

ESCOLA DE MATEMÁTICA APLICADA DA FUNDAÇÃO GETÚLIO
VARGAS
PROGRAMA DE GRADUAÇÃO EM MATEMÁTICA APLICADA

PEDRO HENRIQUE F. AMARO

METODOLOGIA PARA ANÁLISE DE QUALIDADE EM
TEXTOS

TRABALHO DE CONCLUSÃO DO CURSO

RIO DE JANEIRO

2015

PEDRO HENRIQUE F. AMARO

METODOLOGIA PARA ANÁLISE DE QUALIDADE EM TEXTOS

Trabalho de conclusão do curso apresentado ao Programa de Graduação em Matemática Aplicada da Escola de Matemática Aplicada da Fundação Getúlio Vargas como requisito parcial para obtenção do grau de “Graduando em Matemática Aplicada”.

Orientador: Renato Rocha Souza

Co-orientador: Paulo Cezar P. Carvalho

RIO DE JANEIRO

2015

Dedico este trabalho aos meus pais e irmãos pelo exemplo, incentivo, amor e carinho. Aos meus amigos pela convivência, apoio e atenção nos momentos alegres e tristes.

AGRADECIMENTOS

Agradeço a Escola de Matemática Aplicada da Fundação Getúlio Vargas e a todos os professores responsáveis por sua criação e manutenção da qualidade do ensino. Sem dúvidas, eu não poderia ter feito escolha melhor quanto a minha profissão e quanto a escola em que estudei.

Agradeço a minha família - meus pais, Márcia e Ilki, a minha avó, Lúcia e aos meus irmãos - e amigos, por terem me dado o suporte necessário e apoio durante todos os anos de curso e principalmente nos momentos mais estressantes e difíceis.

Agradeço em especial ao meu irmão João Vitor, por ter encontrado as palavras certas quando eu mais precisei, e ter me dado esperanças quando eu mais desacreditei de mim.

Não poderia deixar de agradecer ao meu orientador Renato Rocha Souza, por ter tido a paciência que sempre lhe envolve e pelo exemplo de ser humano que foi para mim, durante o curso e durante a definição do tema deste trabalho.

The bigger the ego, the harder the fall.

RESUMO

AMARO, Pedro. METODOLOGIA PARA ANÁLISE DE QUALIDADE EM TEXTOS. 43 f. Trabalho de conclusão do curso – Programa de Graduação em Matemática Aplicada, Escola de Matemática Aplicada da Fundação Getúlio Vargas . Rio de Janeiro, 2015.

Pesquisa bibliográfica e estudo sobre recursos de Processamento de Linguagem Natural (PLN) importantes para Metodologia para análise de qualidade em textos. Aplicação dos algoritmos estudados a uma base de textos (sugestões) de problema real do cotidiano.

Palavras-chave: Processamento de linguagem natural, sistemas de recuperação de informação, análise de qualidade textual

ABSTRACT

AMARO, Pedro. METHODOLOGY FOR QUALITY ANALYSIS IN TEXTS. 43 f. Trabalho de conclusão do curso – Programa de Graduação em Matemática Aplicada, Escola de Matemática Aplicada da Fundação Getúlio Vargas . Rio de Janeiro, 2015.

Bibliographical research and study on Natural Language Processing resources (NLP) important for Methodology for quality analysis in texts. Application of the algorithms studied a data base of texts (suggestions) of real everyday problem.

Keywords: Natural language processing, information retrieval systems, textual quality analysis

LISTA DE FIGURAS

FIGURA 1	–	Árvore sintática - "O João comprou todos os livros."	13
FIGURA 2	–	Graph attributes	32
FIGURA 3	–	The graph	33
FIGURA 4	–	Análise morfo-sintática completa	34
FIGURA 5	–	Árvore - Análise morfo-sintática completa	35
FIGURA 6	–	Árvore - Estrutura de dependência	36

SUMÁRIO

1 INTRODUÇÃO	9
1.1 MOTIVAÇÃO	9
1.2 DESCRIÇÃO DO PROBLEMA	9
1.3 OBJETIVO DO TRABALHO	10
1.4 ORGANIZAÇÃO DO TRABALHO	11
2 DEFINIÇÕES SOBRE LINGUAGEM	12
2.1 LINGÜÍSTICA E GRAMÁTICA	12
2.1.1 Morfologia	12
2.1.2 Sintaxe	13
2.1.3 Semântica	14
2.1.4 Entidades Nomeadas - NEs	14
3 PROCESSAMENTO DE LINGUAGEM NATURAL - NLP	15
4 PROBLEMAS A TRATAR	18
4.1 DETECÇÃO DE CÓPIAS	18
4.2 ANÁLISE SINTÁTICA (PARSING)	20
4.3 RECONHECIMENTO DE ENTIDADES NOMEADAS - NER	21
4.4 IMPORTÂNCIA DAS PALAVRAS NO TEXTO	22
4.5 COMPLEXIDADE DO TEXTO	24
5 MODELO DE PROCESSAMENTO DO TEXTO	26
5.1 SEQUENCE MATCHER	26
5.2 TERM FREQUENCY – INVERSE DOCUMENT FREQUENCY	27
6 APLICAÇÃO DO MODELO	29
6.1 DATA SET	29
6.2 OBJETIVO DA APLICAÇÃO	29
6.3 DETECÇÃO DE CÓPIAS - SEQUENCEMATCHER	30
6.3.1 Índice de similaridade	30
6.3.2 Que valor de r considerar para dois textos como cópias?	32
6.3.3 O grafo	32
6.4 ANALISADOR SINTÁTICO PALAVRAS	33
6.5 IMPORTÂNCIA DAS PALAVRAS NO TEXTO - TF-IDF	36
7 CONCLUSÃO	39
REFERÊNCIAS	40
Apêndice A – PYTHON CODE AND RESULT: SEQUENCEMATCHER	41
Apêndice A – CÓDIGO PYTHON: TF-IDF	42

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A quantidade de dados, dos mais diversos gêneros, cresce muito a cada dia. O desenvolvimento das sociedades, a transformação de quase todas as ações em processos e estes sendo registrados das mais variadas formas vem transformando o mundo em um gigante abrigo de conjuntos de dados. Há muitos anos este comportamento, como cultura de organização, já podia ser notado. Os registros, que hoje naturalmente são armazenados em sistemas digitais, antes já eram produzidos em grande escala e arquivados com as tecnologias disponíveis. O ser humano não está apto a analisar todo este volume de informação disponível, principalmente em forma de texto, e portanto cada vez mais há a necessidade do auxílio computacional para a construção, armazenamento e consumo desta informação.

O Processamento de linguagem natural é a tecnologia para lidar com o nosso produto mais precioso: a linguagem humana, como aparece em e-mails, páginas da web, tweets, descrições de produtos, artigos de jornal, mídia social, e artigos científicos, em milhares de línguas e variedades. Na última década, as aplicações de processamento de linguagem natural tornaram-se parte da nossa experiência cotidiana - da ortografia e correção gramatical em processadores de texto à tradução automática na web, desde a detecção de e-mail de spam até pergunta/resposta automática, ou identificar opiniões das pessoas sobre produtos/serviços, e ainda extrair compromissos a partir de e-mails.

1.2 DESCRIÇÃO DO PROBLEMA

Um problema comum para pesquisadores de todas as áreas é saber extrair de um grande conjunto de dados informações que sejam relevantes e úteis aos seus interesses. Em geral o acervo de dados é muito extenso e ultrapassa a condição do ser humano de interpretar e analisar o conjunto total.

Este trabalho trata do problema da análise qualitativa de textos em linguagem natural, ou seja na forma como humanos escrevem e se comunicam com humanos, buscando estudar algoritmos e metodologias já propostas por outros autores no campo de Processamento de Linguagem Natural - NLP (*Natural Language Processing*). O problema central, e que serviu de cenário para o estudo, é analisar a base de uma pesquisa de sugestões sobre a antiga constituição brasileira, realizada com civis na década de 80, afim de identificar a qualidade e relevância das sugestões.

