas as aplicações, exceto o teste de sanidade, o número de tópicos que seria criado para o conjunto de dados era baseado na matriz de covariância dos tópicos criados sendo que um número alto era estimado e gradativamente reduzido até que nenhum dos tópicos estivesse com covariância muito alta (mais de 0.5 ou menos de 0.5).

#### **3.2.1 Testes de Sanidade**

Ainda antes de testar os dataframes reais o modelo que seria utilizado para encontrar os tópicos nos Tweets foi testado com um dataframe artificial com textos aleatórios contendo palavras sobre carros, frutas e animais de forma a verificar se com um dataset simples com pouco ruído

(apenas algumas palavras repetidas entre os temas) os procedimentos de criação trariam um bom resultado.

Dessa forma, 10000 amostras aleatórias para os temas foram criadas e uma LDA configurada com 5 iterações para gerar 3 tópicos foi testada tanto para o gensim quanto para o scikit-learn.

### 3.2.2 Media de Tópicos

Após o teste de sanidade o primeiro teste no dataset real foi utilizando a LDA da biblioteca gensim do python verificando se os tweets de 24 horas antes da abertura da bolsa conseguiriam prever se a abertura seria com um valor acima, abaixo ou semelhante ao valor do fechamento do dia anterior.

Para isso foi definida uma zona morta na variação do valor da ação sendo que variações abaixo de 0.2% do valor de fechamento seriam consideradas neutras, ou seja, se a bolsa subisse mais de 0.2% do valor do fechamento o dia seria considerado uma subida, se variasse entre -0.2 e 0.2% seria neutro e se variasse menos de -0.2% seria uma queda.

De forma a condensar os dados dos Tweets todos foram classificados de acordo com um modelo LDA treinado em todo o conjunto de dados e em seguida, considerando os dias disponíveis de dados da Bovespa (não inclui finais de semana e feriados) os grupos de 24 horas de tweets forma criados utilizando a média de todos os tópicos discutidos. Além disso de forma a tentar contemplar a variação dos temas discutidos a matriz de covariância dos tópicos também foi adicionada a cada a mostra de forma a passar mais features para o modelo sobre variabilidade dos assuntos discutidos.

Após esse processo foi possível criar o novo dataset onde X era a média e covariância dos tópicos discutidos nas últimas 24 horas e Y a variação da ação na bolsa.

Com a criação dos dataset com a LDA um modelo de *Machine Learning* clássico de classificação como *Decision Tree*, *Randon Forest*, *Support Vector Classifier* e Regressão logística era utilizado para ver se a previsão

com os tópicos era possível, para isso o método de validação cruzada era utilizado de forma que o número de dados já pequenos 127 não precisasse ser ainda mais reduzido por uma separação treinamento teste.

### **3.2.3 Media de Tópicos + Variação da Bolsa para escândalos**

Outro teste realizado no mesmo modelo do teste anterior foi utilizar o mesmo X do dataset criado no modelo anterior, porém agregando dados de variação do mercado observados para os últimos cinco dias. Em relação ao Y as margens da zona morta foram modificadas para que apenas grandes variações de preço fossem detectadas de forma a garantir que apenas acontecimentos relevantes pudessem ser detectados. A intenção dessa abordagem era verificar se com dados da bolsa e das notícias o modelo teria mais sucesso em entender o que está acontecendo no mercado e como as notícias estão influenciando esse comportamento, e ao verificar quebras de padrão, talvez prever grandes oscilações na bolsa.

Dessa forma, se criou um novo dataset onde X era a média e covariância dos tópicos discutidos nas últimas 24 horas e variação da bolsa nos últimos 5 dias e Y a variação da ação na bolsa em um primeiro teste, e o valor absoluto da bolsa no segundo teste.

Com a modificação dos dados, foi obtido um dataset com 10 subidas drásticas, 7 descidas drásticas e 112 dias relativamente normais, que apesar de constituir poucos dados, seria aceitável para verificar crises de mercado.

Novamente com a criação dos dataset com a LDA um modelo de *Machine Learning* clássico de classificação como *Decision Tree*, *Random Forest*, *Support Vector Classifier* e Regressão logística foi utilizado para ver se a classificação com os tópicos era possível. Além disso, como na segunda abordagem o alvo do modelo era o valor absoluto, modelos de regressão tiveram que ser utilizados sendo alguns deles o *Linear Regressor*, *Support Vector regressor* e o *Stochastic Gradient Regressor*.



### 3.2.4 Tópicos com avaliação individual

Depois do experimento inicial se constatou que fazer as medias dos tópicos não era apropriado uma vez que mesmo filtrando os tweets para emissoras de notícias de política e economia temas irrelevantes ainda apareciam com certa frequência. Por exemplo alguns tweets de emissoras eram literalmente “bom dia pessoal” não transmitindo nenhuma informação relevante e atrapalhando na média geral dos temas. Dessa forma a solução mais simples para um projeto de 1 mês, uma vez que filtrar temas de 44000 tweets é extremamente demorado e repetitivo se optou por separar os tweets e classifica-los em relação a bolsa um por um utilizando a LDA do gensim.

Após esse processo foi criado um novo dataset onde X eram os tópicos de cada Tweet, seu número de likes, retweets e autor e Y a variação da ação na bolsa.

Outra vantagem dessa abordagem é que tanto o autor dos tópicos, quanto sua repercussão na rede também entravam como features fornecendo mais informações sobre aquela noticia em especifico.

Novamente com a criação dos dataset com a LDA um modelo de *Machine Learning* clássico de classificação como *Decision Tree*, *Randon Forest*, *Support Vector Classifier* e Regressão logística era utilizado para ver se a previsão com os tópicos era possivel.

### 3.2.5 LDA + Word2Vec

Após a tentativa de classificação individual dos Tweets também foi aplicado uma combinação dos métodos de LDA e word2Vec para que palavras com sentidos muito parecidos pudessem ser agrupadas em outras features que o modelo poderia observar.

Para isso uma LDA de 20 tópicos foi treinada com os dados e um Word2Vec com 100 features também sendo que para a geração do dataset, as features eram na verdade a medida dos vetores gerado pelo Word2Vec para cada palavra do Tweet.

Isso permitiu criar um novo dataset semelhante ao do último método, mas com features que refletiam mais ou menos os tipos de palavras contidos no tweet além do tópico do tweet. Assim, o novo dataset criado tinha como X os tópicos de cada Tweet, a média do vetor de palavras, seu número de likes, retweets e autor e em Y a variação da ação na bolsa.

Novamente com a criação dos dataset com a LDA e o Word2Vec um modelo de *Machine Learning* clássico de classificação como *Decision Tree*, *Random Forest*, *Support Vector Classifier* e Regressão logística era utilizado para ver se a previsão com os dados era possível.

