UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

WILLER CARNEIRO QUINTANILHA

SIMPLENLP, UMA BIBLIOTECA PARA PROCESSAMENTO DE LINGUAGEM NATURAL, APLICADA A ANÁLISE DE CARACTERÍSTICAS PSICOLÓGICAS

BAURU
Novembro/2019

WILLER CARNEIRO QUINTANILHA

SIMPLENLP, UMA BIBLIOTECA PARA PROCESSAMENTO DE LINGUAGEM NATURAL, APLICADA A ANÁLISE DE CARACTERÍSTICAS PSICOLÓGICAS

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Profª. Dra. Simone das Graças Domingues Prado

BAURU
Novembro/2019

Willer Carneiro Quintanilha SimpleNLP, uma biblioteca para processamento de linguagem natural, aplicada a análise de características psicológicas/ Willer Carneiro Quintanilha. – Bauru, Novembro/2019- 36 p. : il. (algumas color.) ; 30 cm.
Orientador: Profª. Dra. Simone das Graças Domingues Prado
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Ciência da Computação, Novembro/2019.
1. Tags 2. Para 3. A 4. Ficha 5. Catalográfica

Willer Carneiro Quintanilha

SimpleNLP, uma biblioteca para processamento de linguagem natural, aplicada a análise de características psicológicas

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Profª. Dra. Simone das Graças Domingues Prado
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Professor Convidado 1
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Professor Convidado 2
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, _____ de _____ de _____.

Dedico este este trabalho a minha mãe que me apoiou durante todo este percurso sempre me apoiando e acreditando no meu futuro.

Agradecimentos

Agradeço a minha família, ao pessoal da república Alcatraz e ao professores do curso de Ciência da Computação da UNESP campus Bauru.

"If you hit the wall, you should thank God for placing it in front of you — that's when something new is born."

Akira Yoshino

Resumo

O Processamento de Linguagem Natural busca descrever técnicas para a obtenção de informações em conjuntos de dados linguísticos. Com a explosão da internet e a difusão dos meios de telecomunicações, temos uma quantidade absurda de dados transitando pela rede, grande parte destes dados se trata de dados textuais, tweets, posts no facebook, comentários em redes sociais e et cetera. Este trabalho tem o objetivo de delinear o relacionamento entre linguagem e psicologia para fins computacionais e verificar este relacionamento através do desenvolvimento de uma biblioteca para Processamento de Linguagem Natural e experimentação desta aplicada a análise de aspectos psicológicos.

Palavras-chave: Processamento de Linguagem Natural, Latent Semantic Indexing, Linguá, Semântica, Psicologia.

Lista de figuras

Lista de abreviaturas e siglas

PLN	Processamento de Linguagem Natural
LSI	Latent Semantinc Indexing

Sumário

1 Introdução

O objetivo da ciência linguística é caracterizar e explicar os fenômenos linguísticos que nos envolvem, em conversações, escritas e outros formatos de mídia. Parte deste processo tem haver com o lado cognitivo de como os humanos adquirem, produzem e entendem a linguagem, temos também como se da o entendimento do relacionamento entre as expressões linguísticas e o mundo ("part of it has to do with understanding the relationship between linguistic utterances and the world") (MANNING; MANNING; SCHÜTZE, 1999) e o entendimento das estruturas linguísticas pelas quais nos comunicamos.

No que tange a análise semântica da linguagem temos um grande desafio: "All grammars leak"(Sapir 1921: 38), dada essa afirmação temos que apenas a gramática não basta para obtenção de sentido "This is because people are always stretching and bending the 'rules' to meet their communicative needs" (MANNING; MANNING; SCHÜTZE, 1999).

Então para analisar o aspecto de sentido da linguagem existem as teorias semânticas filosóficas da linguagem que descrevem num sistema linguístico o componente do sentido das palavras, da interpretação das sentenças e dos enunciados.

A linguagem esta intrinsecamente ligada a psicologia, segundo Lacan "a linguagem é a condição do inconsciente", é a tese de Lacan (1970, p. 39), para ele "o inconsciente é, em seu fundo, estruturado, tramado, encadeado, tecido de linguagem"(LACAN, 1981, p. 135), em suma "a linguagem estrutura o inconsciente"e portanto serve como meio de acesso a ele.

Estudos recentes demonstram a importância da psicologia na atualidades, temos altas taxas de prevalência em desordens mentais em todo mundo, segundo artigo publicado no WHO International Consortium in Psychiatric Epidemiology podemos observar que "Focusing first on the lifetime prevalences of any disorder, the highest estimates suggest that more than one-third of the sample experienced at least one disorder at some time in their life in Brazil (36.3

Dada importância do tema das desordens mentais o relacionamento da linguagem com o subconsciente e o componente semântico da linguagem este trabalho busca desenvolver uma biblioteca para processamento de linguagem natural e aplica-la a análise de aspectos psicológicos.

2 Motivação

A principais motivações para o desenvolvimento deste trabalho são a falta de bibliotecas que implementem técnicas de processamento de linguagem natural de forma simples e intuitiva principalmente para a linguá portuguesa em 2019 e a experimentação da aplicação de técnicas de processamento de linguagem natural para extração de características psicológicas, dada a importância e o recente crescente do número de pessoas acometidas por doenças psiquiátricas.

3 Objetivos

Neste capitulo serão descritos os objetivos gerais e específicos deste trabalho.

3.1 Objetivo Geral

Implementação de uma biblioteca para Processamento de Linguagem Natural nas linguagens Scala e JAVA e aplicação desta biblioteca a análise de aspectos psicológicos.

3.2 Objetivos Específicos

a) Implementação de técnicas relevantes de Processamento de Linguagem Natural em JAVA e Scala.

b) Compilação da implementação em uma biblioteca chamada SimpleNLP.

c) Obtenção de base de dados relacionadas a aspectos da psicologia.

d) Experimentação dos métodos desenvolvidos na biblioteca SimpleNLP nas bases de dados coletadas

4 Metodologia

Inicialmente realizarei uma revisão bibliográfica sobre processamento de linguagem natural, após esta etapa inciarei o desenvolvimento da biblioteca SimleNLP em JAVA e Scala, assim que terminada a implementação da biblioteca realizarei uma pesquisa sobre bases de dados relacionadas a aspectos psicológicos e experimentarei os métodos implementados, nas bases de dados obtidas.