Os textos foram escritos por indivíduos com distintos padrões sociais e de diversas localidades do território brasileiro e, inicialmente, os metadados não foram levados em consideração para a pesquisa. O problema foi tratado puramente em cima das sugestões escritas, o que envolveu o estudo e solução de um conjunto de problemas relacionados ao processamento de linguagem natural, necessários para se chegar a conclusões sobre a qualidade e pertinência destas sugestões.

O conceito de 'qualidade' e 'relevância' é bastante subjetivo e varia de acordo com os interesses do pesquisador. Neste caso, os interesses para a classificação da qualidade dos textos desta base de dados são bem definidos. Este assunto está sendo estudado por um doutor da University of Texas, em uma pesquisa no departamento de Direito da Fundação Getulio Vargas, e este estudo foi voltado para auxiliar o mesmo.

1.3 OBJETIVO DO TRABALHO

O objetivo deste trabalho foi estudar alguns algoritmos e modelos matemáticos fundamentais para o processamento da linguagem humana e como podemos usá-los para estruturar um modelo para análise de qualidade em textos. Além disso, aplicar o estudo a um problema real, afim de possibilitar o entendimento do potencial destas ferramentas e a importância da matemática aplicada ao processamento de textos.

Os modelos apresentados podem ser usados para melhorar a representação de textos por uma perspectiva de Recuperação de Informação, afim de permitir o acesso a informações relevantes de grandes bases de dados de forma mais objetiva e concisa.

O trabalho concentra-se na aplicação a uma base de sugestões sobre a antiga Constituição do Brasil como corpus, na língua portuguesa. Como consequência, o processo proposto de extração de informações dos textos poderá ser usado para enriquecer futuros trabalhos

sobre a Constituição, assim como o aproveitamento dos resultados para implementação de outros modelos e algoritmos, possivelmente em continuidade posterior.

1.4 ORGANIZAÇÃO DO TRABALHO

No segundo capítulo, são abordados alguns conceitos fundamentais para a compreensão do objeto principal de estudo, a linguagem escrita como representação da comunicação entre seres humanos, em especial a formalização estrutural da escrita da língua portuguesa.

No terceiro capítulo, são apresentados os problemas envolvidos na área de processamento da linguagem natural, as principais aplicações, limitações e ferramentas utilizadas.

No quarto capítulo, são abordados os principais problemas que foram considerados para analisar a qualidade de textos, em especial as sugestões que compõem a base cenário deste trabalho.

No quinto capítulo, é proposto um modelo baseado em artigos e algoritmos já estudados para abordagem dos problemas descritos no capítulo três.

No sexto capítulo, é apresentada a aplicação do modelo proposto no capítulo cinco à base de dados escolhida para este estudo, e os resultados obtidos.

No sétimo capítulo, as conclusões obtidas sobre o estudo dos modelos e algoritmos e a avaliação dos resultados da aplicação do modelo.

2 DEFINIÇÕES SOBRE LINGUAGEM

2.1 LINGUÍSTICA E GRAMÁTICA

A linguística é a ciência da linguagem - sons, palavras e regras gramaticais. Palavras em suas mais variadas línguas são um conjunto finito, mas as sentenças formadas por elas, não. Este é o aspecto diferencial e criativo da linguagem humana, que, diferente de linguagens animais, não são essencialmente apenas respostas aos estímulos.

As regras estruturais de que regem uma língua, também chamada de gramática, são aprendidos conforme se adquire uma língua. Estas regras incluem fonologia, o sistema do som, a morfologia, a estrutura das palavras, sintaxe, a combinação de palavras em sentenças, semântica, as maneiras pelas quais sons/escrita e significados estão relacionados, e o léxico, ou dicionário mental de palavras. Quando se conhece uma linguagem, você sabe as palavras daquela linguagem, ou seja, unidades de som e escrita que estão relacionadas com significados específicos.

Neste capítulo serão apresentados alguns conceitos da linguística utilizados no decorrer do trabalho, bem como divisões da gramática e seus aspectos.

2.1.1 MORFOLOGIA

Na linguística, morfologia é a identificação, análise e descrição da estrutura de morfemas de uma determinada linguagem e outras unidades linguísticas, tais como palavras raiz, afixos, partes do discurso, entonações e tensões, ou contexto implícito. Por outro lado, a tipologia morfológica é a classificação das línguas de acordo com a sua utilização de morfemas, enquanto lexicologia é o estudo dessas palavras que formam a "memória" de uma língua. Desta forma, a morfologia é o ramo da linguística que estuda os padrões de formação de palavras dentro e entre línguas e tentativas de formular regras que modelam o conhecimento dos falantes dessas línguas.

2.1.2 SINTAXE

Na linguística, sintaxe é o conjunto de regras, princípios e processos que regem a estrutura das frases em um determinado idioma. O termo sintaxe é também usado para se referir ao estudo de tais princípios e processos. O objetivo de muitos sintaticistas é descobrir as regras de sintaxe comuns a todas as línguas.

Árvores sintáticas Representam as estruturas dos componentes de uma sentença. Nesta árvore, um nó raiz "F" representa a sentença (frase), os nós não terminais representam os sintagmas ¹, e as folhas representam as palavras, marcadas com as classificações morfológicas.

As árvores sintáticas são usadas para identificar os componentes que formam uma oração: sujeito, predicado e objeto, e complementos nominais. Esta ideia será muito útil para a compreensão do tópico, mais a diante, sobre "Parsing".

Exemplo:

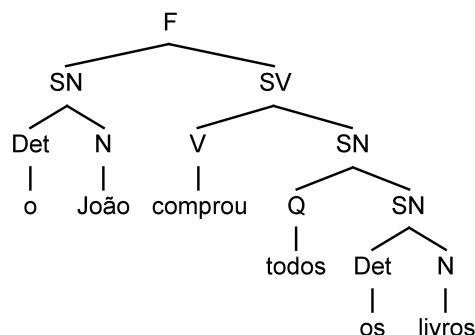


Figura 1: Árvore sintática - "O João comprou todos os livros."

Na Figura 1, a sentença F é composta por um sintagma nominal (SN) e um sintagma verbal (SV). O sintagma verbal (SV) é constituído por um verbo e um sintagma nominal (SN), que por sua vez é composto também por outro sintagma nominal. As palavras estão marcadas com suas classificações morfológicas: Det (Determiner/artigo), N (Noun/nome), V (verb/verbo), Q (Qualifiers/adjetivo) (GALASSO, 2000).

¹Sintagma é uma unidade sintática composta de um ou mais vocábulos que forma orações.

2.1.3 SEMÂNTICA

Na linguística, semântica é o subcampo que se dedica ao estudo do significado, como inerente aos níveis de palavras, frases, sentenças e unidades maiores de discurso (textos). O estudo da semântica está também estreitamente ligado aos assuntos de representação, referência e denotação. O estudo de base da semântica é orientada para a análise do significado dos sinais, bem como o estudo das relações entre diferentes unidades linguísticas. Uma das principais preocupações é a forma como o significado geral é atribuído a blocos de texto, possivelmente como resultado da composição de pequenas unidades de significado (palavras). Tradicionalmente, a semântica incluiu o estudo do sentido e referência denotativa, as condições de verdade, a estrutura do argumento, os atributos temáticos, a análise de discurso, e a ligação de todos estes à sintaxe.

2.1.4 ENTIDADES NOMEADAS - NES

As Entidades Nomeadas (Named Entities - NEs) são estruturas que fazem referência a termos, simples ou compostos, que representam pessoas, lugares ou organizações de conhecimento público. A correta atribuição de um termo à sua entidade é fundamental para definir com exatidão a quem ou que se refere um determinado fato. Esta tarefa está atribuída ao Reconhecimento de Entidades Nomeadas (Named Entity Recognition - NER).