### **3.3 Redes Neurais**

Nesta seção são demonstrados os passos para construção de diferentes iniciativas de leitura dos tweets através de redes neurais. Como abordado na leitura prévia, a principal escolha para análise de sequências, no caso tweets, foram as redes neurais recorrentes, mais especificamente, long-short-term-memory, LSTM. Para todas os pré-processamentos e construções de redes nesta secção a biblioteca keras do tensorflow foi utilizada.

#### **3.3.1 LSTM prediction**

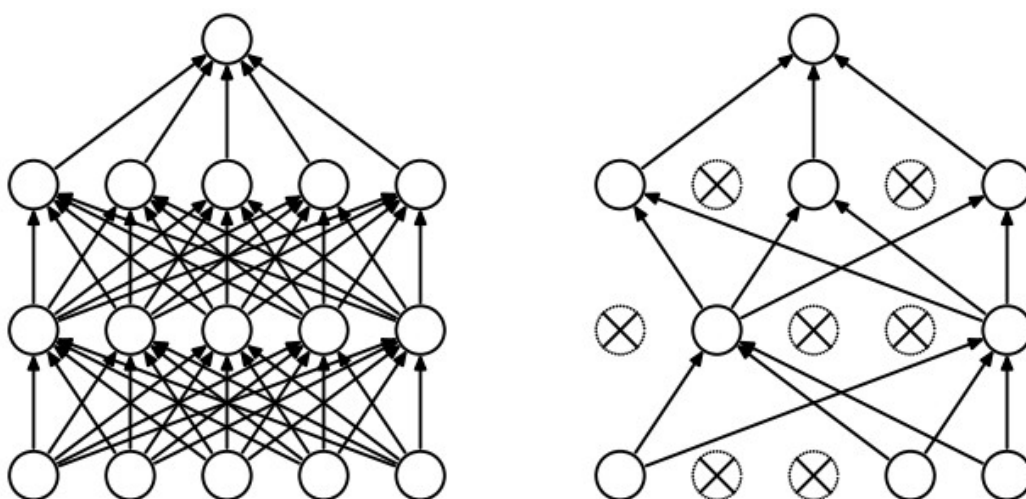
Primeiramente é necessário converter o texto para uma linguagem numérica, na qual os neurônios serão conectados. Para diferenciar o desenvolvimento de word2vec e lda2vec abordados na seção anterior, e, também, pelos diferentes benefícios descritos anteriormente, foi escolhida a abordagem em nível de carácter (*character embedding*).

Utilizando a função tokenizer do keras, os vocabulários de caracteres foram definidos. Um ponto de interesse é a utilização da lista completa de vocabulários encontrados, ao invés de ser substituída por uma lista de vocabulários pré-montada, isso permite que emojis e emoticons sejam considerados também. Com a função tokenizer do keras as sequências são geradas a partir do treinamento em cima do dataset de tweets, agora se tem uma combinação de vetores que é legível por uma rede neural. Como os tweets não possuem o mesmo número de caracteres, é

necessário fazer um *padding* garantindo que todos os tweets possuam o mesmo comprimento de caracteres de 140, evitando erros de input na rede.

Para a arquitetura de redes foi adotado um modelo “sanfona”, como é visto na figura 5, onde o input de 140 é iniciado por uma camada de embedding configurada com um tamanho de 20, ou seja, para os 140 caracteres encontrados em cada um de todos os tweets da amostra, um vetor de dimensão 20 será gerado. Esta camada é seguida por uma LSTM de 256 neurônios, e conectados a uma Dense de 512. Para fazer a junção de uma LSTM com uma Dense foi usada uma camada de Flatten, que retira o componente temporal da LSTM, esta alteração é importante pois a classificação final será feita com uma redução de camadas de Denses até alcançarem apenas 3 neurônios classificatórios na camada de output. A redução entre camadas não é feita somente conectando uma camada Dense maior a uma menor, mas através de uma camada de Dropout (figura 9), esta camada permite o cancelamento aleatório de alguns neurônios ao longo do processo de geração de pesos entre as diferentes camadas, isto é muito interessante para o processo pois dificulta a “saturação” de algumas pesagem (desaparecimento ou explosão de gradientes) ao cancelar alguns neurônios e forçar novas conexões.

*Figura 9: Uma rede Dense à esquerda e uma rede Dense com Dropout à direita*



Por último a camada final, ou camada de output, é um Dense de 3 nodos com uma ativação classificatória, isso significa que ao final das

sequências de camadas com seus respectivos pesos o modelo realizará uma classificação entre três possibilidade de classes e ao final às comparará com o a sequência de target para cada tweet. Para a ativação classificatória na última camada foi usada a função de ativação *softmax*, muito similar à uma regressão logística, mas permitindo um sistema multi-classe.

### 3.3.2 LSTM Autoencoder

A metodologia escolhida para o *autoencoder* foi de um reconstutivo. Isso significa que a rede neural aprende a reconstruir a sequência de input ao longo de seu treinamento. Inicialmente é definida a sequência de input pela mesma metodologia de *character-level embedding* descrita no item anterior.

Para a arquitetura da rede que segue a camada de *embedding* é iniciada com uma camada de LSTM que recebe as sequências dos tweets, esta é configurada para não retornar sequências, isto significa que seu output é um vetor único de tamanho igual ao número de nódulos em sua camada. No final de uma LSTM de tamanho menor que o input não retornando sequência temos um *encoder*, pois toda a informação retida no input da camada da rede recorrente foi condensada em um vetor único de saída.

Como iremos montar o *autoencoder* como uma arquitetura *encoder-decoder*, é preciso em seguida montar a estrutura do *decoder*. Para isso é necessário construir um input para a LSTM *decoder*, isso é feito usando uma camada RepeatVector, que recebe o vetor de saída do *encoder* e o repete para a dimensão original do input, permitindo, então, o treinamento dos diferentes nodos da LSTM *decoder*. Esta camada final se trata de uma LSTM configurada para retornar as sequências esperadas.

É esperado ao final do treinamento um modelo *encoder-decoder*, capaz de, entre outras tarefas, ser um redutor de dimensionalidade para tweets de mesma natureza que o dataset.

## 4 Resultados

Nessa seção são apresentados os dados relativos às etapas da metodologia, sendo que cada modelo apresentou resultados consideravelmente diferentes.

