Classificação de Textos por meio de Redes Neurais e Inferência Bayesiana

Bruno Freire
2º FUND
Instituto Tecnológico de Aeronáutica
São José dos Campos, Brasil
bruno.freire@ga.ita.br

Mateus Thimóteo
2º FUND
Instituto Tecnológico de Aeronáutica
São José dos Campos, Brasil
mateus.thimoteo@ga.ita.br

Lucas Oliveira
2º FUND
Instituto Tecnológico de Aeronáutica
São José dos Campos, Brasil
lucas.oliveira@ga.ita.br

***Resumo*—Esse trabalho discute a aplicação de técnicas de aprendizado supervisionado com redes neurais e do algoritmo de classificação naive bayes para classificar textos de questões de vestibulares de acordo com a matéria.**

***Palavras-chave*—redes neurais, naive bayes, processamento de linguagem natural, classificação de textos**

I. Introdução Teórica

Problemas de classificação são um objeto de estudo muito comum em inteligência artificial, sendo estudados sob o ponto de vista de várias técnicas. Em princípio, trata-se de fornecer a um algoritmo uma entrada, a qual deve ser processada e corretamente identificada como sendo de alguma classe, dentro de um conjunto de classes pré-determinado.

Um exemplo importante é a classificação de textos. Dado um trecho de texto como entrada, como podemos construir um algoritmo que discrimine, a partir dessa amostra de texto, o tipo de conteúdo presente neste? Podemos classificar um texto de diversas maneiras. Podemos querer identificar o idioma em que está escrito, podemos querer analisar uma mensagem de e-mail e identificar se se trata de *spam*, identificar tipos de documentos, ou ainda decidir de que matéria se trata uma questão de vestibular. A essência de como podemos realizar essas tarefas, mais do que a própria escolha de um algoritmo ou técnica, é a definição de *features*, isto é, que aspectos do texto iremos analisar para realizar a classificação.

Em qualquer problema de classificação, devemos obter *features* que sejam quantificáveis, de modo a basear nossos algoritmos. Na classificação de textos, uma *feature* bastante recorrente é a contagem da frequência de ocorrências de uma palavra no texto. Outra possibilidade é analisar grupos articulados de palavras, uma vez que estes são responsáveis por gerar a semântica de um texto. Contudo, é de se esperar que a análise de grupos de palavras, conforme o léxico seja grande, seja uma tarefa exponencialmente mais complicada, de acordo com a quantidade de palavras agrupadas. Portanto, vamos nos concentrar na *feature* frequência de ocorrência de uma palavra.

É possível treinar um agente para classificar textos por meio de técnicas de aprendizado supervisionado, fornecendo um conjunto de textos de treinamento, com suas respectivas classificações esperadas. Com base nas frequências de ocorrência das palavras de um vocabulário nos textos de treinamento, o agente pode inferir a classificação de novos textos. Dentre as várias abordagens possíveis para esse agente, vamos discutir o algoritmo Naive Bayes, além das redes neurais.

O algoritmo Naive Bayes para classificação se baseia em uma estratégia gulosa sobre as probabilidades de uma classificação (saída do algoritmo) dado que é fornecida uma determinada entrada. Isso pode ser formulado mais rigorosamente invocando o conceito de probabilidade condicional. Denotando a saída por y e a entrada por x, representamos a probabilidade de que seja obtido y a partir de x por $P(y|x)$. A resposta do algoritmo será simplesmente

$$\hat{y} = \arg\max_{y} P(y|x) \tag{1}$$

Contudo, para obter essa resposta o algoritmo precisa calcular as probabilidades de todas as saídas condicionadas a uma dada entrada, o que parece bem direto, mas não é uma informação que se obtém diretamente. Por outro lado, é possível obter a partir dos dados de treinamento as probabilidades de uma determinada entrada ter gerado uma dada saída. Basta contabilizar a razão entre quantos exemplos que resultaram na saída y com a entrada x e o total de exemplos classificados como y. Ou seja, é fácil obter $P(x|y)$, mas queremos obter $P(y|x)$. E é aqui que entra o *Teorema de Bayes*, que, apesar de toda sua importância, é um resultado relativamente direto das definições de probabilidade condicional.