5 Linguagem

A linguagem é tida como a qualidade inata ao ser humano de se comunicar através de uma linguá, "Convencionou-se atribuir o termo linguagem à capacidade geral que temos, enquanto seres humanos, de utilizar sinais com vistas à comunicação." (LEITE, 2010) todo ser humano não obstante seus atributos físicos ou de ser acometido por uma patologia é capaz de se comunicar através de um meio concreto ao qual chamamos linguá, esta por sua vez evolui junto a cultura para satisfazer as necessidades comunicativas da sociedade.

5.1 Teorias filosóficas semânticas da linguagem

A obtenção de sentido de um enunciado é uma das tarefas mais complexas do Processamento de Linguagem Natural dada a enorme variação de sentidos que uma sentença pode ter.

As teorias semânticas filosóficas da linguagem buscam generalizar o processo de aquisição de sentido de forma que seja possível estabelecer princípios para a interpretação de qualquer enunciado.

"Os filósofos dedicam muito tempo a esta questão: em que consiste o significado de uma expressão linguística? Esta é uma indagação filosófica. E, como acontece comumente em Filosofia, não se tem única resposta, mas várias. O problema com o qual as pesquisas, em IA e PLN, se deparam é o seguinte: qual das respostas que a Filosofia nos empresta torna possível, aos sistemas de computador, endereçar as necessidades da linguagem? E, se existe mais de uma, qual é a mais adequada?" (PINHEIRO, 2010)

Os próximos capítulos farão uma breve introdução as principais teorias semânticas filosóficas da linguagem

5.1.1 Teorias Ideacionais

As teorias semânticas ideacionais traziam a visão de que uma sentença corresponde a um estado mental, uma ideia ou uma imagem, portanto dizer que duas expressões tem o mesmo sentido seria dizer que duas expressões transmitem o mesmo pesamento e dizer que são ambíguas seria dizer que uma expressão pode transmitir pensamentos diferentes. John Lock é o filósofo mais notório desta linha de teorias que considera que "os significados de expressões linguísticas são as ideias da mente" (PINHEIRO, 2010), temos então que durante o processo comunicativo quem diz uma expressão tem uma ideia do que esta sendo dito e quem recebe esta expressão forma uma ideia do que foi dito, logo poderíamos dizer que o processo

comunicativo ocorreria de maneira satisfatória, quando os interlocutores formula-sem a mesma ideia a cerca das expressões.

Contudo essas teorias foram entrando em desuso no final do século XIX, principalmente em virtude de sua subjetividade e individualização do significado, o legado destes filósofos perdura até hoje e se expressa no consenso de que "a linguagem é de fato a função mental simbólica suprema, e é virtualmente impossível conceber o pensamento na sua ausência."(TATTERSALL, 2006, p.73).

5.1.2 Teorias Representacionalistas

Das teorias a respeito dos significados na linguagem as mais facilmente aceitas pelo senso comum são as representacionalistas, estas dizem que toda palavra e expressão linguística independente de sua complexidade tem significado por que representa coisas da realidade, logo significa o que representa.

O quadro que se formara no século XX era o de que temos a realidade e o meio que usamos para representa - la, este meio consiste em utilizar signos, que são as palavras para denotar, nomear e designar as coisas do mundo, as expressões servem para descrever a forma como estas coisas estão posicionadas no mundo, como estas coisas se relacionam, e qual o estado destas coisas.

As palavras são mais significativas a medidas em que descrevem melhor coisas da realidade sendo então que as palavras que não possuem valor representativo assumem funções auxiliares servindo apenas ao proposito da construção de representações mais complexas.

Esta teorias abandonam a subjetividade do significados, e tem fundamental simplicidade e capacidade de generalização.

5.1.3 Teorias Pragmáticas

As teorias pragmáticas surgem como alternativa as teorias representacionalistas em meados do século XX tendo como principal autor Wittgenstein. Para os teóricos desta linha de pensamento uma teoria de significado deveria descreve - lo em termos de uso da linguagem, dado que diferentes contextos sociais levam a usos distintos.

Temos então uma teoria funcionalista de linguagem, que diz respeito a importância do uso ou contexto sobre a significação literal do mesmo.

Sendo assim para estes teóricos o significado de uma expressão não é determinado apenas pela expressão em si, mas também pelo contexto onde se encontra, ou seja a função da expressão dado o contexto.

Figura 1 – A importância do uso.

Fonte: Elaborada pelo autor.

5.1.4 Teorias Inferencialistas

Estas teorias surgiram ao final do seculo XX, segundo os teóricos desta linha de pensamento as inferências prevalecem sobre a representação na ordem do processo de descrição do sentido em conceitos e sentenças.

"Os termos de uma língua representam objetos do mundo real não pela propriedade primitiva de serem, respectivamente, representações e representados, mas em razão das participações dos conceitos (expressos pelos termos) em raciocínios, como premissas ou conclusões de inferências." (PINHEIRO, 2010)

Para estes teóricos entender o mundo significa compreender os relacionamentos expressos entre conceitos em raciocínios. Sellars mostra que o relacionamento inferencial denota a diferença entre relatos dados por humanos e maquinas. Enquanto humanos conseguem justificar as razões pelas quais deram determinados relatos as máquinas não conseguem. Pois as razões são expressas na forma de conhecimento inferencial, uma maquina pode dizer: "O céu é azul"mas desconhece as potenciais inferências derivadas do uso destes conceitos, como "azul"é uma cor, "azul"não é vermelho. Assim podemos distinguir o nível de entendimento dos conceitos entre humanos e uma máquina.

5.2 Linguagem e psicologia

A psicologia esta intrinsecamente ligada a linguagem, segundo Lacam "linguagem é a condição do inconsciente", é a tese de lacan (1970, p39), para ele "o inconsciente é, em seu fundo estruturado, tramado, encadeado, tecido de linguagem"(LACAN, 1981, 9. 135) em suma "a linguagem estrutura o inconsciente"e portando serve como meio de acesso a ele.

Então com o uso das teorias semânticas filosóficas da linguagem aplicadas ao processamento de linguagem natural podemos em tese entender o sentido sintomático expresso pelo

discurso dos indivíduos acometidos por patologias psíquicas.

6 Processamento de Linguagem Natural

O Processamento de Linguagem Natural tem como objetivo obter informações de um conjunto de dados linguísticos, sejam estes representados em texto, áudio ou qualquer outro meio. Um dos grandes, se não, o maior desafio desta areá cabe a subjetividade da linguagem, dada que está sempre esta sujeita a interpretação do individuo, não podendo ser avaliada portanto como algo independente das experiências do interlocutores.