#### 4.1 Resultados Filtragens

Alguns dos dados iniciais mais interessante que se pode obter são o tamanho dos tweets e a distribuição inicial das palavras. Para isso histogramas com a frequência de cada comprimento de tweet e com a frequência de cada palavras puderam ser criados. No histograma com o comprimento dos tweets (figura 10) foi possível verificar que a distribuição parece a junção de duas normais, uma com média em 12 palavras e outra com a média em 18 palavras, isso permitiria estipular que existem dois grupos principais de tweets. Porém, observando as medias de comprimento de tweets por fonte de noticia é possível perceber que na verdade essas duas normais refletem os diferentes perfis das emissoras que tendem a ficar em torno de 13 palavras por tweet ou 17 palavras por tweet.

Figura 10: Histograma do número de palavras nos Tweets

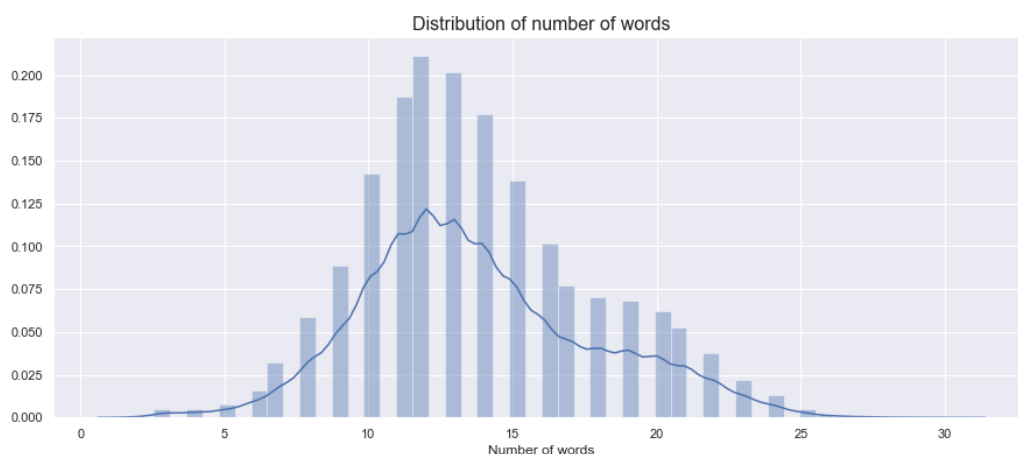


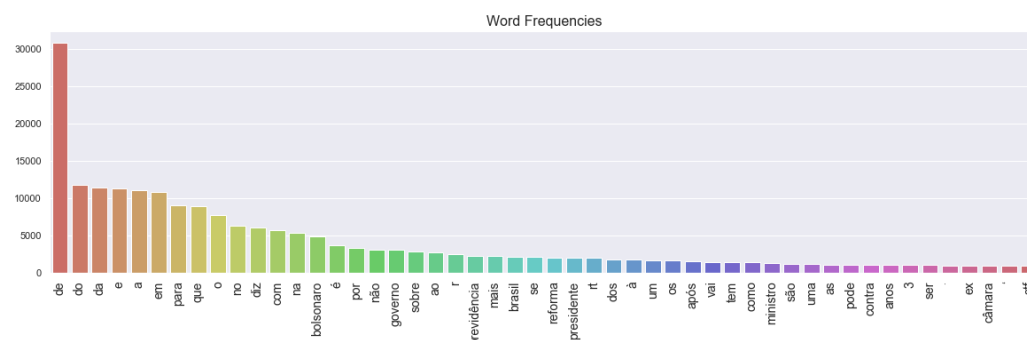
Tabela 1: Número medio de palavras por Tweets para cada fonte de noticia

Fonte	Número de Tweets	Comprimento do Tweet
OGloboPolitica	3200	14.51
folha_poder	3200	12.93
GloboNews	3193	18.65

EstadaoPolitica	3085	13.33
RevistaEpoca	3195	17.75
valoreconomico	3197	10.80
g1politica	3200	14.25
conexaopolitica	3199	15.48
EstadaoEconomia	3200	12.90
UOLEconomia	3200	13.02
folha_mercado	3200	12.47
g1economia	3200	13.25
OGlobo_Economia	3199	12.48

Já observando o gráfico de frequência de palavras (Figura 11) é possível observar que as palavras mais faladas são principalmente preposições e artigos como: de, da, dos, e, a, o, em, nos. Essas palavras por si só transmitem muito pouco significado e, para os testes com LDA's serão filtradas através de uma função já implementada nas bibliotecas utilizadas de StopWords para que não sejam criados tópicos de preposições por exemplo. Os acentos e números também serão filtrados para os testes com LDA uma vez que, do ponto de vista de arquitetura de tópicos, não transmitem nenhum significado relevante.

*Figura 11: Gráfico de frequência das 50 palavras mais utilizadas no dataset*

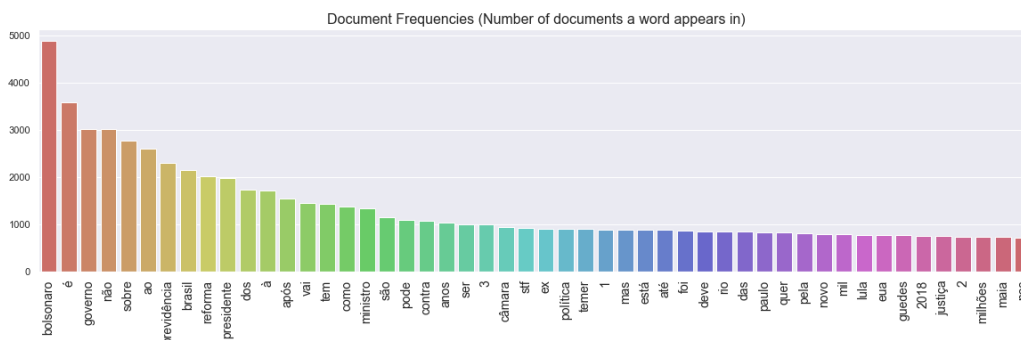


## 4.2 LDA

Nessa subseção são discutidos os resultados relativos as aplicações de LDA ao para classificação dos tweets. Porém, antes disso alguns aspectos sobre filtragem de dados serão discutidos fora desses tópicos por serem comuns a todos os modelos.