Se temos, pela definição de probabilidade condicional, que

$$P(x|y) = \frac{P(x,y)}{P(y)} \text{ e } P(y|x) = \frac{P(y,x)}{P(x)},$$

basta isolar o termo comum $P(x,y)$ e obteremos a equação de equilíbrio

$$P(x|y)P(y) = P(y|x)P(x),$$

que pode ser reescrita de modo mais interessante para nossa aplicação como

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \tag{2}$$

Aplicamos então o Teorema de Bayes ao nosso modelo, de modo que podemos obter a probabilidade de cada saída

condicionada à nossa entrada, a partir de dados que podemos obter a partir do treinamento. No caso, como estamos interessados apenas em determinar o argumento y que maximiza $P(y|x)$, podemos ignorar o denominador em (2), dado que este é constante para uma dada entrada.

Voltando especificamente ao problema da classificação de textos, temos que a ocorrência de cada palavra é um *feature*, de modo que ao tomarmos uma trecho de textos, temos várias palavras ocorrendo simultaneamente. Podemos modelar isso como a probabilidade conjunta de todas essas *features*, por exemplo,

$$P(\text{"O Manga é top"}) = P(\text{"O"}, \text{"Manga"}, \text{"é"}, \text{"top"}).$$

Contudo, a probabilidade de um conjunto de palavras ocorrer num texto é algo muito complexo de se contabilizar, e é aqui que entra a premissa que dá o nome *Naive* ao algoritmo. Vamos assumir, "ingenuamente", que a ocorrência das palavras num texto são eventos independentes, ou seja,

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i) \tag{3}$$

O nome *Naive* do algoritmo não é sem razão. De fato, não parece ser razoável presumir que as ocorrências de palavras sejam independentes umas das outras. Existem diversas expressões chave que aparecem conjuntamente em determinados contextos, como "Manga" e "melhor professor", ao passo que algumas palavras são difíceis de imaginar juntas numa frase, como "Manga" e "imperfeito". Contudo, apesar da ingenuidade da premissa do algoritmo, este em geral obtém resultados muito bons no contexto de processamento de linguagem natural.

Para resumir a ideia do algoritmo Naive Bayes, queremos determinar gulosamente a classe y correspondente à entrada x, e para isso calculamos as probabilidades condicionais, que podem ser revertidas por meio do Teorema de Bayes (2). Quando temos um *feature vector* $x = (x_1, \ldots, x_n)$, por exemplo, a ocorrência de várias palavras num texto, escrevemos

$$P(y|x) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)},$$

onde podemos ignorar o denominador por conta do mesmo ser constante em relação a y, donde extraímos uma releitura de (1),

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y) \tag{4}$$

Vale citar que o algoritmo de *Naive Bayes* possui o seguinte inconveniente: para toda *tag L*, se existe uma palavra *b* que nunca ocorre num texto com a *tag L*, mas ocorre em alguma outra e, portanto, está na lista global de palavras, o algoritmo retorna um *score* nulo para *L*. Na medida que isso ocorre com frequência no conjunto de treinamento, o algoritmo pode vir a retornar diversos zeros como *scores* e ser incapaz de realizar boas predições. Uma solução para tal problema, conhecida por *Laplace Smoothing*, consiste na inserção de questões "artificiais" que contêm **todas** as palavras da lista global exatamente uma vez, sendo uma questão para cada *tag*. Na prática, isso é feito apenas modificando as probabilidades das *features* artificialmente, de modo que é equivalente a realizar uma média entre a distribuição de probabilidades original das palavras e a distribuição uniforme. Com isto, o fenômeno descrito acima não pode mais ocorrer, levando a uma maior "suavização" da avaliação de textos realizada pelo algoritmo. Em outros termos, isto reduz a importância relativa de palavras com pequena frequência na determinação da classe prevista.

No problema abordado neste trabalho, queremos classificar questões do vestibular do ITA quanto à matéria a que se referem, com base nos seus enunciados. Para isso, adotamos duas abordagens: aplicar o algoritmo Naive Bayes e alimentar uma rede neural com um conjunto de questões previamente rotuladas, para realizar o treinamento de ambos os agentes.

