Relatório - Inteligência Artificial

João Guilherme da Costa Seike nUSP: 9784634

Rafael Nazima nUSP 11208311

Daniel Adan Pereira nUSP 11295866

São Paulo
2022

Introdução

O objetivo deste trabalho é colocar em prática os conceitos aprendidos em aula sobre Naive Bayes, implementando o algoritmo para realizar uma análise de sentimento de tweets, classificando-os entre “sentimento positivo” ou “sentimento negativo”.

Construção do vocabulário

Nosso primeiro passo para começar a implementação do Naive Bayes foi a criação de um vocabulário para os dados que serão analisados, que nesse caso se tratam de 1.578.627 *tweets* rotulados, sendo o rótulo 1 para sentimento positivo e o rótulo 0 para sentimento negativo. Para que o programa tenha todos os dados suficientes para funcionar precisamos do SentimentAnalisys Dataset.csv e stopwords_en.txt.
Os arquivos do Holdout criados são treinamento.txt (⅔ do arquivo SentimentAnalysis Dataset.csv) teste.txt(o ⅓ restante), para o Cross Validation foram gerados arquivos foldn.csv (n de 1 a 10) que servem como arquivo para teste, e os Treinofoldn.csv (n de 1 a 10).
Para a criação de um vocabulário,inicialmente utilizamos uma lista ligada para armazenarmos **todas** as palavras de todos os tweets (inclusive palavras repetidas),mas .Mas para a estrutura que iremos utilizar para limpar os dados, optamos por uma estrutura de mais eficiente, o HashSet, que usa uma HashTable e seus elementos não são ordenados, fazendo com que as operações nessa estrutura tenham a complexidade O(1), ou seja, independentemente da quantidade de dados adicionados, removidos, retirados, o tempo de execução será o mesmo, ideal para nosso cenário que temos que trabalhar com milhões de dados inseridos no set.Feito isso nosso próximo passo foi realizar um trabalho de limpeza nesses dados, para separar as palavras utilizamos como critério os caracteres [\t , ' . ; ? ! \ ") (# * : (+ / -] + espaço, removemos os termos neutros (links, e usuários do twitter).
Nossa classe responsável pelos métodos principais que executa o Naive Bayes tem também o método para achar a distribuição de frases de sentimento positivo e negativo, e a guardamos numa lista de objetos com atributos de número de repetições positivas, repetições negativas da palavra, a probabilidade que contribui para frase ser positiva ou negativa, e a palavra em string, essa lista de palavras é guardada com serializable em um arquivo **.ser**

Implementação do algoritmo Naive Bayes

A aplicação da fórmula de bayes pode ser verificada na imagem abaixo, onde consta trecho de código com a execução em java do método:

Seguindo para o bayes:
Lê o arquivo escolhido.
Faz os cálculos de Naive Bayes:

```
double tempN = 1;
double tempP = 1;
        for(int i = 0;i < palavrasIndv.length;i++) {
                Palavras Paltemp = ClassPalavras.get(palavrasIndv[i]);
                if(Paltemp != null) {
                        if(Paltemp.repeticoesN != 0) {
                                tempN *= ((double)Paltemp.repeticoesN/(double)PalNeg);
                        }
                        if(Paltemp.repeticoesP != 0) {
                                tempP *= ((double)Paltemp.repeticoesP/(double)Palpos);
                        }
                }
        }

classificadorN = tempN * distNeg;
classificadorP = tempP * distPos;
```

Após realizados os cálculos a partir da fórmula de bayes, utilizamos um classificador para verificar qual é o resultado encontrado pela rede.

```
classificadorN = tempN * distNeg;
classificadorP = tempP * distPos;
        if(classificadorP > classificadorN && tempN != 1) {
                resultadoFinal = true;
        }
        else if(tempN == 1) {
                resultadoFinal = false;
        }
        if (classificadorP < classificadorN && tempP != 1) {
                resultadoFinal = false;
        }
        else if(tempP != 1) {
                resultadoFinal = true;
        }

        if(resultadoFinal == sentimento) {
        acertos++;
        }
        else {
                erros++;
        }
        temp = "";
        }
```

Este trecho é extremamente importante, uma vez que é nele onde é realizado o cálculo que auxilia na classificação de cada tweet.