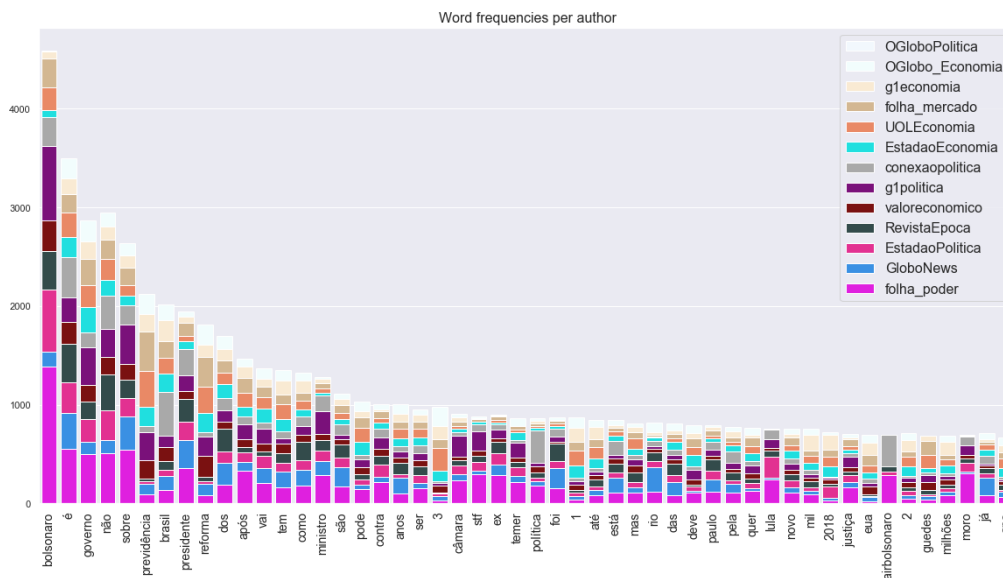
Nesse contexto, como foi citado anteriormente, a utilização de *StopWords* permitiu remover a maioria das palavras irrelevantes fazendo com que um novo gráfico de frequência pudesse ser criado. Nesse gráfico é possível observar que a palavra mais utilizada no dataset é o nome do atual presidente do país, seguida de governo, reforma, previdência, ministro entre outras que indicam o caráter político-econômico das notícias mostrando que a seleção de fontes de informação relacionadas ao tema foi bem-sucedida (figura 12).

Figura 12: Gráfico de frequência das 50 palavras mais utilizadas para uso na LDA



Ainda no contexto de frequência das palavras utilizadas outro resultado interessante é a distribuição das palavras por autor dos tweets, observando-se o gráfico na figura 13 é possível observar que a “folha\_poder” é responsável por quase 1/3 das utilizações do nome do atual presidente indicando que essas frequências não são iguais para todas as fontes de informação.

Figura 13: Frequência das palavras por autor



#### 4.2.1 Teste de sanidade

Antes de se testar os modelos de classificação no dataset de palavras foi realizado um teste de sanidade com um dataset artificial de baixo ruído para verificar se a estrutura de tratamento de dados (remoção de pontuação, letras maiúsculas e *stopwords*) não influenciava na qualidade dos tópicos criados. Para isso tanto a LDA da biblioteca gensim quanto a do scikit-learn foram testadas e ambas apresentaram resultados consideravelmente bons para o modelo.

Porém, ao longo desse teste se verificou também que a LDA realizada pela biblioteca do gensim era consideravelmente mais otimizada (cerca de 10 vezes mais rápida), e após testes no dataset de tweets se verificou que a qualidade dos tópicos mesmo fora do ambiente controlado do teste de sanidade era praticamente igual.

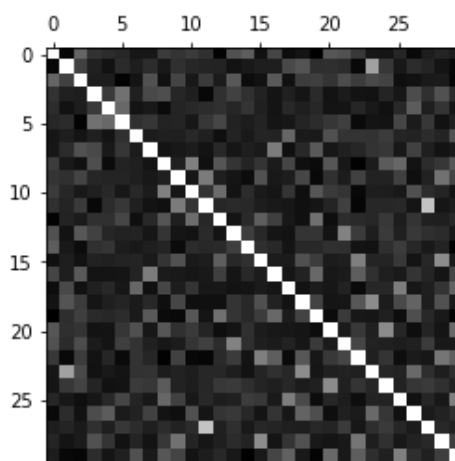
Dessa forma, como o projeto exige um número muito grande de teste de diversas LDA's se optou por utilizar a LDA do gensim para os testes realizados no presente relatório.



### 4.2.2 Media dos tópicos

Nesse modelo inicial que utilizava a média dos tópicos para previsões de 24 horas da bolsa o primeiro resultado relevante é que o número de tópicos que apresentou uma baixa correlação foi de 30 tópicos com a distribuição apresentada na figura 10.

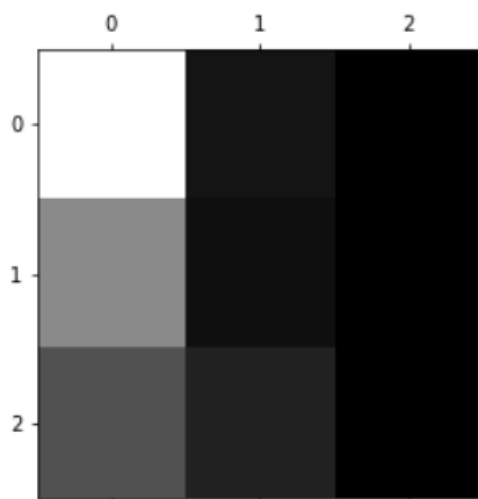
Figura 14: Matriz de correlação dos tópicos da LDA utilizada para média dos tópicos (mais escuro representa menor valor absoluto)



Com esse processo de classificação dos tópicos realizado gerando listas de vetores de 930 pontos de dados a melhor acuracia que pode ser obtida na previsão foi de 55% com um modelo de *Random Forest* com 100 estimadores e índice de impureza de  $1e-6$ .

Infelizmente, como os dados estavam mal distribuídos entre neutros, subidas e descidas (50%, 28%, 22% respectivamente), esse resultado supera o baseline de 50% (chutar todos como neutro) por apenas 5%, sendo um modelo consideravelmente impreciso cujas previsão tende a ser sempre zero exceto em raras exceções como pode ser observado na figura 15 da matriz de confusão das predições.