Contudo grandes massas de dados nos proporcionam a possibilidade de extrapolação da subjetividade da interpretação linguística única de cada individuo para a informação contida nas interpretação comuns a muitos indivíduos a exemplo do substantivo: "bolsa"os significado deste substantivo se analisado pela subjetividade da cada individuo terá significados distintos, contudo se avaliado dentro do grupo de WhatsApp das pessoas que discutem a respeito do mercado de ações, significara na maioria das vezes o local onde se transacionam valores mobiliários ou seja a bolsa de valores.

Para a realização deste tipo de analise o PLN fornece ferramental para representação e análise de dados linguísticos nos mais variados formatos.

Nas próximas subseções trataremos de algumas das técnicas de PLN utilizadas para o processamento de texto.

6.1 Tokenização

O processo de tokenização tange ao aspecto léxico da linguagem e busca a representação de texto na forma de tokens, geralmente os tokens são tidos como a menor unidade de sentido em um texto, habitualmente são desconsiderados para o processo de tokenização pontuações e espaços, em suma o processo de tokenização é a representação de documentos na forma de um conjunto de palavras: Lista de tokens.

6.2 Representações

Serão apresentadas a seguir formas de representação de dados textuais apropriadas a aplicação de técnicas algébricas, geométricas e de heurísticas e amplamente usadas no âmbito do Processamento de Linguagem Natural.

6.2.1 Binary Encoding

A codificação binária é um modelo da representação de dados textuais em formato numérico, para este tipo de representação são dados valores numéricos para cada palavra

presente nos textos a serem codificados, cada palavra tera apenas um valor numérico que será o mesmo para todos os documentos, logo se a palavra "vetores"é representada por 5 no documento A_1 será representada por 5 também no documento A_2, neste formato cada documento será representado como um vetor onde o índice do vetor representa as palavras contidas nos documentos e os valores contidos nesse vetor a presença ou não destes termos contando ou não a quantidade de ocorrências dos termos, para exemplificar em um modelo que desconsidera o numero de ocorrências de cada termo nos documentos os vetores que representariam A_1 e A_2 teriam $V[5] = 1$,

6.2.2 Word Embendding

Os modelos de Word Embendding buscam um tipo de representação onde a posição no espaço de cada documento diz respeito de alguma forma a informação que se busca obter, há vários tipos de representação de documentos Word Embedding para diferentes tipos de processamento, com este tipo de representação podemos representar documentos em espaços 2D e 3D e visualizar graficamente a posição destes no espaço, geralmente esta classe de representações utiliza processos de redução dimensional para melhor representações de dados.

6.3 Algorítimos

Este capitulo descreverá a fundamentação teórica dos principais algorítimos utilizados para o desenvolvimento deste trabalho.

6.3.1 Similaridade de cosseno

Para o Processamento de Linguagem Natural a medida de similaridade de cossenos é utilizada para determinar a similaridade entre dois documentos, dados que estes estejam representamos em um espaço vetorial.

6.3.2 Clustering

Algorítimos de clustering particionam partes de um conjunto de dados em grupos este grupos são comumente conhecidos como clusters, o objetivo deste tipo de algorítimo é agrupar objetos similares no mesmo grupo e objetos distintos em grupos diferentes, existem duas principais técnicas de clusterização, a clusterização hierárquica e a não hierárquica, aqui iremos tratar somente da clusterização não hierárquica.

Algoritmos de clusterização não hierárquica geralmente começam com partições aleatoriamente selecionadas, e a cada iteração estas partições são melhoradas, geralmente esta classe de algorítimos emprega o uso de varias iterações onde a cada iteração são calculadas melhores partições.

Se este tipo de algorítimo pode ter varias iterações qual seria o melhor critério de parada ? Provavelmente o critério de parada mais importante para este tipo de algorítimo seja o da vizinhança, dado que o objetivo desta classe de algorítimos é agrupar dados semelhantes, contudo seja qual for o meio de mesuração para a parada, podemos simplesmente continuar executando iterações enquanto a qualidade dos clusters (agrupamentos) estiver aumentando e podemos parar quando a curva de melhoria da qualidade dos clusters estabilizar ou quando a qualidade começar a decrescer.

Outra grande questão para essa classe de algorítimos é como determinar o número ideal de clusters, em algumas situações teremos conhecimento prévio a respeito do número correto de clusters, mas se a situação não for está o processo será experimenta..

O algorítimo de clusterização implementado neste trabalho foi do tipo K-means baseado na implementação do algoritimo de Lloyds, este algoritmo executa os seguintes passos:

a) O primeiro passo é atribuir para cada cluster um ponto que será o seu centro.

b) O segundo passo é atribuir cada dado ao cluster mais próximo, ou seja com menor distância centro do cluster para o dado

c) O terceiro passo é re computar cada ponto central de cada clusters com o valor da média dos dados atribuídos ao mesmo.

6.3.3 Algorítimo genético

Algorítimos genéticos são algorítimos de busca e otimização baseados nos princípios da evolução e genética, algoritmos genéticos podem prover soluções quase ótimas para problemas de busca e otimização.

Utilizarei um neste trabalho um algorítimo genético para otimização de uma função linear a ideia é desenvolver uma função linear que dada um conjunto de entras codificadas de forma binária, devolva a que classe ela pertence.

Cada solução será representada como um cromossomo, um conjunto de cromossomos em um determinado instante de tempo será uma geração, os cromossomos mais adaptados ou seja as melhores soluções, serão perpetuados para as próximas gerações, enquanto as piores soluções serão descartas.

6.3.4 Latent Semantic Indexing

Este algorítimo tem a finalidade de obter informação através da co-ocorrência de termos em documentos, é fato que dois ou mais termos aparecem nos mesmos documentos com mais frequência do que o acaso, logo podemos evidenciar que a aparição destes termos pode denotar alguma informação.

LSI é uma técnica que projeta consultas(queries) e documentos em um espaço com dimensões semânticas latentes, termos que co - ocorrem são projetados na mesmas dimensões e termos que não co-ocorrem são projetados em dimensões diferentes.

Uma característica interessante desta técnica é que em um espaço semântico latente uma query e um documento pode ter alta similaridade de cosseno até se não compartilharem nenhum termo, podemos evidenciar neste ponto uma analise não somente das ocorrências dos termos, mas também do seu contexto, esse tipo de analise pode ser entendida como tendo algo em comum com as teorias semânticas pragmáticas, pois não analisa somente a representação, mas também o contexto.

O espaço semântico latente é um método de redução dimensional. Técnicas de redução de dimensionalidade são utilizadas também para tradução de objetos que existem num espaço altamente dimensional para um espaço com menos dimensões, geralmente em espaços com duas ou três dimensões para propósitos de visualização.

