Recuperação inteligente da informação
Relatório atividade E-1.10

André Pacheco

Doutorado em Ciência da Computação
Programa de pós-graduação em informática
Universidade Federal do Espírito Santo

Conteúdo

1 Introdução

A atividade E-1.10 tem como objetivo a realização de um detector de plágio utilizando um buscador web para a busca de possíveis documentos similares. Sendo assim, foi construída uma aboradagem que utiliza uma API da Google para verificar se um dado documento é ou não suspeito de plágio.

2 Projeto de um detector de plágio

O primeiro passo da abordagem proposta para detectar possíveis plágio em documentos é determinar qual o buscador web utilizar e o que buscar. A abordagem usa a uma API de busca na Google [1] utilizando Python como linguagem de programação. O documento a ser avaliado está no formato .txt e foi considerado cada parágrafo como uma linha dentro do txt. Com isso, a solução proposta busca plágios a cada parágrafo do texto.

Para facilitar a busca, primeiro foi retirado todas as *stopwords* do parágrafo e foi considerado as primeiras 15 palavras como chave de busca. Essas palavras representarão todo o parágrafo. Esse valor pode ser considerado um hiperparâmetro que pode ser alterado em busca de uma solução melhor. Tendo em mãos as palavras chave da busca, as mesmas são submetidas a API da Google que retorna uma lista com n URLs (no código atual está setada como 5). Dessas URLs

[1] https://pypi.python.org/pypi/google

são retornadas os conteúdos da página HTLM. Vale ressaltar, como o objetivo da atividade é uma prova de conceito, não são considerados documentos em PDF, DOC, DOCX, etc. Todavia, pode ser implementado no futuro utilizando o mesmo código.

Os conteúdos retornados das páginas em HTLM são processados para retirar carcateres especiais da linguagem e o texto é utilizado para verificar a possibilidade do plágio. A estratégia para verificar o plágio é separar o texto recuparado em conjuntos de k caracteres, que será chamado de *batch*. O valor de k escolhido é 100, porém também é um hiperparâmetro que pode ser alterado. Cada *batch* é comparado com o parágrafo em questão, também utilizando k caracteres. Essa comparação de similaridade é realizada via similaridade por cosseno. Os *batches* de texto com similaridade acima de um *threshold* pré-definido são escritos em um arquivo chamado Resultado.txt que informa a URL na qual o possível plágio ocorreu e o texto do parágrado. O valor desse *threshold* está definido como 0.3, mas também é um parâmetro sensitivo.

3 Exemplo de funcionamento

Um exemplo de funcionamento é apresentado no arquivo *query.txt*. Nele o seguinte texto é descrito:

```
Um SOM possui diferentes arquiteturas, mas a mais utilizada é a bidimensional
como ilustrado na Figura 1. Como pode ser observado, a rede é como um grid 2D,
na qual cada entrada estará conectada a todos os neurônios do mapa, que neste
caso possui 20 neurônios em um grid 4 x 5. Além disso,
observe que não existe conexões entre os neurônios da rede. Cada neurônio
possui um conjunto de peso de dimensão igual ao número de features do
conjunto de dados. Por exemplo, se nosso conjunto fora a base de dados
da Iris, ou seja, possui 4 features (largura e comprimento de pétalas
e sépalas), cada neurônio vai possuir 4 pesos, cada um deles conectados
a uma feature diferente

O treinamento de um SOM é relativamente simples. A ideia principal do
algoritmo é criar grupos de neurônios especialistas em certas entradas.
Essa ideia é inspirada no cérebro humano, na qual temos regiões do mesmo
responsáveis pela visão, audição, etc. Podemos dividir o treinamento em duas
etapas: a competitiva e a cooperativa. Começamos sempre pela etapa competitiva.

As CNNs são similares a redes neurais tradicionais: ambas são compostas por
neurônios que possuem pesos e bias que necessitam ser treinados. Cada neurônio
recebe algumas entradas, aplica o produto escalar das entradas e pesos além
de uma função não-linear. Além disso, ambas possuem a última camada toda
conectada e todos os artifícios utilizados para melhorar a rede neural
tradicional também são aplicados nesta camada. Dessa forma, qual a
vantagem de se utilizar uma CNN? Uma CNN assume que todas as entradas são
```

```
imagens, o que permite codificar algumas propriedade na arquitetura.
Redes neurais tradicionais não são escaláveis para imagens, uma vez que a
mesma produz um número muito alto de pesos a serem treinados [6].
```

Cada parágrafo foi avaliado com a abordagem descrita anteriormente e os resultado obtido é:

```
## SUSPEITA DE PLÁGIO ENCONTRADA ##
URL: http://www.computacaointeligente.com.br/algoritmos/mapas-auto-organizaveis-som/

## PARTE DO TEXTO SUSPEITA ##
som possui diferentes arquiteturas utilizada é bidimensional ilustrado figura 1
pode observado rede é grid 2d cada entrada estará conectada todos neurônios
mapa neste caso possui 20 neurônios grid 4 x 5 além disso observe não existe
conexões neurônios rede cada neurônio possui conjunto peso dimensão igual
número features conjunto dados exemplo conjunto fora base dados iris possui 4
features (largura comprimento pétalas sépalas) cada neurônio vai possuir 4
pesos cada conectados feature diferente

## SUSPEITA DE PLÁGIO ENCONTRADA ##
URL: http://www.computacaointeligente.com.br/artigos/redes-neurais-convolutivas-cnn/

## PARTE DO TEXTO SUSPEITA ##
cnns são similares redes neurais tradicionais ambas são compostas neurônios
possuem pesos bias necessitam treinados cada neurônio recebe algumas entradas
aplica produto escalar entradas pesos além função não-linear além disso ambas
possuem última camada toda conectada todos artifícios utilizados melhorar rede
neural tradicional também são aplicados nesta camada dessa forma vantagem utilizar
cnn cnn assume todas entradas são imagens permite codificar algumas propriedade
arquitetura redes neurais tradicionais não são escaláveis imagens vez mesma
produz número alto pesos serem treinados [6]
```

As URLs apresentadas apontam para os locais nas quais os textos foram retirados.