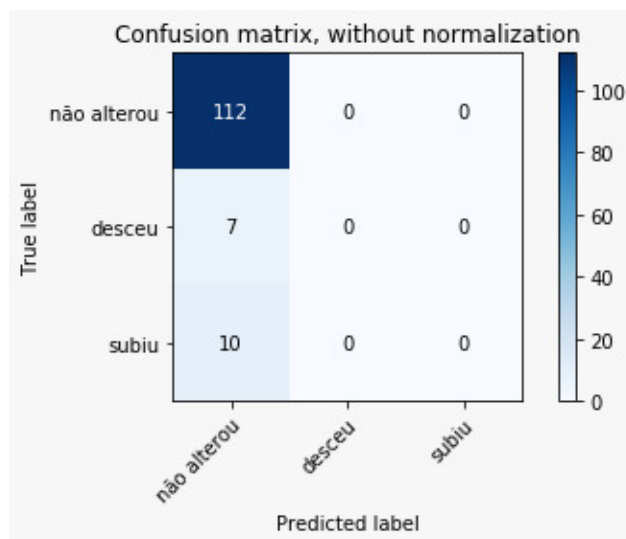
Figura 15: Matriz de confusão do teste com média dos tópicos em 24 horas



#### 4.2.3 Média dos tópicos + Variação da bolsa para escândalos

A partir dos testes foi possível verificar que o primeiro modelo novamente foi totalmente ingênuo dado que 100% das previsões foram em não oscilação, atingindo sempre o baseline de acurácia de 84%. Analisando a fundo a proposta desse modelo, percebe-se para prever “crises” ou seria preciso muito mais dados uma vez que esses eventos outliers não se repetem num curto espaço de tempo. Para os dados treinados na configuração já descrita acima, resultou-se na matriz de confusão mostrada na figura 16. Na qual o modelo treinado basicamente, sempre prevê que não vai haver uma variação relevante no preço da opção.

Figura 16: Matriz de confusão do teste com média dos tópicos em 24 horas e com variações dos últimos 5 dias da Bovespa



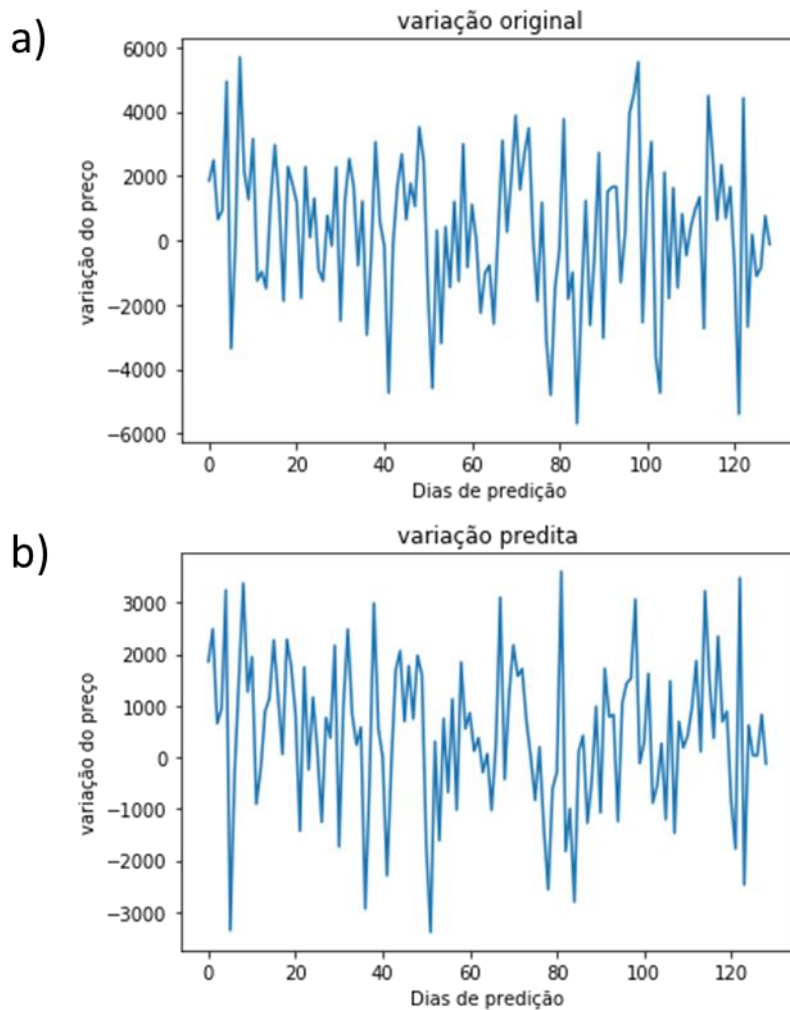
A segunda hipótese foi uma contramedida da ingenuidade da primeira ao utilizar um valor absoluto ao invés de uma classificação. Como visto no modelo anterior, o próprio baseline do problema já é bem alto, dessa forma foi decidido que o desafio seria tentar prever da melhor forma possível como esse conjunto de tweets poderia auxiliar a descobrir qual a variação em reais de uma opção depois de X dias, ou seja, o modelo apesar de ter a mesma quantidade de amostras (129), teria mais detalhes dentro de cada input para poder tomar sua decisão. Com essa modificação de variável independente os seguintes resultados foram obtidos em relação à raiz do erro medio quadrático das previsões, ou “*Root Squared Mean Error*” (RSME) do inglês:

Tabela 2: Resultados dos modelos para previsão do valor da bolsa com 24 horas de tweets

Modelo	Precisão (RSME)
SGDRegressor	535.48,
LinearRegressor	477.65,
SVR	502

Com os modelos também foi possível gerar gráficos das variações da Bovespa real e da Bovespa predita sendo que o *Linear Regressor* foi adotado por possuir um erro RSME menor. A figura 17 ilustra o resultado da previsão.

Figura 17: Variação da Bovespa nos dias de teste, a - original, b - predição

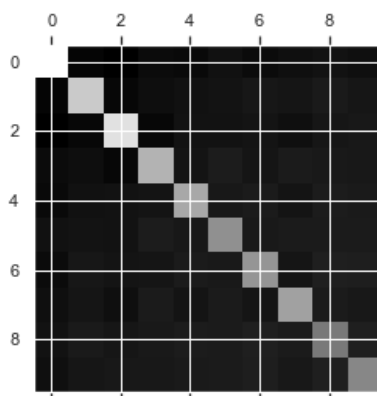


Na figura 17a têm-se as reais variações observadas na IBOVESPA, na figura 17b têm-se as variações previstas pelo modelo. Uma análise superficial desses dois gráficos relata uma certa similaridade entre ambos. Entretanto somente 4% das vezes o modelo realmente variou na direção certa (variação negativa ou positiva). Inclusive a escala de ambos os gráficos é bem distinta, levando a conclusão de que até mesmo o output mais parecido dentre os modelos testados, está bem longe da realidade esperada. Por fim, a partir de uma comparação mais detalhada do resultado foi possível verificar que o modelo estava basicamente chutando o valor do dia anterior, sendo uma cópia relativamente boa dos valores originais, mas com um offset de um dia justificando a acurácia no sinal das variações tão ruim.