3 PROCESSAMENTO DE LINGUAGEM NATURAL - NLP

Sobre "linguagem natural" nós entendemos como a linguagem utilizada no cotidiano para a comunicação entre os seres humanos, escrita ou falada; linguagens como o nosso o Português, Inglês ou Alemão. Diferente das linguagens artificiais, como as linguagens de programação ou as notações matemáticas, as linguagens naturais foram se modificando e evoluindo de acordo com as gerações e por isso é tão difícil lidar com suas distintas regras. Vamos considerar o Processamento de Linguagem Natural - ou NLP (Natural Language Processing) - como qualquer tipo de solução computacional para manipular a linguagem natural.

Neste capítulo vamos abordar uma lista de algumas das tarefas mais utilizadas e pesquisadas em Processamento de Linguagem Natural. Algumas dessas tarefas tem aplicações reais diretas, enquanto outras geralmente servem como subtarefas para ajudar na resolução de tarefas maiores. O que distingue as tarefas não é só o volume de pesquisa dedicada a elas, mas o fato de que, para cada uma existe tipicamente uma configuração de problema bem definido, uma métrica padrão para avaliar a tarefa, um corpus padrão onde a tarefa pode ser avaliada.

- **Sumarização automática** Produzir um resumo legível de um pedaço de texto. Muitas vezes usado para fornecer resumos de texto de um tipo conhecido, como artigos na seção financeira de um jornal.
- **Correferência** Dada uma frase ou um texto maior, determinar quais palavras ("menções") referem-se aos mesmos objetos ("entidades"). Uma tarefa mais geral de correferência inclui a identificação de chamadas "relações de transição" envolvendo expressões referenciais. Por exemplo, em uma frase como "Ele entrou na casa de João pela porta da frente", "porta da frente" é uma expressão referencial e a relação de a ser identificada é o fato de que a porta referida é a porta da frente da casa de João.

- **A tradução automática** Traduz automaticamente o texto de uma linguagem humana para outra. Este é um dos problemas mais difíceis, e é membro de uma classe de problemas coloquialmente chamado de "AI-completos", ou seja, exigindo todos os diferentes tipos de conhecimento que os seres humanos possuem (gramática, semântica, fatos sobre o mundo real, etc.), a fim de resolver adequadamente.
- **Segmentação morfológica** Separar as palavras em morfemas individuais e identificar a classe dos morfemas. A dificuldade desta tarefa depende grandemente da complexidade da morfologia (ou seja, a estrutura de palavras) da língua sendo considerada. Inglês tem morfologia bastante simples, especialmente morfologia flexional e, portanto, muitas vezes é possível ignorar esta tarefa completamente e simplesmente modelar todas as formas possíveis de uma palavra (por exemplo, "aberto, abre-se, abriu, abertura") como palavras separadas.
- **Reconhecimento de entidades nomeadas (NER)** Dado um fluxo de texto, determinar quais itens no texto são nomes próprios, como pessoas ou lugares, e que tipo cada nome é (por exemplo, pessoa, localização, organização). Apesar da capitalização na letra inicial auxiliar no reconhecimento de entidades nomeadas em idiomas como Português, esta informação não pode ajudar a determinar o tipo de entidade nomeada, e em qualquer caso, é muitas vezes imprecisas ou insuficientes. Por exemplo, a primeira palavra de uma frase também é capitalizada, e entidades nomeadas geralmente se estendem por várias palavras, das quais apenas algumas são capitalizadas.
- **Geração de linguagem natural** Converter informações de bancos de dados de computador em linguagem humana legível.
- **Compreensão de linguagem natural** Converter blocos de texto em representações mais formais, como estruturas lógicas de primeira ordem que são mais fáceis para programas de computador manipular.
- **Reconhecimento óptico de caracteres (OCR)** Dada uma imagem que representa texto impresso, determinar o texto correspondente.
- **Parsing** Determinar a árvore de parsing (análise gramatical) de uma determinada frase. A gramática para as línguas naturais é ambígua e frases típicas podem ter várias análises possíveis.
- **Pergunta eletrônica** Dada uma questão de língua humana, determinar a sua resposta. As perguntas típicas têm uma resposta certa específica (como "Qual é a

capital do Brasil?”), Mas, por vezes, questões abertas também são consideradas (como ”Qual é o sentido da vida?”).

- **Extração de Relacionamento** Dado um pedaço de texto, identificar as relações entre entidades nomeadas (por exemplo, quem é casado com quem).
- **Análise de sentimento** Extrair informações subjetivas geralmente a partir de um conjunto de documentos, muitas vezes usando opiniões on-line para determinar ”polaridade” sobre objetos específicos. É especialmente útil para identificar as tendências da opinião pública nos meios de comunicação social para fins de marketing.
- **Segmentação de tópicos** Dado um pedaço de texto, separá-lo em segmentos cada um dos quais é dedicado a um tópico, e identificar o tema do segmento.
- **Ambiguidade** Muitas palavras têm mais de um significado; temos que escolher o sentido que faz mais sentido no contexto.

Em alguns casos, conjuntos de tarefas relacionadas são agrupadas em sub-campos de PNL que são muitas vezes considerados separadamente de PNL como um todo:

- **Recuperação de informação (IR)** Esta área é responsável pelo armazenamento, pesquisa e recuperação de informações. É um campo separado dentro da ciência da computação (relacionado a bancos de dados), mas que depende de alguns métodos de PNL.
- **Extração de informações (IE)** Extração de informação semântica do texto. Isso abrange tarefas como reconhecimento de entidades nomeadas, resolução de referência, extração de relacionamento, etc.
- **Processamento de voz** Isto abrange o reconhecimento de voz, texto-fala e tarefas relacionadas.

4 PROBLEMAS A TRATAR

A análise de qualidade de um texto gera diversos questionamentos por conta da subjetividade implícita neste tipo de problema. A primeira pergunta natural a ser feita é: o que é um bom texto ou um texto ruim? Que parâmetros e pontos deve-se levar em consideração para fazer este tipo de ponderação? As respostas variam de acordo com tipo do problema. Um exemplo prático é o de uma empresa hipotética que resolveu realizar uma pesquisa com seus clientes buscando avaliar a qualidade do atendimento dos seus vendedores. Os clientes deveriam preencher um campo "sugestões". Os dados gerados com a pesquisa são volumosos e por isso não é possível focar individualmente em cada sugestão. A escolha aleatória de alguns textos obviamente não seria a melhor opção. De que forma essa empresa poderia saber quais textos contêm as sugestões mais relevantes?

4.1 DETECÇÃO DE CÓPIAS

O problema de detecção de cópias é muito comum em análises de textos, em geral em pesquisas de opinião, notícias, matérias de blogs, artigos, etc. No caso dos dados que serviram de base para este estudo, identificar sugestões que sejam cópias idênticas ou textos similares, nos diz muito sobre a relevância desta sugestão. Existem alguns tipos de cópias, dentre elas cópias idênticas, cópias com palavras trocadas, cópia da ideia do texto, cópia metafórica e cópia de estilo (ALI et al., 2011).

Existem duas formas de se detectar uma cópia: manualmente ou computacionalmente. Assim como quase todas as análises textuais estudadas neste trabalho, um ser humano pode facilmente realizar manualmente. Mas devido a quantidade de dados, isso se torna uma função muito custosa e não eficaz. Para isso vamos abordar as soluções computacionais para este problema.

A detecção automatizada (com auxílio do computador), conta com diversos softwa-

res já desenvolvidos para esta finalidade, como: PlagAware, PlagScan, Check for Plagiarism, iThenticate, PlagiarismDetection.org, Academic Plagiarism, The Plagiarism Checker, Urkund, Docoloc (ALI et al., 2011), entre outros. Vamos tratar de duas abordagens para este problema: a clusterização e a contagem de sequências de palavras como discretizador.

A clusterização dos textos nos ajuda a encontrar conjuntos de textos que possuem certo grau de similaridade. O objetivo é particionar os objetos (textos, para a base em estudo as "sugestões") em *clusters* de forma que textos dentro de um *cluster* são similares e textos de *clusters* diferentes são diferentes. Com isso é possível a criação de categorias dos textos de forma *não-supervisionada*, ou seja, as classes não são definidas a priori.