II. Implementação

A. Pré-processamento

A implementação dos classificadores de texto se iniciou com um pré-processamento (organização) dos dados de entrada. Para ambos os algoritmos (Naive Bayes e Rede Neural), os dados de treinamento consistem em (i) um arquivo *Questions.txt*, que contém cada uma das questões apresentadas como exemplo, separadas por enter, e (ii) um arquivo *Tags.txt*, com as classificações correspondentes às questões.

Os textos das questões são inicialmente separados por questão e, a seguir, as palavras de cada questão são separadas, utilizando o espaço para identificar tais pontos de separação. Com isso, cria-se uma lista de listas de palavras (uma para cada questão). Ademais, cria-se uma outra lista com todas as palavras que aparecem ao menos uma vez nos exemplos de treinamento.

As palavras passam então por um processo de normalização, que consiste na substituição de caracteres não-ASCII por seus correspondentes (por exemplo, remover acentos e cedilhas), na remoção de caracteres de pontuação e no processo de *stemming*, que busca reduzir uma palavra a seu radical, para tornar iguais palavras com significados semelhantes.

Após a normalização, as palavras são reorganizadas (em ordem alfabética) e é contado o número de ocorrências de cada uma.

Os dados do arquivo Tags.txt são extraídos e cria-se um dicionário (ordenado) que relaciona cada *tag* ao conjunto de questões que são classificados com ela.

Por fim, os dados já obtidos são concatenados em uma tabela (denominada *occurrences table*). Ela é responsável por registrar, para cada questão, a *tag* e as quantidades de ocorrências de cada uma das palavras, dados que serão a base para o algoritmo de Naive Bayes.

B. Treinamento e Predição com Naive Bayes

Uma vez organizados os dados, foi possível implementar os cálculos utilizados no algoritmo de Naive Bayes. A probabilidade de se ter uma determinada *tag* para uma questão é obtida, pelo teorema de Bayes e pela premissa de independência entre as *features* (palavras), em função (i) da probabilidade

de uma certa palavra dada a *tag*, (ii) das probabilidades das tags e (iii) das probabilidades das palavras, de acordo com (2). Tais probabilidades são calculadas por amostragem, ou seja, considerando as ocorrências (palavras) e os *labels* (*tags*) do *input*, e calculando as razões entre quantidades de casos favoráveis e possíveis.

Vale citar que, para o uso do *Laplace Smoothing* para o *Naive Bayes*, foi necessário alterar as funções responsáveis por processar a entrada de *tags* e por criar a *occurrences table*, basicamente inserindo as questões "artificiais"que contêm todas as palavras.

Uma vez obtidos tais dados, o algoritmo já está "treinado"e é capaz de predizer classificações de novas questões. Isto é realizado no método *get label probabilities from occurrences*, que retorna uma "probabilidade" (na verdade, um *score*, pois não há normalização) para cada possível *tag*. Por fim, retorna-se como predição do algoritmo a *tag* que obteve o maior *score*.

C. Treinamento e predição com a Rede Neural

Quanto à estrutura da Rede Neural, utilizando-se o *framework* Keras, foi criada uma rede de 3 camadas densas, as duas primeiras com 20 neurônios cada uma e com função de ativação *ReLU* e a última camada com 5 neurônios, correspondentes às 5 categorias de questões, e com função de ativação *softmax*. Por fim, utilizou-se como *Loss function* da rede, ou seja, a função responsável por quantificar o erro entre o valor esperado e o valor predito pela rede, a função raiz quadrada da variância - *Root Mean Square* (RMS) em inglês -, além do otimizador *Adam* para acelerar o aprendizado.

Para o treinamento da Rede Neural, utilizou-se de um processo semelhante de pré-processamento, tal que as questões contidas em um arquivo *Questions.txt* foram separadas e inseridas em uma lista de *strings*, sendo tais *strings* posteriormente tendo seus caracteres convertidos em seus respectivos valores da tabela ASCII. Para uma normalização do tamanho das *strings* foram adicionados caracteres "0", da tabela ASCII, para que a lista de entradas da rede ficasse com um tamanho uniforme, sendo adotado o valor de 1200 caracteres. Outro fator necessário foi a conversão das *strings* contidas no arquivo *Tags.txt*, que fazem a classificação das questões nas respectivas linhas, para uma lista de *numpy arrays* com 5 elementos binários, tal que para uma dada classe de questão haverá um elemento unitário em uma posição específica e os restante dos elementos serão nulos.