#### 4.2.4 Tópicos com avaliação individual (Comparação de fontes)

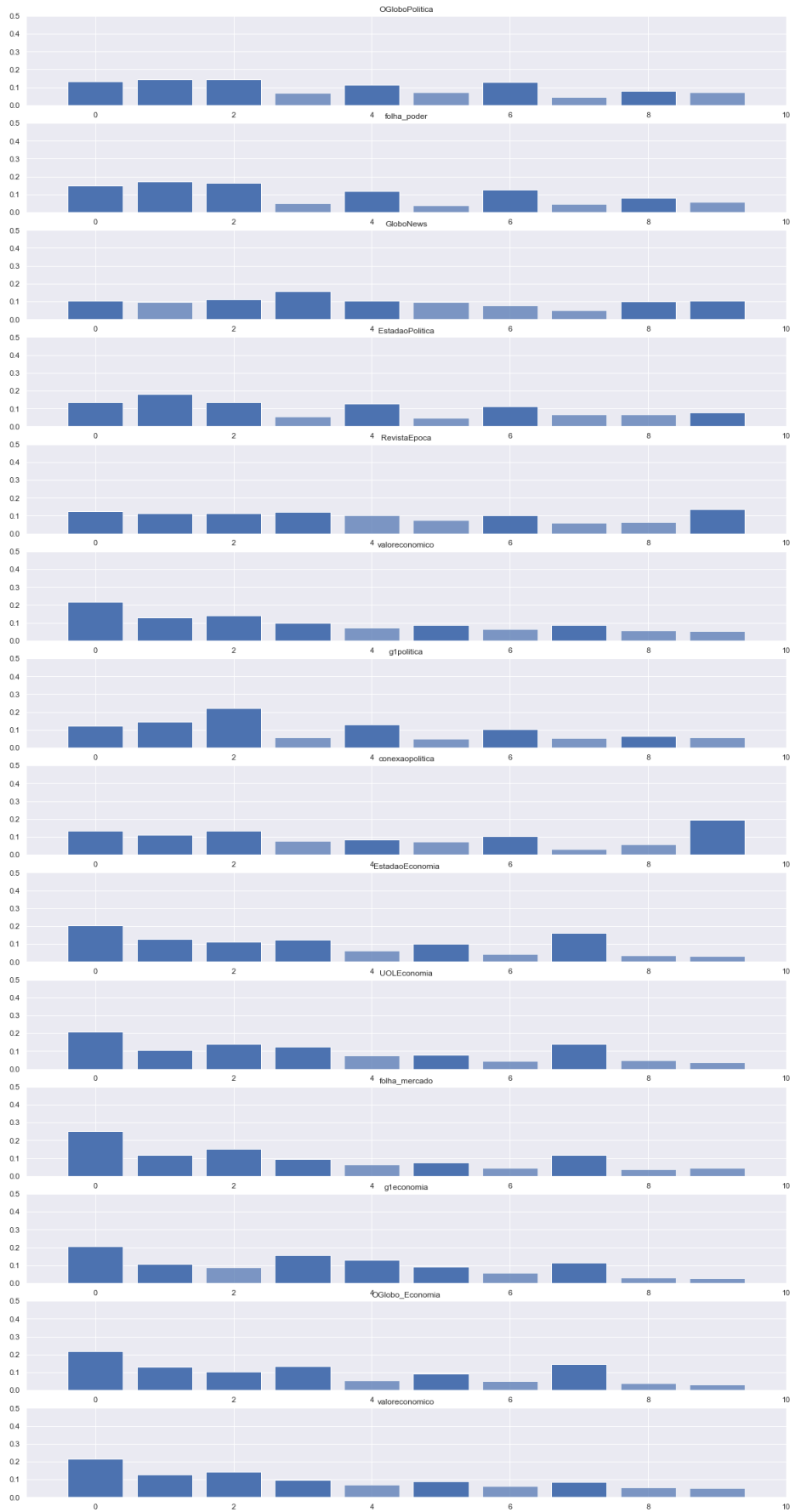
No primeiro teste com a LDA do ginsin com avaliação individual, se optou por utilizar menos tópicos para permitir uma interpretação visual da distribuição de tópicos de cada fonte de notícia de forma que fosse possível verificar se existe um padrão entre algumas fontes e os tópicos em que são classificados os tweets, mas, novamente, a matriz de correlação foi utilizada para verificar se os tópicos não eram complementares, e pela figura 18, pode-se verificar que não foram.

*Figura 18: Matriz de correlação dos tópicos da LDA utilizada para análise individual dos tópicos (mais escuro representa menor valor absoluto)*



A partir da LDA foi possível construir uma sequência de histogramas, sendo um para cada fonte, e comparar como em média as notícias de cada fonte se distribuí entre eles. A figura 13, mostra os resultados obtidos por essa verificação.

Figura 19: Histogramas da distribuição média de tópicos por fonte de notícia



A partir da figura, foi possível verificar que de fato existem relações entre as fontes de notícias, especialmente entre as duas primeiras (oglobo\_politica e folha\_poder) e as fontes 9, 10 e 11 (UOL\_economia, Folha\_mercado e g1economia) foi possível perceber que a distribuição de tópicos foi muito semelhante sendo que os picos do primeiro grupo tratam principalmente do terceiro tópico, e do segundo grupo principalmente do primeiro tópico.

Observando as palavras desses dois tópicos é possível verificar que de fato o primeiro tópico está mais relacionado com a economia com palavras como “previdência, reforma, Petrobrás, Vale, preço, EUA, ações, mercado” entre as primeiras 25 palavras enquanto o terceiro tópico está mais vinculado à política com palavras como “governo, previdência, Bolsonaro, ministro, congresso, STF, mourão, CCJ, militares, câmara” estando em suas primeiras 25 palavras como pode ser verificado na lista abaixo.

Palavras Topico 0: bolsonaro, previdência, é, brasil, governo, reforma, petrobras, não, após, nova, vale, mercado, sem, contra, mundo, sobre, à, novo, vai, eua, tem, ' 'preço, moro, anuncia, como, ao, 2018, brumadinho, são, deve, dos, ações, nas, pode, mas, diesel, plano, pela, presidente, ser, das, trump, ministro, dados, paulo, lança, anos, trabalho, quer, entre

Palavras Topico 2: sobre, governo, reforma, previdência, não, bolsonaro, está, ministro, congresso, política, mourão, é, guedes, ao, stf, posse, ccj, militares, câmara, ser, presidente, pede, dos, vai, após, projeto, pode, foi, brasil, das, decreto, proposta, quer, secretário, orçamento, inquérito, ter, deputados, à, pec, equipe, maia, até, moro, temer, senado, votação, acordo, análise, deve'