Figura 2 – Matriz de termos em documentos

$A =$

	d_1	d_2	d_3	d_4	d_5	d_6
cosmonaut	1	0	1	0	0	0
astronaut	0	1	0	0	0	0
moon	1	1	0	0	0	0
car	1	0	0	1	1	0
truck	0	0	0	1	0	1

Fonte: Foundations of statistical natural language processing.

Neste trabalho utilizamos o método de Decomposição em Valores Singulares para análise de co - ocorrência. O método de SVD projeta um espaço de n dimensões em um espaço de k dimensões onde $n > k$. Temos geralmente que n é o número de tipos de palavras na nossa coleção e k é frequentemente escolhido entre 100 e 150.

Veja nas figuras de Figura 4 a Figura 9 exemplo retirado de Grossman and Frieder's Information Retrieval, Algorithms and Heuristics do algoritmo LSI utilizando SVD.

Para uma coleção dos seguintes documentos:

a) $d1$: Shipment of gold damaged in a fire

b) $d2$: Delivery of silver arrived in a silver truck

c) $d3$: Shipment of gold arrived in a truck.

Figura 3 – Representação no espaço

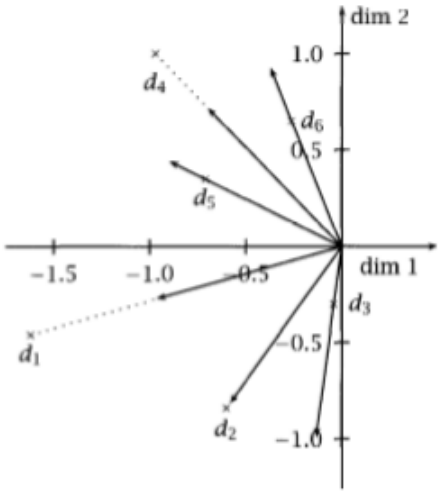
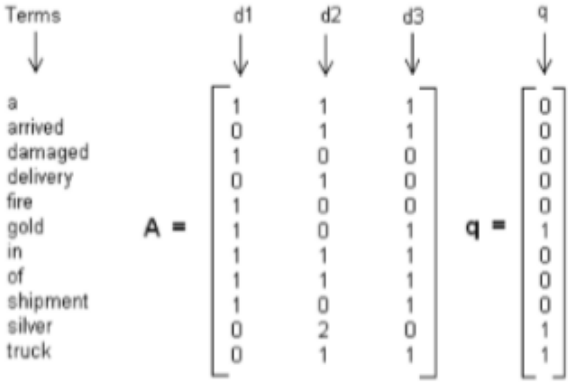


Figure 15.6 Dimensionality reduction. The documents in matrix 15.5 are shown after the five-dimensional term space has been reduced to two dimensions. The reduced document representations are taken from figure 15.11. In addition to the document representations $d_1, \ldots, d_6$, we also show their length-normalized vectors, which show more directly the similarity measure of cosine that is used after LSI is applied.

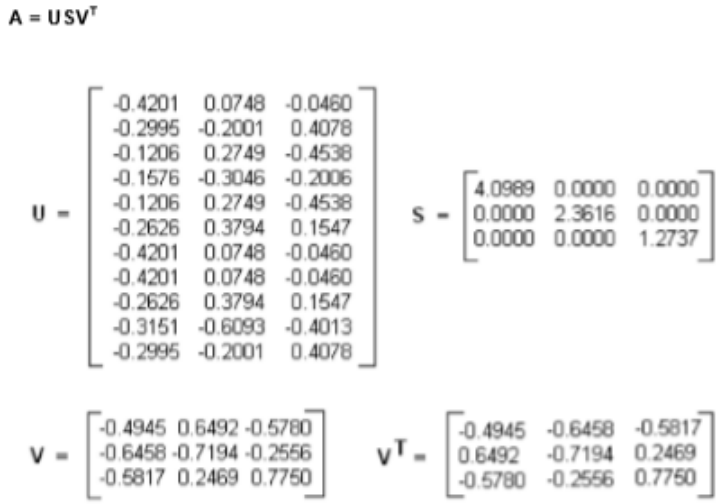
Fonte: Foundations of statistical natural language processing.

Figura 4 – Primeiro passo representação dos documentos em forma matricial



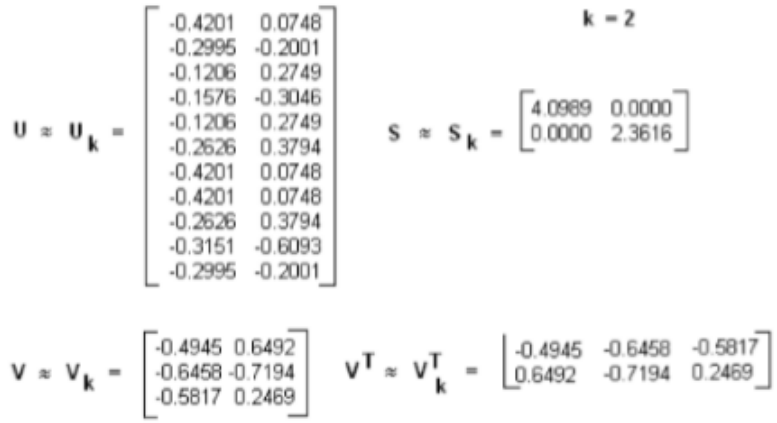
Fonte: Grossman and Frieder's Information Retrieval, Algorithms and Heuristics.

Figura 5 – Segundo passo decompor a matriz A utilizando SVD em $A = USV^t$



Fonte: Grossman and Frieder's Information Retrieval, Algorithms and Heuristics.

Figura 6 – Terceiro passo realizar a diminuição dimensional, neste caso foi escolhida uma aproximação em duas dimensões

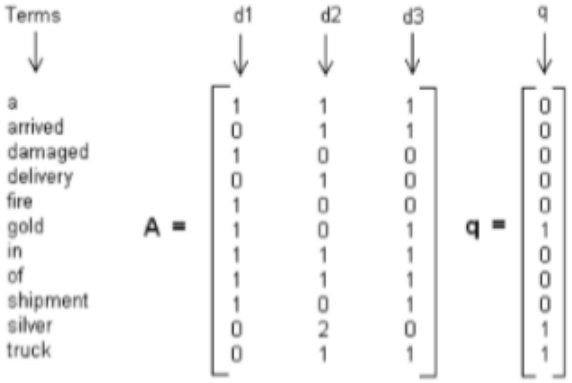


Fonte: Grossman and Frieder's Information Retrieval, Algorithms and Heuristics.