A representação dos textos para a clusterização pode ser feita de alguns formas. A abordagem mais comum é através do TF-IDF (Term Frequency–Inverse Document Frequency), que mais a frente será abordada com mais detalhes para o estudo da importância das palavras no texto. Esta abordagem permite representar cada documento como um ponto em um espaço de tamanho T (tamanho do vocabulário).

$$\vec{d}_i = (d_{i1}, \dots, d_{iT}) \quad (1)$$

$$d_{ij} = T f_{ij} * Idf_j \quad (2)$$

Os elementos do vetor \vec{d} são definidos na equação acima e os termos serão explicitados mais a frente.

Agora que os textos estão representados por vetores, podemos calcular a similaridade entre dois textos através do cosseno. O $\cos(a,b)$ representa a correlação entre os documentos a e b . Os valores variam de 0 a 1 e quanto mais próximo de 1 maior a "proporcionalidade" entre estes vetores (textos).

$$sim(\vec{a}, \vec{b}) = \cos(\vec{a}, \vec{b}) \quad (3)$$

Após a representação dos textos através do método descrito acima, existem algoritmos utilizados para encontrar os centróides dos *clusters* e o que é utilizado para esta finalidade com mais frequência é o *K-means*. O algoritmo *K-means* encontra de forma iterativa os centróides, ou "pontos médios", dos clusters. A relação dos textos com os

clusters é feita de acordo com a similaridade dos centróides com os textos.

$$\vec{c} = \frac{1}{|C|} \sum_{d_i \in C} \vec{d}_i \quad (4)$$

Uma forma mais específica para de detectar cópias é através do merge de sequências dos textos. Este método tende a analisar cópias do tipo literal e baseado apenas nos períodos e sequências formados pelas palavras. Um texto completamente igual a outro terá uma grande, e única, sequência que compõe os textos. O objetivo deste método é identificar cópias que sejam possivelmente idênticas, considerando os possíveis erros de escrita e "lixos" (conceito que será definido mais a frente).

O método que será considerado (*SequenceMatcher*) para o modelo do problema em questão é um método deste tipo, pois o objetivo é identificar conjuntos de sugestões que foram direcionadas por uma única pessoa, ou seja, os textos que são claramente cópias uns dos outros. Com esse método foi possível clusterizar estes grupos e ainda identificar os "hubs"¹.

4.2 ANÁLISE SINTÁTICA (PARSING)

Parsing ou Analisador morfo-sintático é um recurso muito importante e utilizado em NER. Como dito anteriormente, nada mais é do que a análise das estruturas de um texto como períodos e as relações entre seus sintagmas, nominais e verbais. O particionamento vai ser muito usado auxiliando diversos outros recursos do PLN, como por exemplo na análise da complexidade de um texto, quando a profundidade das árvores sintáticas vão influenciar nesta complexidade.

O NLTK, biblioteca desenvolvida em Python e que já foi descrita em capítulo anterior, é uma ferramenta muito importante para o parsing. Além do Inglês como língua suportada, este pacote tem a grande vantagem de também suportar o idioma Português. Apesar disto, o módulo de particionamento do NLTK não funciona tão bem como outras opções disponíveis para o português.

Assim, como a Universidade de Stanford desenvolveu o "Stanford Parser"² para

¹Vértices altamente conectados de uma rede

²<http://nlp.stanford.edu/software/lex-parser.shtml>

a língua inglesa, e é hoje um dos sistemas de parsing mais utilizados para o Inglês, no português existe uma plataforma de *Grammar parser*, chamada PALAVRAS, desenvolvida por Eckhard Bick em seu projeto de Ph.D. Sobre esta plataforma que será abordado nos próximos capítulos, incluindo sua utilização para a geração de árvores das estruturas morfo-sintáticas. (SOUZA, 2008).

4.3 RECONHECIMENTO DE ENTIDADES NOMEADAS - NER

O problema de Reconhecimento de entidades nomeadas é um dos problemas da área de PLN que mais demandam pesquisa e diversos estudos estão em decorrência a respeito disto. A maior questão deste problema são as variações entre as línguas, entre os idiomas. Cada um com suas particularidades, muitas das vezes em que estruturas comuns entre os idiomas ocidentais compartilham as mesmas soluções em outros problemas de PLN, no problema de NER (Named-entity recognition) esta soluções são mais específicas.

O NER é o problema de encontrar os nomes próprios em um texto e classificá-los entre várias determinadas categorias de interesse ou a uma categoria padrão (as categorias usuais são: Pessoa, Organização e Locais). Alguns exemplos de entidades nomeadas (NE):

- **A Presidente do Brasil Dilma Rousseff** cancelou seus eventos. (Local/Pessoa)
- **ONU** considera que Terceira Guerra Mundial já iniciou.(Organização)

As abordagens estudadas de NER usam três mecanismos básicos de machine learning (aprendizado de máquinas, em português: Hidden Markov Models, Transformation-Based Learning and Support Vector Machines.(DUARTE; MILIDIÚ, 2007)

O **Hidden Markov Modeling (HMM)**(RABINER, 1989) é uma estrutura probabilística poderosa usada para modelar dados sequencial. O HMM é amplamente utilizado em PLN em tarefas como part-of-speech (POS) tagging, análise morfológica e reconhecimento de voz. Em HMM, temos dois conceitos básicos: observações e estados ocultos. Nas tarefas de PNL, a sequência de palavras que formam uma sentença são geralmente considerados como os dados observados, e os estados representam informação semântica relacionada com a sentença. Os parâmetros HMM são definidos para maximizar a log-verossimilhança entre a sentença e a semântica da informação. Os estados obtidos podem, então, ser mapeados para as tags semânticas gerando um classificador

PNL. O sucesso no processo de classificação é altamente dependente da escolha dos estados e respectivas correspondentes. Mesmo assim, modelos genéricos podem executar muito bem em alguns problemas. Uma maneira simples de modelar NER usando HMM é usar os NER-tags (PER (Pessoa), ORG (Organização), LOC (Local)) como os estados ocultos e os pos-tags como as observações. Cada sentença é então mapeado para a sua sequência de pos-tag.(DUARTE; MILIDIÚ, 2007)

O **Transformation Based Learning (TBL)** é um método simbólico de machine learning, introduzido por Eric Brill (BRILL, 1992). A idéia principal em um algoritmo de TBL é gerar um conjunto ordenado de regras que podem corrigir erros de marcação no corpus, que tenham sido produzidos por uma classificação suposição inicial, processo chamado de Linha de Base do Sistema (BLS, Baseline System). As regras são gerados de acordo com uma lista de modelos dados pelo desenvolvedor, que são voltados para capturar as combinações de características relevantes para o problema, corrigindo os erros sucessivamente gerados pelo BLS e também, pelo próprio TBL. Para o NER, este método vai utilizar como classificação inicial os *gazetteers*³ que existem disponíveis da língua portuguesa.

4.4 IMPORTÂNCIA DAS PALAVRAS NO TEXTO

O problema de identificar palavras importantes, ou palavras-chaves (em inglês, *keywords*) em um conjuntos de textos é primordial para a análise geral de um texto. A extração de palavras relevantes em um texto atualmente é uma técnica muito utilizada em diversas aplicações e é o primeiro passo, por exemplo, para modelar os tópicos de conteúdo de um *corpus*. Neste caso, através da extração das palavras corretas em uma consulta, um pesquisador poderia facilmente saber que documentos devem ser lidos e que documentos devem ser ignorados.

Para resolução deste tipo de problema temos diversos tipos de abordagens. Uma delas por meio de abordagens linguísticas. Essas abordagens são capazes de extrair informações de documentos usando informações linguísticas (morfológicas, sintáticas ou

³Índice de topónimos, ou gazetteer, é um dicionário geográfico ou diretório, uma importante referência para obter informações sobre lugares e nomes de lugar, utilizado em conjunto com um mapa ou um atlas completo.

semânticas) sobre um determinado texto. Outra abordagem linguística, como o usado em (Heid, 2000), extrai os termos relevantes usando expressões regulares.