Essa análise permitiu concluir não só que algumas fontes de notícia tendem utilizar vocabulários semelhantes e fazer notícias de assuntos semelhantes como mostra que essa classificação por tópicos permitiria distinguir qual o principal tema das fontes de notícia baseado em quais tópicos são tratados. A partir dessa LDA por exemplo os tópicos 0, 3, 7 e

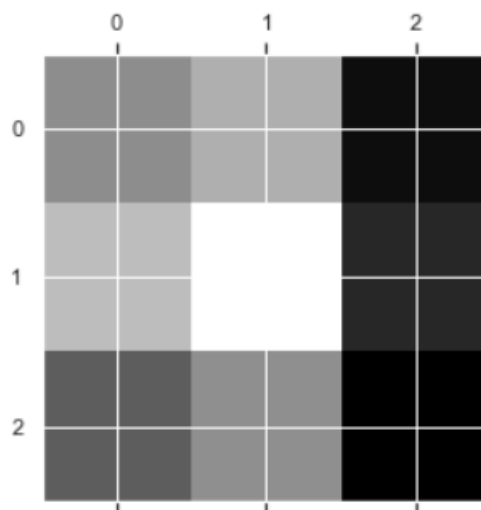
9 parecem ser mais relacionados com economia, enquanto os tópicos 1, 2, 4 e 8 parecem estar mais relacionados com política.

Após essa conclusão interessante, o mesmo modelo de 10 tópicos foi associado aos dados da Bovespa e modelos de *Machine Learning* clássicos utilizados para verificar se alguma relação poderia ser encontrada entre os tópicos, número de retweets, likes e autor dos textos e a variação da bolsa.

Dessa vez o resultado foi uma acurácia de 34% na classificação, entretanto, para esse experimento, os tópicos haviam sido distribuídos de forma mais adequada por uma modulação da zona morta assimétrica que deixou o dataset com 37% de neutros, 33% de decidas e 30% de subidas sendo que na verdade 34% seria o baseline do modelo.

Porém, foi interessante observar que ao invés do modelo focar em apenas uma condição, como os dados são distribuídos de forma mais igualitária a forma da matriz de confusão dos dados (figura 20) possui uma distribuição muito mais esparsa que a obtida no ultimo experimento.

Figura 20: Matriz de confusão do teste com 10 tópicos tweet por tweet



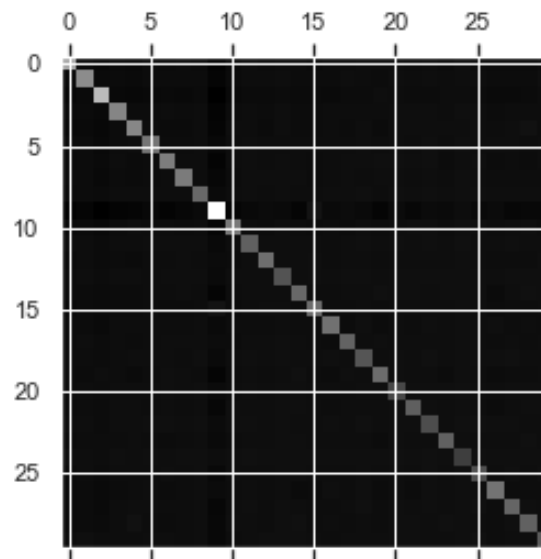
#### 4.2.5 Tópicos com avaliação individual

Após o teste do experimento anterior um novo teste utilizando mais tópicos foi realizado de forma a verificar se tópicos mais refinados sobre os temas trariam um resultado mais adequado ao dataset, nesse caso a



matriz de covariância foi utilizada para garantir que não existissem tópicos muito semelhantes ficando com o resultado apresentado na figura 21.

Figura 21: Matrix de covariância do teste com LDA com 30 tópicos para tweets individuais

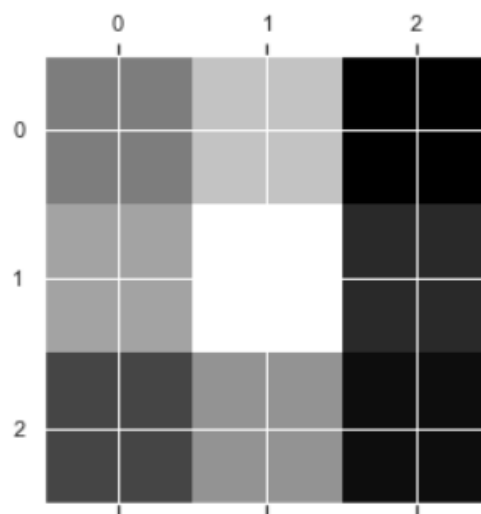


Após essa verificação o modelo foi associado aos dados da Bovespa e modelos de *Machine Learning* clássicos utilizados para verificar se alguma relação poderia ser encontrada entre os tópicos, número de retweets, likes e autor dos textos e a variação da bolsa.

Porém, novamente o resultado foi negativo com uma acuracia de 35% na classificação, sendo 39% o baseline do modelo para chutes sempre em neutro.

Dessa forma, foi possível verificar que mesmo adicionando mais tópicos a distribuição encontrada pelos algoritmos clássicos não melhorou. Uma verificação da matriz de confusão do modelo confirma esse resultado muito semelhante do anterior.

Figura 22: Matriz de confusão do teste com 30 tópicos tweet por tweet



#### 4.2.6 LDA2Vec

Após o teste final com LDA, a implementação do word2vec tinha o objetivo de trazer mais features para o modelo de forma que novas relações pudessem ser encontradas pelos algoritmos clássicos testados para criar as correlações.

Um efeito interessante dessa aplicação, no entanto, foi a criação de um “dicionário” político econômico do Brasil devido às propriedades dos vetores do word2vec. Om ele foi possível encontrar palavras semelhantes do ponto político-econômico e encontrar como elas se organizam em relação ao seu contexto medio.

A figura 23 e 24 mostram dois exemplos de palavras similares e dois de busca por contexto para ilustrar como esse “dicionário” funciona, e algumas relações encontradas por ele.

É Interessante observar, especialmente pelos exemplos da figura 24 que de fato o “dicionário” funciona, vinculando por exemplo a empresa vale às notícias vinculadas a empresa no periodo estudado, incluindo palavras como brumadinho, barragem, e rompimento.