Figura 7 – Quarto passo encontrar as coordenadas dos vetores que representam os documentos no espaço dimensionalmente reduzido

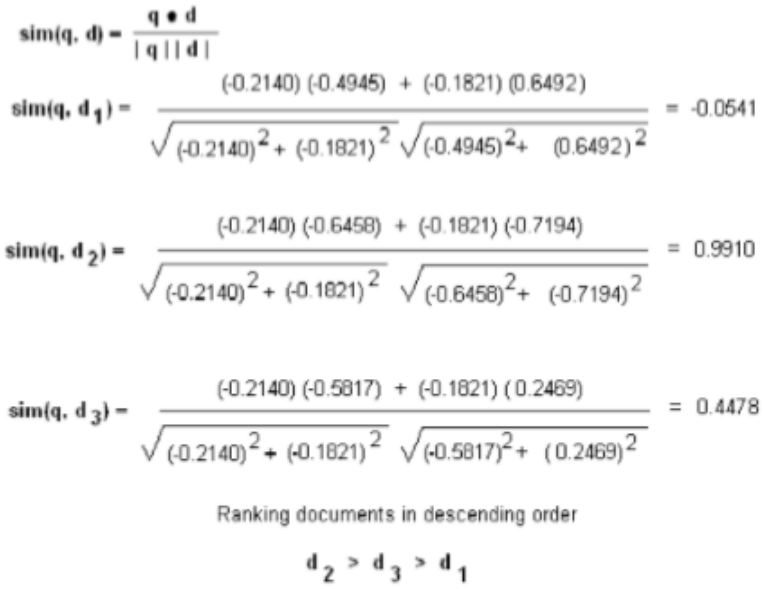
a) $d1 = (-0.4945, 0.6492)$
b) $d2 = (-0.6458, -0.7194)$
c) $d3 = (-0.5817, 0.2469)$

Figura 8 – Quinto passo encontrar a nova consulta (query) no espaço dimensionalmente reduzido $q = q^t U_k S_k^{-1}$.



Fonte: Grossman and Frieder's Information Retrieval, Algorithms and Heuristics.

Figura 9 – Sexto passo ordenar os documentos de acordo com a similaridade dos cossenos.



Fonte: Grossman and Frieder's Information Retrieval, Algorithms and Heuristics.

7 Implementação

A implementação da biblioteca foi realizada utilizando as linguagens JAVA e SCALA.

7.0.1 Versionamento

Para o versionamento foi utilizando o sistema de controle de versionamento open source chamado git, temos duas branchs dentro do repositório que contém o projeto, uma branch develop que possui as versões em desenvolvimento do projeto e a outra branch master que possui as versões estáveis do projeto.

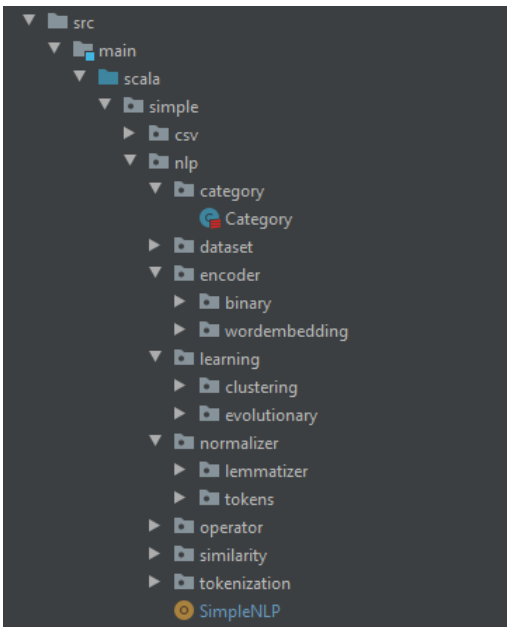
O projeto pode ser acessado no repositório do github: https://github.com/willer007/Simple-NLP

7.0.2 Automação de compilação

Para automação de compilação foi utilizado o sistema de automação de compilação open-source gradle, que gerencia as dependências e compilação do projeto.

7.0.3 Estrutura do projeto

Figura 10 – Estrutura de projeto



Fonte: Elaborada pelo autor.

Na imagem acima podemos ver a estrutura adotada para este projeto, cada seta indica uma pasta, chamaremos cada pasta de pacote, nesta versão inicial temos o pacote csv que

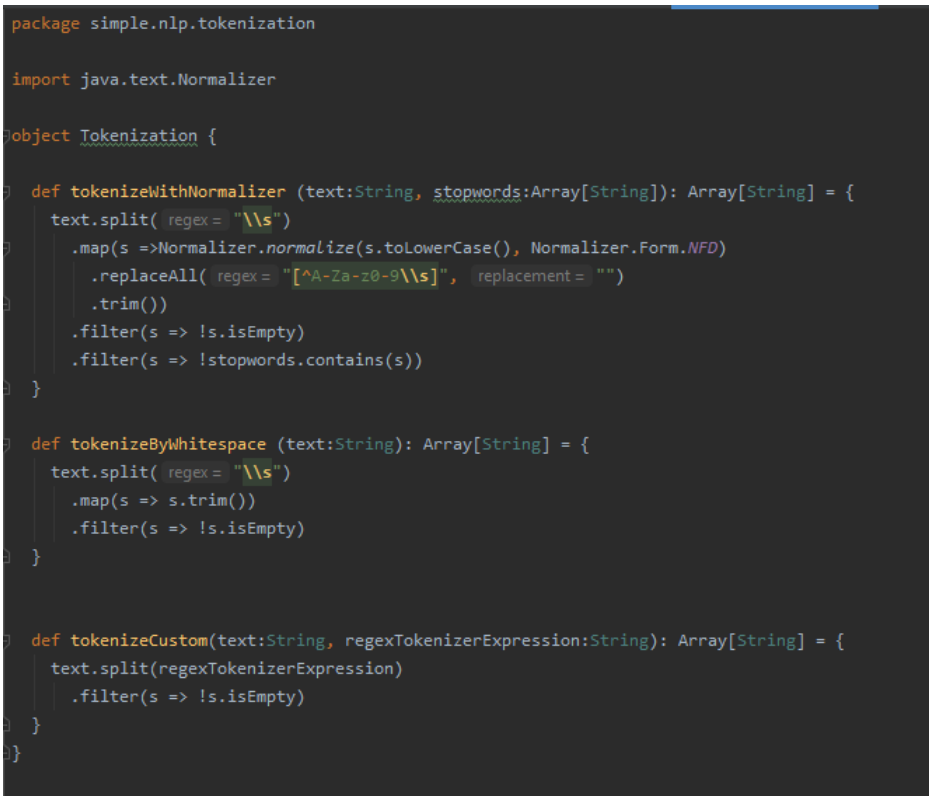
possui ferramentas para manipulação de arquivos com extensão ".csv"com a finalidade de facilitar a leitura e escrita de dados para os teste dos algorítimos.