Na mesma linha, temos abordagens baseadas em conhecimento. Estas abordagens são geralmente associadas a ontologias⁴ onde a idéia principal é obter um modelo representativo da realidade específica dos documentos analisados. Um exemplo para a extração de informações relevantes usando abordagens baseadas no conhecimento pode ser associado com o conhecimento da estrutura de documentos para, por exemplo, extrair palavras-chave a partir dos títulos e resumos de documentos científicos.

Alguns autores também tentaram utilizar Redes Neurais para fazer a extração de termos relevantes. Uma Rede Neural é um modelo de programação que se assemelha, de certa forma, a um modelo neural biológico. Aplicada a extração de informações, a aplicação mais comum é baseado em consultas eletrônicas. Um usuário consulta um conjunto de documentos e os verifica em uma rede neural se a consulta é relevante em um determinado documento ou não. Se for, esse documento é recuperado e apresentado ao utilizador. Em (DAS et al., 2008) uma técnica baseada em Redes Neurais é apresentada. A idéia básica é que cada um dos nós (ou neurônios) tem uma palavra da consulta do usuário associada com ele. Para cada palavra em um artigo científico dado como entrada, os nós que têm as palavras de consulta e que existem no artigo de entrada estão aumentando o seu "nível de energia". Este processo continua até que a rede neural se estabiliza. A partir disso, pode-se ver que nós temos níveis de energia mais elevados para esse documento e, portanto, mais relevantes para a consulta. No entanto, também este tipo de abordagem tem problemas. Redes Neurais são geralmente lentos. Desta forma, a manipulação de uma rede neural com 15.000 palavras, o tamanho médio de um único artigo científico, ou 700 palavras distintas já seria muito lento, considerando que você teria que criar uma rede neural cada vez que um usuário faz uma consulta, multiplicando-o para a quantidade de documentos onde o usuário deseja pesquisar. (VENTURA; SILVA, 2008)

Já em uma linha diferente do que os métodos anteriores, temos a baseada abordagens estatísticas. As principais vantagens desses tipos de abordagens são a rápida

⁴Ontologias são utilizadas em inteligência artificial, web semântica, engenharia de software e arquitetura da informação, como uma forma de representação de conhecimento sobre o mundo ou alguma parte deste.

implementação e da independência em relação à língua usado nos textos, em relação à estrutura utilizada e ao contexto dos documentos testados. Alguns dos mais conhecidos métodos estatísticos para a recuperação da informação são o critério de frequência de Luhn, método Tf-Idf e o método Zhou Slater. Mais adiante o método escolhido, Tf-Idf, para a resolução do nosso problema será detalhado.

4.5 COMPLEXIDADE DO TEXTO

Nesta seção descrevemos algumas características que foram consideradas relevantes para o estudo da complexidade de um texto, baseado em uma pesquisa realizada na Universidade de Amsterdã (TUMP, 2014).

Dificuldade das palavras Existem dois fatores que são importantes para determinar a dificuldade de uma palavra - considerar o nível dos indivíduos que escrevem o texto e o nível do leitor ou a finalidade proposta. Na pesquisa realizada a maioria dos participantes (67%) da avaliação reportaram que a especificidade da palavra mencionada é a principal característica para avaliar a complexidade de um texto. Os restantes participantes citaram o comprimento das palavras como uma grande influência da dificuldade das palavras. Antes de considerar o valor dos recursos abaixo, apenas as palavras significativas, ou palavras-chave, devem ser usadas. Isto é feito para evitar a influência de palavras muito frequentes, como "o" e "e", o que poderia alterar o valor dos recursos. Um corpus de stopwords do NLTK pode ser usado, eliminando as palavras de alta frequência, como 'a', 'a', 'e', 'também' e 'que', que normalmente têm pouco conteúdo lexical e sua presença em um texto não contribui para distinguir um texto de outros textos (BIRD et al., 2009).

Especificidade A especificidade da palavra é determinada através do cálculo da profundidade da árvore de hiperonímia ⁵. A profundidade média de todas as possíveis árvores de hiperonímia da palavra é calculada e foi considerada como o valor especificidade por palavra no texto. Para calcular este valor para todo o texto, o valor médio da especificidade de todas as palavras-chave é calculado.

Comprimento das palavras O comprimento médio das palavras de um texto é um fator que pode ser considerado importante para analisar a complexidade.

⁵Relação estabelecida entre um vocábulo de sentido mais genérico e outro de sentido mais específico (p.ex., animal está numa relação de hiperonímia com leão, gato etc.).

Comprimento das sentenças O comprimento médio das frases de um texto é determinado pelo cálculo da média da quantidade de palavras por frase.

Número de entidades nomeadas O valor desse recurso é determinado pelo cálculo do número médio de entidades nomeadas por sentença em um texto. O uso de referências a entidades pode ser considerado um fator muito importante para esta análise.

Comprimento do Parágrafo O número médio de sentenças por parágrafo é usado como valor deste recurso. Esta é a medida do número médio de períodos, em vez do número médio de palavras.

A estrutura da sentença A estratégia inicial para determinar um valor para a estrutura da frase é calculando a profundidade média da árvore de parsing. Esta árvore de parsing mostra a estrutura dos sintagmas das frases em relação uns aos outros e, por conseguinte, faz com que uma frase complexa tenha uma profundidade maior do que a árvore de frases relativamente simples. O analisador sintático PALAVRAS, citado anteriormente, pode ser utilizado para esta tarefa.

5 MODELO DE PROCESSAMENTO DO TEXTO

A seguir serão descritas as soluções propostas para a resolução dos problemas considerados no capítulo anterior, considerando o problema em questão - a análise das sugestões sobre a antiga Constituição do Brasil. Baseado no estudo realizado sobre os problemas considerados no capítulo anterior, foram escolhidos alguns métodos e algoritmos já implementados por outros pesquisadores, afim de buscar uma solução aplicável ao problema cenário deste trabalho.

As aplicações dos algoritmos e os resultados obtidos, assim como a descrição da base de dados, serão abordados no capítulo seguinte.

5.1 SEQUENCE MATCHER

Uma classe/biblioteca python

class difflib.SequenceMatcher: Esta é uma classe flexível para comparar pares de sequências de qualquer tipo. O algoritmo que serviu de base, e é um pouco mais sofisticado do que este, é um algoritmo que foi publicado no final dos anos 1980 por Ratcliff e Obershelp sob o nome "gestalt pattern matching." A idéia é encontrar a maior subsequência de correspondência contígua que não contenham elementos "lixo" (Ratcliff e Obershelp não consideraram a abordagem dos "lixos"). A mesma ideia é então aplicado de forma recursiva para as todas as sequências para a esquerda e para a direita da subsequência correspondente.¹

Complexidade: O algoritmo básico Ratcliff-Obershelp tem complexidade de ordem cúbica, no pior caso e complexidade de ordem quadrática no caso esperado. O Sequence-Matcher tem complexidade quadrática para o pior caso e no caso esperado, o comportamento dependente de forma complexa de quantos elementos as sequências têm em comum; o melhor caso é tem complexidade linear.

¹7.4. difflib — Helpers for computing deltas (<https://docs.python.org/2/library/difflib.html>)

Automatic junk heuristic: SequenceMatcher suporta uma heurística que trata automaticamente certos termos de uma sequência como "lixo". A heurística conta quantas vezes cada termo aparece individualmente na sequência. Se a duplicata de um termo (após o primeiro) são responsáveis por mais de 1 % da sequência e a sequência tem, pelo menos, 200 itens de comprimento, este item é marcado como "popular" e é tratado como lixo para fins de correspondência entre as sequências. Essa heurística pode ser desativada, definindo o argumento "autojunk" para "Falso" ao chamar o SequenceMatcher.