Figura 23: Exemplos do dicionário de Word2Vec para "morão" e "reforma, previdência, câmara, bolsonaro"

```
w2vmodel.wv.most_similar("morão")
[('fhc', 0.8229206800460815),
 ('bebianno', 0.8019016981124878),
 ('interpretado', 0.8003249168395996),
 ('helena', 0.7902745008468628),
 ('embaixada', 0.7719518542289734),
 ('explicar', 0.7666263580322266),
 ('olavo', 0.7646915316581726),
 ('áudio', 0.7640225291252136),
 ('bolsonaros', 0.7590665817260742),
 ('marina', 0.7588032484054565)]

w2vmodel.predict_output_word(["reforma", "previdência", "câmara", "bolsonaro"])
[('aprovar', 0.0019693756),
 ('pec', 0.0017059752),
 ('previdência', 0.0016497344),
 ('proposta', 0.0015357264),
 ('votar', 0.0014927655),
 ('aprovação', 0.0014642773),
 ('reforma', 0.0013776376),
 ('votação', 0.0013151352),
 ('aprovada', 0.0012833914),
 ('texto', 0.0012492477)]
```

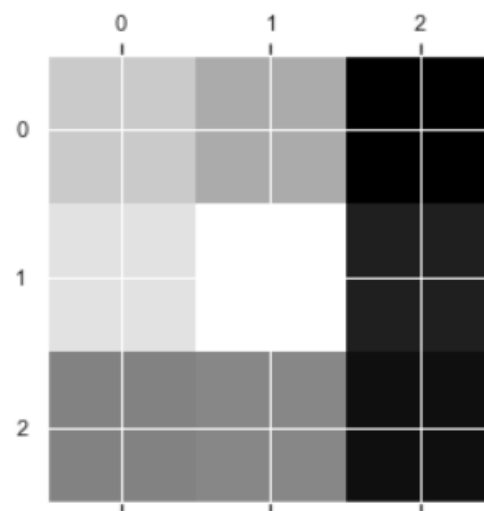
Figura 24: Exemplos do dicionário de Word2Vec para "vale" e "petrobras, caminhoneiros, diesel"

```
w2vmodel.wv.most_similar("vale")
[('brumadinho', 0.7475146055221558),
 ('barragem', 0.7413856983184814),
 ('rompimento', 0.7244623899459839),
 ('schvartsman', 0.6959048509597778),
 ('mina', 0.685188889503479),
 ('tragédia', 0.6843151450157166),
 ('engenheiros', 0.6806573867797852),
 ('desastre', 0.6778314709663391),
 ('mg', 0.6655687093734741),
 ('rompeu', 0.6650184392929077)]

w2vmodel.predict_output_word(["petrobras", "caminhoneiros", "diesel"])
[('preço', 0.0077047176),
 ('petrobras', 0.006490294),
 ('diesel', 0.0064040497),
 ('eleva', 0.0036783705),
 ('gasolina', 0.0035026644),
 ('refinarias', 0.0022892444),
 ('médio', 0.0021490697),
 ('recua', 0.0017782422),
 ('reajuste', 0.0017356029),
 ('preços', 0.0017265801)]
```

Em relação ao modelo preditivo em si, o resultado foi novamente ruim, a acurácia do melhor preditor (*random forest*) foi cerca de 37%, o que indica uma melhora em relação ao com LDA puramente (35%), mas ainda representa um resultado dentro da baseline.

Figura 25: Matriz de confusão do teste com 20 tópicos tweet por tweet e um Word2Vec com 100 features



Dessa forma, o resultado do “dicionário” foi um efeito colateral muito interessante, mas em relação ao resultado geral em relação a previsão foi consideravelmente ruim.

### 4.3 Redes Neurais

Em relação às aplicações de redes neurais infelizmente não houveram resultados relevantes até o prazo final do projeto devido a limitações de hardware e tempo para treinamento das redes.

Ainda assim, todas as etapas de implementação foram concluídas e redes neurais ainda que não completamente treinadas puderam ser criadas para os testes de implementação da RNN e do *Autoencoder*.

## 5 Conclusões

A partir do projeto foi possível apreender bastante sobre os métodos de interpretação de linguagem natural com os estudos feitos, porém foi possível observar que não houve nenhum resultado relativo a capacidade de previsão dos tweets satisfatório.

Para explicar essa falta de relação encontrada 3 hipóteses foram levantadas. A primeira é que os tweets poderiam de fato não conferir informações importantes relevantes sobre a bolsa, ou que o fazem com um delay consideravelmente alto. A segunda seria que apenas alguns

tweets conferem informações relevantes, mas como estão imersos em grupos de muitos tweets irrelevantes os resultados poderiam ficar distorcidos. E por fim, a terceira poderia ser que os modelos implementados não conseguiam extrair informações relevantes dos textos uma vez que o emprego de diversas palavras poderia ser usado para gerar notícias boas e ruins sobre a empresa dependendo apenas de sua ordem ou duplo sentido.

Um exemplo da terceira hipótese seriam as duas notícias abaixo (criadas para efeito do exemplo): “Barreira da empresa Vale rompe e milhares tem expectativa de que a empresa quebre” e “A empresa Vale rompe barreiras e quebra milhares de expectativas”

Dessa forma, o projeto não obteve o resultado esperado, mas alguns efeitos colaterais dos modelos implementados trouxeram resultados surpreendentes.

A criação do “dicionário” político-econômico do Brasil por exemplo, foi um subproduto do projeto que poderia se tornar um foco de estudo vinculando palavras com outras por causa de notícias e não significados. A palavra “água” por exemplo ao invés de estar vinculada a rios e mares ficou vinculada a Santa Catarina no dicionário, devido a falta de água em partes da cidade durante 5 dias no litoral sul do estado no início de janeiro de 2019.

A possibilidade de classificar fontes de notícia baseado nos tópicos normalmente descuidados das suas manchetes também foi um resultado muito interessante que possibilitaria a criação de um classificador de jornais completamente imparcial.

Assim, apesar do resultado final do classificado do projeto não ser um sucesso total, o projeto possibilitou a criação de ferramentas interessantes, que poderiam ser exploradas para obter outros estudos de análise político-econômica e midiática com interpretação por linguagem natural.

## 6 Referências Bibliográficas

1. Akshay Kulkarni, Adarsha Shivananda. "Natural Language Processing Recipes: Unlocking Text Data with Machine Learning and Deep Learning Using Python", Apress, 2019
2. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, "Introduction to Information Retrieval", Cambridge University Press. 2008.
3. Daniel Jurafsky and James H. Martin. "Speech and Language Processing", 3rd edition. Pearson Prentice Hall. 2018
4. Martin R., Beel J., Tkaczyk D. - "Citation Data-set for Machine Learning: Citation Styles and Entity Extraction from Citation Strings" - 2018
5. Zhang X., Yann L. - "Text Understanding from Scratch" - 2016
6. Felix A. - "Learning to Forget: Continual Prediction with LSTM" - 2000
7. Graves A. - "A Novel Connectionist System for Unconstrained Handwriting Recognition" - 2009