7.0.4 Algorítimos

Neste capítulo serão descritos os principais algorítimos desenvolvidos neste trabalho bem como serão postas possibilidades de melhorias a serem desenvolvidas em trabalhos futuros.

7.0.4.1 Tokenization

Figura 11 – Tokenização



```
package simple.nlp.tokenization

import java.text.Normalizer

object Tokenization {

  def tokenizeWithNormalizer (text:String, stopwords:Array[String]): Array[String] = {
    text.split( regex = "\\s")
      .map(s =>Normalizer.normalize(s.toLowerCase(), Normalizer.Form.NFD)
        .replaceAll( regex = "[^A-Za-z0-9\\s]", replacement = "")
        .trim())
      .filter(s => !s.isEmpty)
      .filter(s => !stopwords.contains(s))
  }

  def tokenizeByWhitespace (text:String): Array[String] = {
    text.split( regex = "\\s")
      .map(s => s.trim())
      .filter(s => !s.isEmpty)
  }


  def tokenizeCustom(text:String, regexTokenizerExpression:String): Array[String] = {
    text.split(regexTokenizerExpression)
      .filter(s => !s.isEmpty)
  }
}
```

Fonte: Elaborada pelo autor.

Foram desenvolvidas três funções de tokenização diferentes, todas devem receber no mínimo um parâmetro $text$ do tipo $String$ que representa o texto a ser tokenizado e devolvem um vetor do tipo $Array[String]$ contendo o texto na forma de um vetor de tokens.

Para implementação futura pretendo desenvolver bases nativas de palavras de parada (stopwords) e uma base de expressões regulares que contemplem as diferenças lexicais de cada linguá.

7.0.4.2 Binary Encoding

O algoritmo de binary encoding desenvolvido neste trabalho funciona da seguinte forma.

Primeiro precisamos chamar o método $loadTokens(setTokens : Set[String])$ este método requer um parâmetro do tipo $Set[String]$ que deverá conter todos os tokens presentes na base a ser codificada, um Set é uma estrutura de dados que não aceita repetição assim este algorítimo garante que o set carregado não conterá tokens duplicados.

Após o carregamento dos tokens podemos realizar a transformação de texto tokenizado para o formato de um vetor de inteiros $Array[Int]$ pelo método $encode(arrayToken : Array[String])$, contanto que todos os tokens presentes neste texto estejam contidos em $setTokens$ e anteriormente tenham sido carregados através do método $loadTokens(setTokens : Set[String])$. O método $encode(arrayToken : Array[String])$ retorna um vetor de inteiros do tamanho de $setTokens$ considerando o número de ocorrências de cada token no texto, para realização do enconde sem consideração do número de ocorrências podemos utilizar o método $encodeWithoutCount(arrayToken : Array[String])$ que considera apenas a ocorrência ou não do termo no texto.

Figura 12 – Binary Encoding

```
object EncoderBinary {
  var tokenEncoder: Map[String, Int] = Map.empty
  var tokenDecoder: Map[Int, String] = Map.empty
  var numberOfTokens = 0

  def loadTokens(setTokens: Set[String]): Unit = {
    numberOfTokens = 0
    for (t <- setTokens) {
      tokenEncoder += (t -> numberOfTokens)
      tokenDecoder += (numberOfTokens -> t)
      numberOfTokens += 1
    }
  }

  def encode(arrayToken: Array[String]): Array[Int] = {
    var tokensEncoded = Array.ofDim[Int](numberOfTokens)
    arrayToken.foreach(token => {
      var tokenPosition = tokenEncoder(token)
      var tokenOccurrence = (tokensEncoded(tokenPosition))
      tokenOccurrence = (tokenOccurrence + 1)
      tokensEncoded(tokenPosition) = tokenOccurrence
    })
    tokensEncoded
  }

  def encodeWithoutCount(arrayToken: Array[String]): Array[Int] = {
    var tokensEncoded = Array.ofDim[Int](numberOfTokens)
    arrayToken.foreach(token => {
      tokensEncoded(tokenEncoder(token)) = 1
    })
    tokensEncoded
  }

  def decode(arrayToken: Array[Int]): Array[String] = {
    var stringDecoded: Array[String] = Array.empty
    for (tokenPosition <- arrayToken.indices) {
      if (arrayToken(tokenPosition) >= 1) { stringDecoded = tokenDecoder(tokenPosition) +: stringDecoded }
    }
    stringDecoded
  }
}
```

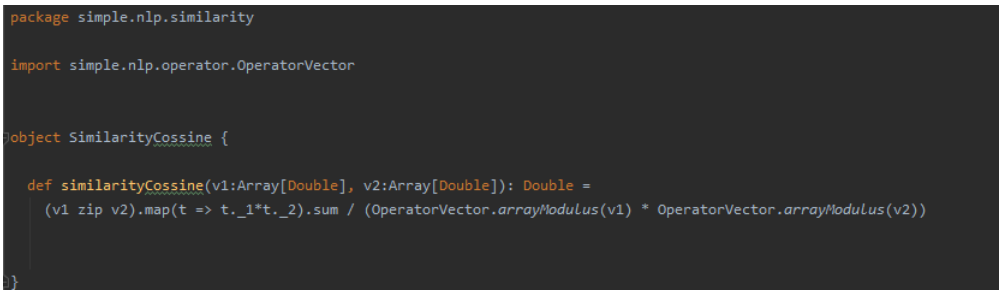
Fonte: Elaborada pelo autor.

Para a realizar a operação inversa, a tranformação de um vetor de inteiros em um vetor

de tokens podemos utilizar o método $decode(arrayToken : Array[Int])$.

7.0.4.3 Cossine Similarity

Figura 13 – Binary Encoding



```
package simple.nlp.similarity

import simple.nlp.operator.OperatorVector


object SimilarityCossine {

  def similarityCossine(v1:Array[Double], v2:Array[Double]): Double =
    (v1 zip v2).map(t => t._1*t._2).sum / (OperatorVector.arrayModulus(v1) * OperatorVector.arrayModulus(v2))


}
```

Fonte: Elaborada pelo autor.

Para o calculo da similaridade podemos utilizar a função $similarityCossine(v1 : Array[Double], v2 : Array[Double])$ que devolve o coeficiente de similaridade entre dois vetores em um valor tipo $Double$ baseado na similaridade de cossenos.