5.2 TERM FREQUENCY – INVERSE DOCUMENT FREQUENCY

Tf-Idf Term Frequency - Inverse Document Frequency (SALTON; BUCKLEY, 1988), é uma métrica para calcular a relevância de termos em documentos, muito usado em Recuperação de Informação e Text-Mining. Essencialmente, esta técnica mede o quão importante uma determinada palavra está em um documento a respeito de outros documentos da mesma coleção. Basicamente, uma palavra se torna mais importante num certo documento quanto mais ocorre no referido documento. Mas se essa palavra ocorre em outros documentos, sua importância diminui. As palavras que são muito frequentes em um único documento tendem a ser mais valorizadas do que as palavras mais comuns em outros documentos, como artigos ou preposições. O procedimento formal para a implementação de Tf-Idf muda ligeiramente de aplicação para aplicação, mas a abordagem mais comum foi a utilizada neste trabalho.

Geralmente, o cálculo do Tf-IDF é feito em separadamente. É composto por dois termos: um primeiro calcula a frequência do termo normalizado, que é o número de vezes que uma palavra aparece em um documento, dividido pelo número total de palavras contidas nesse documento (essa contagem é normalizada para evitar que a palavra, em documentos muito longos, tenham valores de Tf mais altos). Em seguida, o segundo termo é o *Document Frequency Inverse*, que é calculado como o log do número dos documentos do corpus, dividido pelo número de documentos onde o termo aparece. A equação a seguir mede a probabilidade de que um termo i ocorra em um documento j :

$$Tf_{ij} = \frac{n_{i,j}}{n_j} \quad (5)$$

onde $n_{i,j}$ é o número de vezes em que o termo i aparece no documento j e n_j é o número de palavras no documento j .

O componente Idf mede a relevância geral do termo no corpus e pode ser calculado de acordo com a seguinte equação:

$$Idf_i = \log \left(\frac{|D|}{|\{d_j : t_i \in d_j\}|} \right) \quad (6)$$

onde $|D|$ é a quantidade de documentos no corpus e $|\{d_j : t_i \in d_j\}|$ é a quantidade de documentos em que o termo t_i aparece.

O Tf-IDF é então calculado multiplicando-se as duas equações anteriores:

$$TfIdf_{i,j} = Tf_{i,j} * Idf_i \quad (7)$$

6 APLICAÇÃO DO MODELO

6.1 DATA SET

O conjunto de dados que estamos a trabalhar para este fim é um banco de dados sugerido na antiga constituição brasileira, escrito por cidadãos civis brasileiros. A base é composta de uma série de metadados e um texto escrito em Português, com a sugestão. Este conjunto de dados contém um grande número de sugestões (780.000), e para efeitos da presente proposta inicial, iremos utilizar um subconjunto de 1.000 sugestões. Abaixo está um exemplo com o formato de texto, neste caso, esta é a sugestão do índice 248.

```
1 | #from 'Sugestao-1-9999.tsv'
2 | df = pd.DataFrame().from_csv('Sugestao-1-9999.tsv', sep='\t')
3 | #indexed for "COD_DOCUMENTO"
4 | df = df.set_index("COD_DOCUMENTO")
5 | df = df.dropna(subset=['SUGESTAO'])
6 | df.SUGESTAO[248]:
```

"As minhas sugestões são as seguintes: Como prioridade a agricultura, sendo que, é preciso ajudar os pequenos e médios agricultores dando-lhes condições para que possam viver nas pequenas e médias propriedades, dando-lhes financiamentos à baixo custo, para que possam adquirir máquinas e emprementos que são necessários para o cultivo de suas terras. Referente as pequenas empresas precisamos de mais condições com financiamentos com juros mais baixos e menos burocracia. Refente salários, devemos elevá-los que esses aumentos sejam repassados ao próprio assalariado."

6.2 OBJETIVO DA APLICAÇÃO

O objetivo desta aplicação é executar os algoritmos propostos no capítulo anterior, buscando gerar resultados que possam nos auxiliar a inferir a qualidade das sugestões. A análise da qualidade das sugestões considera como parâmetros a complexidade da escrita dos textos, baseado no estudo realizado neste trabalho. A detecção de cópias, a inferência de palavras importantes nos textos e o particionamento sintático através do PALAVRAS,

são apenas para mostrar os resultados da aplicação do estudo a esta base, com o intuito de mostrar a relevância dos dados obtidos e também, futuramente, dar continuidade a este trabalho, em outro escopo, agregando a implementação de algoritmos mais complexos como os de reconhecimento de entidades nomeadas e complexidade do texto.

6.3 DETECÇÃO DE CÓPIAS - SEQUENCEMATCHER

6.3.1 ÍNDICE DE SIMILARIDADE

Os textos foram comparados dois a dois, e então foi gerado um índice de similaridade para o par. O interesse de se gerar este índice é indicar o quão possivelmente um texto é cópia do outro. Como já foi descrito no capítulo anterior, utilizamos uma biblioteca em Python para a resolução.

```
1 G = nx.Graph() #the result will be stored in a graph
2 for par in itertools.combinations(df.index[:1000], 2):
3     r = SequenceMatcher(None, df.SUGESTAO[par[0]], df.
        SUGESTAO[par[1]]).ratio()
4     if r > 0.65:
5         G.add_edge(par[0], par[1], weight=r)
```

Para esta análise foram considerados que nos casos em que o índice de similaridade (r) é maior que 0.65, as sugestões são consideradas cópias. Os resultado foram armazenados em um grafo G em que os nós são os índices dos textos, e o peso das arestas é o índice.

```
1 for line in nx.generate_edgelist(G):
2     print(line)
3
4 225.0 253.0 {'weight': 0.8653846153846154}
5 292.0 248.0 {'weight': 0.9391602399314481}
6 261.0 211.0 {'weight': 1.0}
7 261.0 212.0 {'weight': 1.0}
8 261.0 213.0 {'weight': 1.0}
9 ...
```

Este exemplo dos resultados obtidos, mostra que os textos de índice 225 e 253 não são idênticos, porém muito similares. Assim como o par 292/248. Mas o par 261/211 é, pelo algoritmo, textos idênticos. A seguir vamos analisar as sugestões com este índices a fim de checar a qualidade do resultado obtido através deste algoritmo.

```
1 df.SUGESTAO[225]
```

'Que dentro da constituinte, houvesse uma comparação salarial para o funcionalismo público municipal estatutário, como ocorre com o vereador municipal através decreto-lei federal n. 049 de dez/85.'

1 | df . SUGESTAO [253]

' Eu sugiro o seguinte: "Que dentro da constituinte houvesse uma comparação salarial para o funcionário público municipal estatutário, como ocorre com o vereador municipal através do decreto-lei no.049 de dezembro dez/85.

O par (225/253) foi classificado com índice de similaridade 0.8653 e os trechos em vermelho são as subsequências em comum entre os textos.

1 | df . SUGESTAO [292]

'- As minhas sugestões são as seguinte: - Como prioridade a agricultura, sendo que, é preciso ajudar os pequenos e médios agricultores, dando-lhes condições para que possam viver nas pequenas e médias propriedades, dando-lhes financiamentos à baixo custo, para que possam adquirir máquinas e empreendimentos agrícolas, que são necessários para o cultivo de suas terras. - Referente as pequenas empresas, precisamos de mais condições com financiamentos com juros mais baixos e menos burocracias. -Referente a salários, devemos elevá-los sem que estes aumentos sejam repassados ao próprio assalariado.'

1 | df . SUGESTAO [248]

' As minhas sugestões são as seguintes: Como prioridade a agricultura, sendo que, é preciso ajudar os pequenos e médios agricultores dando-lhes condições para que possam viver nas pequenas e médias propriedades, dando-lhes financiamentos à baixo custo, para que possam adquirir máquinas e emprements que são necessarios para o cultivo de suas terras. Referente as pequenas empresas precisamos de mais condições com financiamentos com juros mais baixos e menos burocracia. Refente salários, devemos elevá-los que esses aumentos sejam repassados ao próprio assalariado.'

O par (292/2248) foi classificado com índice de similaridade 0.9391 e os trechos em vermelho são as subsequências em comum entre os textos.

1 | df . SUGESTAO [261]

' Eleições Municipais 86 diretas já.'

1 | df . SUGESTAO [211]

' Eleições Municipais 86 diretas já.'

O par (261/211) foi classificado com índice de similaridade 1.0 e os trechos em vermelho são as subsequências em comum entre os textos.