III. Resultados e Discussões

O algoritmo Naive Bayes e a rede neural implementados foram submetidos a um treinamento com um conjunto de teste de 130 questões, sendo 30 de Química, 30 de Matemática, 30 de Física, 20 de Inglês e 20 de Português, todas contidas no arquivo Questions.txt, e marcadas através do arquivo Tags.txt. Após o treinamento ambos os agentes foram testados num pequeno conjunto de 16 questões, contendo 10 questões de Matemática e 6 de Física.

Os resultados obtidos pelo Naive Bayes foram relativamente satisfatórios. O algoritmo classificou corretamente todas as questões de física, acertando 4 de 10 das questões de matemática, o que leva a um desempenho de 62.5%.

Os resultados obtidos pela rede neural foram em sua maioria incorretos. A rede classificou metade das questões de Matemática como Química, e a outra metade como Inglês. Das questões de Física houve 4 corretamente classificadas, e duas que foram classificadas como Inglês. É razoável presumir que o conjunto de treinamento (composto de apenas 130 questões) não foi suficiente para treinar a rede adequadamente, ao menos não com a arquitetura implementada.

Um novo teste foi realizado com o algoritmo de Naive Bayes, tomando 6 questões de cada matéria do vestibular do ITA de 2019 para serem classificadas. Os resultados obtidos foram: física: 6/6; química: 5/6; matemática: 3/6; inglês: 0/6; e português: 4/6. Neste caso, obteve-se um desempenho de 60.0%.

Vale citar que os resultados nitidamente inferiores em inglês podem estar relacionados (i) à semelhança textual entre enunciados de português e de inglês (dado que a prova de inglês do ITA é escrita majoritariamente em português) e (ii) ao uso do *stemmer* (importado do *nltk*) especificamente em língua portuguesa, que dificulta a análise de textos em inglês, ocasionando resultados inferiores.

Ademais, com a ampliação da quantidade de questões de treinamento, o algoritmo *Naive Bayes* mostra-se promissor para a obtenção de melhores desempenhos de classificação de questões.

IV. Conclusão

Devido ao tamanho limitado do conjunto de treinamento, pode ser que o aprendizado dos agentes não tenha sido suficientemente amplo para obter um bom desempenho na classificação do conjunto de testes, o qual, por sua vez, também pode ter sido ineficaz para testar o aprendizado devido ao seu tamanho.

Outro fator que pode ter prejudicado os resultados obtidos foi a própria estrutura organizada para a rede, de forma que não foi tão eficiente quanto o esperado. Assim, uma possível alteração seria utilizar uma estrutura de rede convolucional de uma dimensão, com um volume menor de caracteres para *input*, ainda fazendo uma normalização caso necessário. Imagina-se que uma rede convolucional facilite na formação de padrões para as diferentes classes de questões em camadas intermediárias, auxiliando na classificação final.

Enquanto o Naive Bayes possui uma *feature engineering* bem definida, baseada nas ocorrências das palavras, a rede neural possui uma arquitetura que camufla o significado das *features* em suas camadas, de modo que o projeto de uma arquitetura eficiente para realizar a tarefa de classificação é uma tarefa complicada.

Há de se considerar também a questão da formatação dos dados de entrada. Questões de ciências exatas são carregadas de notações especiais que não podem ser representadas perfeitamente com caracteres ASCII, o que pode ter dificultado o aprendizado dos agentes a partir do texto. Além disso, a rede neural possuía problemas em receber como entrada caracteres

especiais (letras acentuadas, por exemplo), os quais foram substituídos por caracteres coringa. Isso pode ter prejudicado a identificação de palavras e por conseguinte a classificação dos textos.

Por fim, podemos concluir a partir dos resultados que o Naive Bayes, apesar de ser de natureza muito mais simples que uma rede neural, se mostra bastante adequado para a tarefa de classificação de textos.

AGRADECIMENTOS

Agradecemos ao Prof. Dr. Máximo, pela consideração em estender o prazo de entrega deste trabalho.

REFERÊNCIAS

[1] S. Russel and P. Norvig, Artificial Intelligence - A Modern Approach, 3rd ed., Prentice Hall, 2010, pp. 865–867, 480–538.

[2] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016, pp.70–71.