7.0.4.4 Genetic Algorithm

O algorítimo genético foi implementado em um primeiro momento utilizando uma função de otimização linear, a implementação deste algorítimo permite a alteração de uma serie de parâmetros como tamanho do cromossomo, número de cromossomos em cada geração , chance de mutação, numero de indivíduos escolhidos pelo elitismo e número de indivíduos escolhidos para a mutação.

A ideia deste algorítimo é que dada uma ou mais entradas do tipo $Array[Float]$ possamos calcular os pesos de uma função linear, tal que a combinação linear destes pesos com as entradas (inputs) resulte no valor objetivo.

Em implementações futuras pretendo utilizar este algoritmo base para implementação de algorítimos de clusterização.

7.0.4.5 Lloyds Algorithm

A implementação do Algorítimo de Lloyds desenvolvida neste trabalho permite a personalização do número de clusters ($numOfClusters$) e do número de dimensões de cada cluster ou seja o tamanho dos vetores de entrada($numOfDimension$), após setados estes parâmetros podemos inicializar o algoritmo através do método $optimize(inputs : Array[Array[Float]], iteration : int)$, este método recebe como parâmetros de entrada um vetor de vetores do tipo $Array[Array[Float]]$ e o número de iterações a ser realizadas pelo algorítimo e devolve um vetor de objetos do tipo $Cluster$

Figura 14 – Genetic Algorithm

```
package nlp.learning.evolutionary

import simple.nlp.learning.evolutionary.LearningGenetic

object LearningGeneticTest {

  def main(args: Array[String]): Unit = {
    LearningGenetic.initModel( generationSize = 9, chromosomeSize = 10)

    //MUTATION IN 10%
    LearningGenetic.setMutationChance(10)

    //MANTAIN BEST 3 INDIVIDUALS IN GENERATION
    LearningGenetic.setElitismSize(3)

    //EXECUTE 2 CROSSOVER WITH THE ELITE
    LearningGenetic.setCrossoverSize(2)


    val input:Array[Float] = Array(1,5,6,5,7,4,8,1,4,3)
    var testSingleInput = LearningGenetic.optimize( objective = 1,input, iterations = 200);


    val mutipleInput:Array[Array[Float]] = Array.fill(10) (input)
    var testMutipleInputs = LearningGenetic.optimize( objective = 1,mutipleInput, iterations = 200);

  }

}
```

Fonte: Elaborada pelo autor.

Figura 15 – Lloyds Algorithm

```
package nlp.learning.clustering

import simple.nlp.learning.clustering.LearningLloyds

object LearningLloydsTest {

  def main(args: Array[String]): Unit = {

    LearningLloyds.initModel( numOfClusters = 10,  numOfDimension = 10)

    var inputs: Array[Array[Float]] = Array.empty
    for (num <- 1 to 10) {
      inputs = inputs :+ Array.fill[Float](10)(num.toFloat)
    }


    var test = LearningLloyds.optimize(inputs, iterations = 100)

  }
}
```

Fonte: Elaborada pelo autor.

Os objetos do tipo Cluster possuem dois atributos $centroid$ um vetor do tipo $Array[Float]$, que representa o centro do cluster, e $data$ um vetor de vetores do tipo $ArrayBuffer[Array[Foat]]$, que contém os inputs atribuídos a este cluster.

7.0.4.6 Latent Semantic Indexing

Para utilização do algorítimo LSI implementado neste trabalho teremos que utilizar o método $encode(inputs : Array[Vector], numOfDimensions : Int)$ este método realiza a modelagem das entradas $inputs$ para o modelo word embending de dimensões

Figura 16 – Cluster

```
package simple.nlp.learning.clustering

import scala.collection.mutable.ArrayBuffer
import scala.util.Random


case class Cluster(centroid: Array[Float], data: ArrayBuffer[Array[Float]])

object Cluster {
  def addData(cluster: Cluster, data: Array[Float]): Unit = cluster.data += data
  def setCentroid(cluster: Cluster, centroid: Array[Float]) = Cluster(centroid, cluster.data)
  def createCluster(size:Int): Cluster = Cluster(Array.fill(size) {Random.nextFloat()}, ArrayBuffer.empty)
  def createCluster(centroid: Array[Float]):Cluster = Cluster(centroid,ArrayBuffer.empty)

}
```

Fonte: Elaborada pelo autor.

$numOfDimensions$ após codificados os dados, podemos codificiar a querys a ser utilizadas, com o método $encodeQuery(query : Array[Double])$, então teremos os inputs codificados e a query codificada e a partir da ai podemos executar toda sorte de operações desejadas, como ordenar documentos de um espaço semanticamente latente de acordo com a similaridade de cossenos com a query codificada,

Figura 17 – LSI

```
object EncoderLSITest {

  def main(args: Array[String]): Unit = {

    var inputs:Array[Vector] = Array.empty


    inputs = inputs :+ Vectors.dense( firstValue = 1, otherValues = 0,1,0,1,1,1,1,0,0)
    inputs = inputs :+ Vectors.dense( firstValue = 1, otherValues = 1,0,1,0,0,1,1,0,2,1)
    inputs = inputs :+ Vectors.dense( firstValue = 1, otherValues = 1,0,0,0,1,1,1,1,0,1)

    var test = EncoderLSI.encode(inputs, numOfDimensions = 2)

    var query:Array[Double] = Array(0,0,0,0,0,1,0,0,0,1,1)
    var testQuery = EncoderLSI.encodeQuery(query)


  }

}
```

Fonte: Elaborada pelo autor.

8 Resultados

Foi desenvolvida uma versão inicial da biblioteca SimpleNLP, os métodos implementados foram testados e os resultados foram positivos. Contudo não foi possível realizar a implementação do algorítimo de clusterização com o algorítimo genético em tempo hábil, portando deixei os dois algorítimos funcionando de forma isolada, assim temos uma implementação de algorítimo genético para otimização e uma implementação de clusterização baseada no algorítimo de Lloyds. A implementação da biblioteca SimpleNLP foi mais morosa que o previsto impactando na experimentação dos métodos desenvolvidos a análise de aspectos psicológicos.Para análise de aspectos psicológicos foi possível apenas experimentar a implementação do algorítimo LSI.