6.3.2 QUE VALOR DE R CONSIDERAR PARA DOIS TEXTOS COMO CÓPIAS?

Uma questão importante e que deve ser considerada futuramente é como escolher o valor de "r" para considerar dois textos como cópias. Neste estudo inicial, empiricamente escolhemos o valor $r > 0,65$. O resultado foi muito consistente. Até que valor pode diminuir o "r" para considerar dois textos como cópias?

6.3.3 O GRAFO

Como foi referido anteriormente, os resultados foram armazenados em um grafo. Os nós são os IDs dos arquivos, ou seja, a numeração que identifica cada sugestão. As arestas representam a comparação entre dois nós e tem pesos em conformidade com o coeficiente de similaridade. A representação deste gráfico foi usado para clusterizar textos que são possivelmente cópias um do outro, agrupados por cor. A visualização é interativa e pode ser acessada através do link constituicao.cxseg.com (a senha é "emap123 "). Os nós são acessíveis e quando clicados explicitam os textos das sugestões, como na figura que se segue..

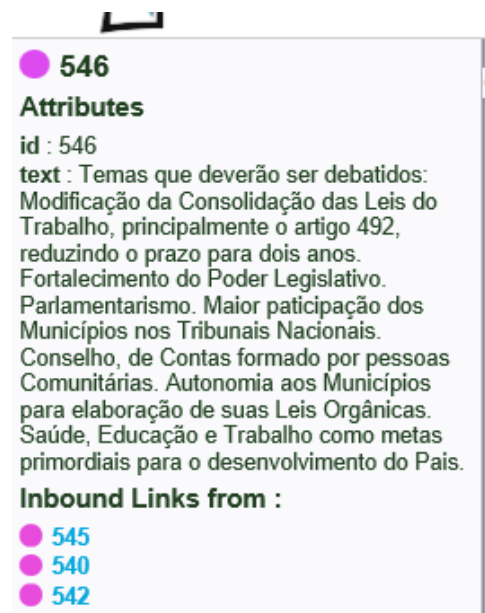


Figura 2: Graph attributes

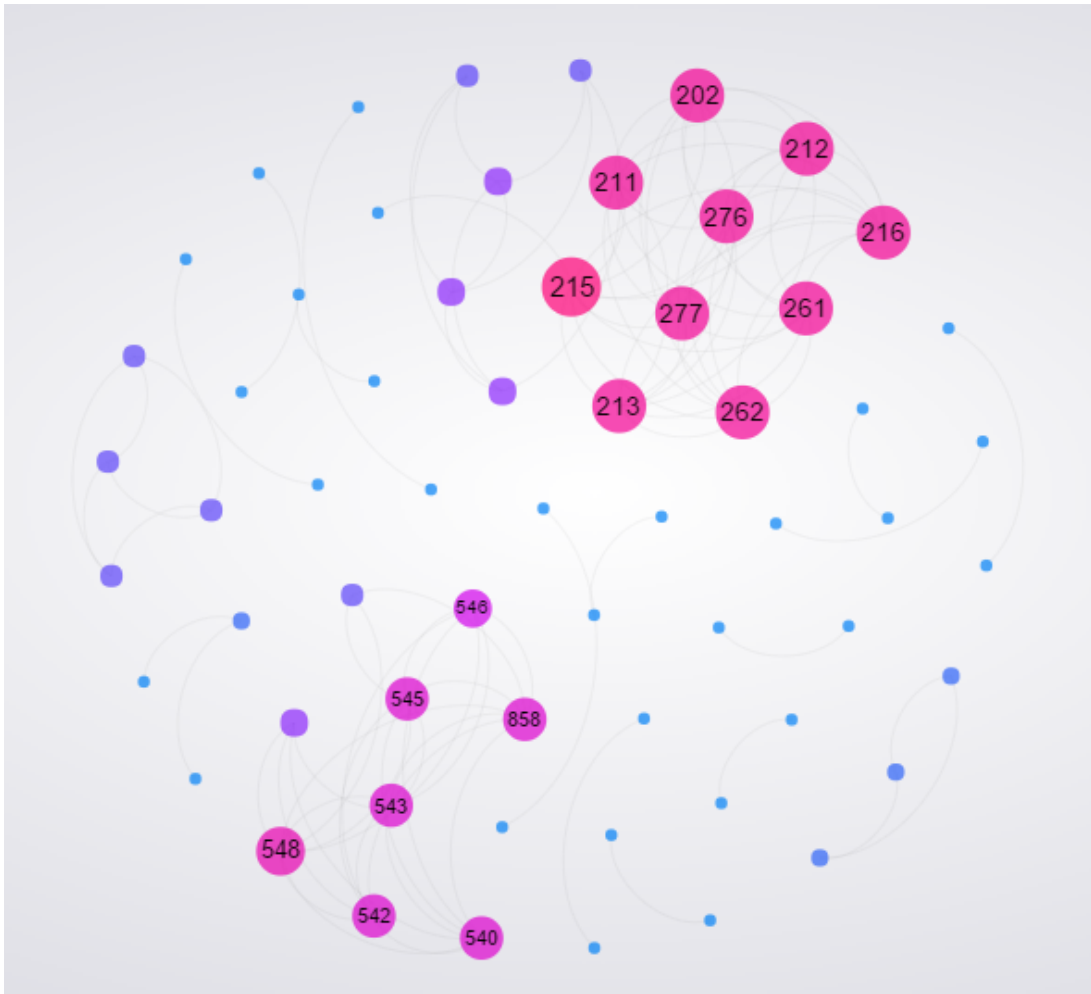


Figura 3: The graph

O custo computacional para executar este algoritmo é muito alto. Neste estudo inicial só foi possível usando 1000 sugestões, o que é uma quantidade muito limitada do conjunto de dados total. Para o propósito deste estudo, este é o melhor algoritmo, e pode ser adaptado para se otimizar o custo computacional, mas ainda assim continua a ser muito custoso. Será necessário um conjunto mais avançado do hardware para a continuidade do estudo. Sobre a escolha do valor ideal de "r" para detecção de cópias, ainda são necessários mais debates para chegar a este valor, quer empiricamente ou matematicamente.

6.4 ANALISADOR SINTÁTICO PALAVRAS

Os resultados gerados no analisador sintático PALAVRAS são representados pelas visualizações do particionamento dos textos, de acordo com as variações. A plataforma online permite o input dos dados e a escolha dos parâmetros para o parsing, permitindo assim gerar "outputs" de variadas formas. A seguir alguns exemplos dos resultados ob-

tidos com o algoritmo PALAVRAS e que serão úteis para o aprofundamento do estudo, em outra oportunidade, como por exemplo, para auxílio na análise da complexidade dos textos.

Flat structure Estruturas geradas para análises gramaticais em modos não-árvore

Exemplo:

”Referente as pequenas empresas precisamos de mais condições com financiamentos com juros mais baixos e menos burocracia.”

```
.
referente [referente] <ac> N M S @SUBJ>
as [o] <artd> DET F P @>N
pequenas [pequeno] ADJ F P @>N
empresas [empresa] <HH> N F P @PRED>
precisamos [precisar] <fmc> V PR 1P IND VFIN @FMV
de [de] PRP @<PIV
mais [muito] <quant> <KOMP> DET F P @>N
condições [condição] <sem-c> <sit> <state> N F P @P<
com [com] PRP @N<
financiamentos [financiamento] <act> N M P @P<
com [com] PRP @N<
juros [juro] <mon> N M P @P<
mais [muito] <quant> <KOMP> ADV @>A
baixos [baixo] ADJ M P @N<
e [e] KC @CO
menos [pouco] <quant> <KOMP> DET F S @>N
burocracia [burocracia] <HH> N F S @P<
```