Experimentos

Foi embaralhado o arquivo SentimentAnalysisData.csv para cada experimento o método Bayes() da classe Esqueleto multiplicou por palavra da frase o número de repetições para cada sentimento dividido pelo número de palavras de cada sentimento, obtendo dois valores para cada palavra, um a probabilidade da palavra ser usada para sentimentos positivos e outro

Resultados

Desempenho do classificador:

Holdout

		Avaliado	
		Sentimento Positivo	Sentimento Negativo
Tweet	Sentimento Positivo	154520	109000
	Sentimento Negativo	36507	226156

Foi embaralhado o arquivo SentimentAnalysisData.csv para cada experimento resultando em uma acurácia de por volta de 72,5% para os Holdouts. Observando a matriz de confusão do Holdout percebemos que os erros do Naive Bayes foram substancialmente maiores para resultar em um Falso negativo

Cross Validation:

Inicialmente para podermos utilizar o método k-fold cross validation, realizamos a divisão do dataset de tweets em 10 folds. O trecho de código abaixo realiza a leitura de cada linha do dataset em .csv gerando outros 10 dez arquivos .csv, sendo que cada um destes novo 10 arquivos contém um número de tweets igual a n/10, onde n = quantidade total de tweets do dataset.

```
int controladora = 1;
int texto = 2;
try {
                FileWriter writer = new FileWriter("fold1.csv");
                int TamanhoLista = Linhas.size();
                for(int i = 0; i < TamanhoLista/10; i++) {
                        writer.append(Linhas.get(i));
                }
        writer.flush();
                writer.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
for(int i = controladora; i < 10;controladora++)
{
    try {
        FileWriter writer = new FileWriter("fold"+texto+".csv");
        for(int x = (Linhas.size()/10)*i; x < (Linhas.size()/10)*(i+1); x++) {
                if(x >=Linhas.size()){
                    break;
                };
                writer.append(Linhas.get(x));
        }
            writer.flush();
            writer.close();
            i++;
            texto++;

                } catch (IOException e) {
                        // TODO Auto-generated catch block
```

O conteúdo localizado em cada um dos folds foram randomizados em passo anterior do algoritmo, e a rede foi executada dez vezes, uma para cada fold criado.

O Cross Validation apresentou estabilidade na acurácia durante os folds, o mesmo SentimentAnalysisData.csv embaralhado para holdout e cross validation teve 72,54% como acurácia no Holdout e cada fold em ordem crescente teve como acurácia

Fold	Erro	Acertos	Acurácia
1	43343	114512	0.72542523
2	42856	114999	0.72851034
3	43343	114512	0.72542523
4	43021	114834	0.72746508
5	43231	114624	0.72613474
6	43177	114678	0.72647683
7	43390	114465	0.72512749
8	42999	114856	0.72760445
9	42912	114943	0.72815559
10	43043	114812	0.72732571

Erro Padrao 10-fold : 0.0003546877024790768
Erro Verdadeiro vai ficar entre : 0.27256191514073885 e 0.2739522909344569

Desempenho do algoritmo (dados originais vs sem stop words)

O stop words não necessariamente ajuda com o desempenho do algoritmo, havendo quase nenhum ou decrescimento na acurácia do modelo

Conclusão

O Naive Bayes teve uma taxa de erros consistente para todos os testes, seu algoritmo surpreende com a velocidade que é capaz de processar a quantidade de dados processados. A simplicidade do algoritmo permite ao programador encontrar os erros do programa com uma maior precisão.