Figura 18 – Aplicação do Algoritimo LSI

```
def main(args: Array[String]): Unit = {

 //NORMALIZAÇÃO E CARREGAMENTO DO DATASET
  var dataset = SimpleCSV.readCSV( path = "C:\\Users\\Willer\\Desktop\\tcc_dataset\\tcc_dataset_tweets_100k.csv")
    .replaceAll( regex = "[,.!?\"]", replacement = "")
    .replaceAll( regex = "@\\w+", replacement = "")
    .replaceAll( regex = "http.+\\s", replacement = "")
    .toLowerCase()

  //SEPARAÇÃO DO DATASET EM MULTIPLAS LINHAS
  var datasetRows = dataset.split( regex = "\r\n").slice(0,1000)

  //CARREGAMENTO DE TODOS OS TOKENS PERTENCENTES AO DATASET
  var allTokens: Set[String] = SimpleNLP.Tokenizator.tokenizeByWhitespace(datasetRows.reduce((a,b) => a + " " + b)).toSet
  SimpleNLP.EncoderBinary.loadTokens(allTokens)

  //TOKENIZACAO
  var datasetTokenized = datasetRows.map(row => SimpleNLP.Tokenizator.tokenizeByWhitespace(row))
  //ENCODING
  var datasetEncoded = datasetTokenized.map(row => SimpleNLP.EncoderBinary.encodeWithoutCount(row))
  //EXPRESSAO FELIZ
  var happyExpression =  SimpleNLP.EncoderBinary.encodeWithoutCount(Array("happy"))
  //EXPRESSAO TRISTE
  var sadExpression =  SimpleNLP.EncoderBinary.encodeWithoutCount(Array("sad"))
  //ENCODE SEGUNDO MODELO LSI
  var encodedDataset   =datasetRows zip SimpleNLP.EncoderLSI.encode(datasetEncoded, numOfDimensions = 30)
  var happyLSI = SimpleNLP.EncoderLSI.encodeQuery(happyExpression)
  var sadLSI = SimpleNLP.EncoderLSI.encodeQuery(sadExpression)
  //EXPRESSAO FELIZ MODELO LSI
  var happy = encodedDataset.
    sortBy(data => SimpleNLP.SimilarityCossine.similarityCossine(data._2,happyLSI))
  var happyText:String = happy.map(d => d._1).reduce((a,b) =>a + "\n" +b)
  SimpleCSV.writeCSV( path = "C:\\Users\\Willer\\Desktop\\tcc_dataset\\happy.txt",happyText)
  //EXPRESSAO TRISTE MODELO LSI
  var sad = encodedDataset.
    sortBy(data => SimpleNLP.SimilarityCossine.similarityCossine(data._2,sadLSI))
  var sadText:String = sad.map(d => d._1).reduce((a,b) =>a + "\n" +b)
  SimpleCSV.writeCSV( path = "C:\\Users\\Willer\\Desktop\\tcc_dataset\\sad.txt",sadText)
}
```

Fonte: Elaborada pelo autor.

Utilizei o algorítimo LSI para analisar as expressões "happy"e "sad"em cerca de mil tweets, reduzindo dimensionalmente cada documento a 30 dimensões em um espaço latente sematicamente, ordenei os documentos de acordo com a similaridade com as querys codificadas para o espaço latente semanticamente: sad e happy, em dois documentos: "sad.txt"e "happy.txt",

da analise dos documentos processados o documento menos similar a expressão happy foi: - i think i need to find better anti-depressants i think this paxil/wellbutrin combo is losing its efficacy", enquanto que o documento mais similar a expressão sad foi : "that's sad", dessa análise penso que talvez haja um relacionamento dos termos "sad"e "happy"com a depressão e seja interessante pensarmos para o processamento de linguagem natural a depressão não como a abundancia de termos relacionados a tristeza, mas sim como a ausência de termos relacionados a felicidade, contudo mais estudos têm de ser realizados para análise dessa patologia e verificação do seu relacionamento com a linguá.

9 Conclusão

O desenvolvimento de uma biblioteca para processamento de linguagem natural consome tempo e requer ampla revisão bibliográfica, estas questões foram grandes impeditivos para ao desenvolvimento deste trabalho, contudo a implementação realizada foi um sucesso, os algorítimos desenvolvidos são altamente customizáveis, inclusive é possível reescreve-los sem maiores dificuldades dado que o acoplamento entre os algorítimos é extremamente baixo. Foi possível nesta primeira versão da biblioteca implementar alguns dos algorítimos mais utilizados para o Processamento de Linguagem Nautural de forma satisfatória e funcional.

Devido a questão da complexidade da implementação da biblioteca não houve tempo hábil para experimentação de todos os algorítimos para análise de aspectos psicológicos, foi experimentada apenas a implementação do algoritmo de LSI e os resultados foram promissores, da análise realizada foi possível traçar um possível relacionamento entre a ausência de termos relacionados a felicidade ("happy") em um documento estar atrelada a depressão.

Das análises realizadas penso que seja possível analisar aspectos psicológicos em texto, quiça inclusive processar diagnósticos para patologias que acomentem uma grande parcela da população.

Referências

GROSSMAN, D. A.; FRIEDER, O. *Information retrieval: Algorithms and heuristics*. [S.l.]: Springer Science & Business Media, 2012. v. 15.

LEITE, J. E. R. Fundamentos da linguística. *Lingua Portuguesa e Libras: teorias e práticas. Joao Pessoa. Ed Universitaria da UFPB*, v. 1, p. 171–232, 2010.

MANNING, C. D.; MANNING, C. D.; SCHÜTZE, H. *Foundations of statistical natural language processing*. [S.l.]: MIT press, 1999.

MAULIK, U.; BANDYOPADHYAY, S. Genetic algorithm-based clustering technique. *Pattern recognition*, Elsevier, v. 33, n. 9, p. 1455–1465, 2000.

NETO, M. L. R. O discurso e as narrativas na vivência da depressão. *Psicologia, Saúde & Doenças*, Sociedade Portuguesa de Psicologia da Saúde, v. 6, n. 2, p. 131–138, 2005.

ORLANDI, E. P. *O que é linguística*. [S.l.]: Brasiliense, 2017.

PINHEIRO, V. *SIM: Um Modelo Semântico Inferencialista para Expressão e Raciocínio em Sistemas de Linguagem Natural*. Tese (Doutorado) — Phd Thesis, Universidade Federal do Ceará, 2010.

ROCHA, L. M.; CAPPABIANCO, F. A. M.; FALCÃO, A. X. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, Wiley Periodicals, v. 19, n. 2, p. 50–68, 2009.

STONEBRAKER, M.; AGRAWAL, R.; DAYAL, U.; NEUHOLD, E. J.; REUTER, A. Dbms research at a crossroads: The vienna update. In: *VLDB '93 Proceedings of the 19th International Conference on Very Large Data Bases*. [S.l.]: Morgan Kaufmann Publishers Inc., 1993. p. 688–692.