Figura 4: Análise morfo-sintática completa

Tree structure Estruturas geradas para análises gramaticais em modos árvore

```

- S:n("referente" M S)  referente
- A:g(np)
  | -D:nil(pron det "o" &lt;artd&gt; DET F P)  as
  | -D:adj("pequeno" F P)  pequenas
  | -H:n("empresa" F P)  empresas
  | -P:v(fin "precisar" &lt;fmc&gt; PR 1P IND VFIN)  precisamos
- A:g(pp)
  | -H:nil(prp "de" &lt;right&gt;)  de
  | -D:g(np)
    | -D:nil(pron det "muito" &lt;quant&gt; &lt;KOMP&gt; DET F P)  mais
    | -H:n("condição" F P)  condições
    | -D:g(pp)
      | -H:nil(prp "com" &lt;np-close&gt;)  com
      | -D:g(np)
        | -H:n("financiamento" M P)  financiamentos
        | -D:g(pp)
          | -H:nil(prp "com" &lt;np-close&gt;)  com
          | -D:g(np)
            | -H:n("juro" M P)  juros
            | -D:adjp
              | -D:adv("muito" &lt;quant&gt; &lt;KOMP&gt; &lt;fr:24&gt;)  mais
              | -H:adj("baixo" M P)  baixos
            | -&lt;-&gt;:nil(conj "e" &lt;f:2349620&gt;)  e
            | -D:g(np)
              | -D:nil(pron det "pouco" &lt;quant&gt; &lt;KOMP&gt; DET F S)  menos
              | -H:n("burocracia" F S)  burocracia

```

Figura 5: Árvore - Análise morfo-sintática completa

Dependency structure Estruturas de dependência geradas para análises gramaticais em modo árvore.

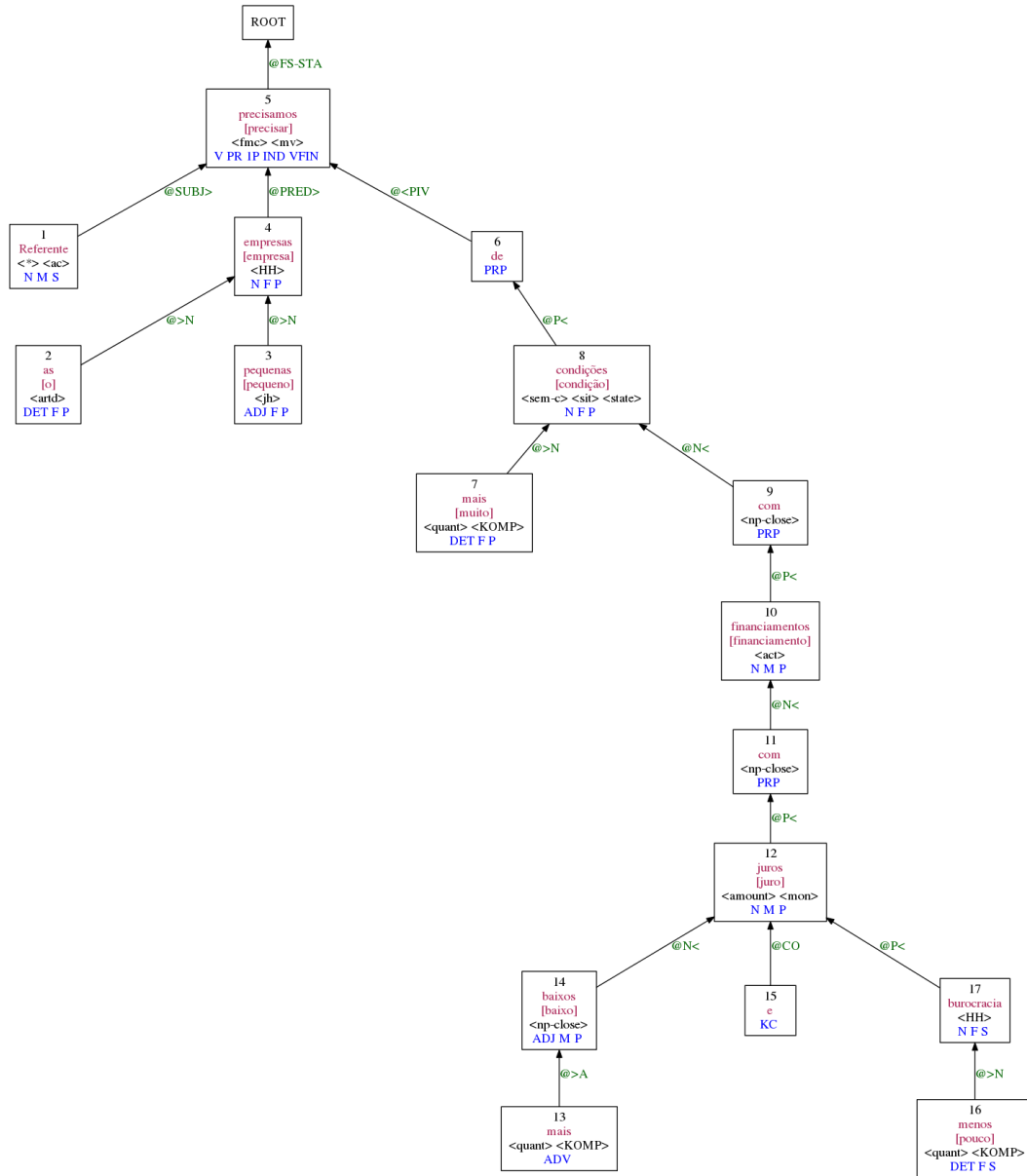


Figura 6: Árvore - Estrutura de dependência

6.5 IMPORTÂNCIA DAS PALAVRAS NO TEXTO - TF-IDF

Para esta parte do trabalho, utilizamos um algoritmo já estudado e proposto, na linguagem Python, utilizando a biblioteca NLTK.

Para calcular os pesos TF-IDF para cada documento do corpus, foram necessárias três etapas:

- Tokenizar o corpus
- Modelar o Space Vector
- Calcular o peso TF-IDF para cada documento no corpus

O primeiro passo é tokenizar o texto. Para fazer isso, podemos usar a biblioteca NLTK que é uma coleção de algoritmos de processamento de linguagem natural escritos em Python. O processo de tokenizar os documentos no corpus se dá em dois passos: primeiro o texto é dividido em frases, em seguida, as sentenças são divididas em palavras individuais. É importante notar que há várias palavras que não são relevantes, ou seja, termos como "a, é, em, em", etc, como já vimos anteriormente no capítulo 4, e desta forma precisamos ignorar estas "stopwords".

Depois disso o algoritmo vai tokenizar todos os documentos e representá-los como vetores (linhas) de uma matriz $D \times F$, onde D é quantos documentos temos e o F é o número de palavras.

Agora que cada um dos documentos no corpus foi indexado, o próximo passo é calcular a frequência de cada termo no documento. Como já foi descrito anteriormente, é importante normalizar estas frequências. Dado que temos um termo muito repetido no documento, com a intenção não criar um viés em documentos longos, fazendo-os parecer mais importante do que eles são apenas por causa da sua alta frequência do termo no documento, é necessário esta normalização.

Depois de normalizar, o algoritmo terá que calcular o segundo termo do tf-idf: a frequência inversa do documento, que matematicamente já foi descrita anteriormente.

O código em Python, desenvolvido por Marcel Caraciolo, e que serviu de base para este estudo pode ser visto na seção "Apêndices".

A seguir um exemplo dos termos mais relevantes identificados através deste algoritmo para uma porção da base:

```

1 ...
2
3 for tip in df.SUGESTAO[:100]:
4 .
5 .
6 .
7 for item in sorted(words.items(), key=lambda x: x[1], reverse=
    True):
8     print("%f <= %s" % (item[1], item[0]))
9
10 1.304008 <= reajuste trimestral
11 1.304008 <= trimestral
12 1.168853 <= reajuste
13 0.391202 <= trimestralidade
14 0.391202 <= o funcionalismo publico
15 ...

```

No exemplo acima, que foi aplicado a apenas 100 sugestões da base, os cinco termos considerados mais relevantes foram :”reajuste trimestral”, ”trimestral”, ”reajuste”, ”trimestralidade” e ”o funcionalismo publico”. Considerando esta pequena amostra, é possível acreditar que os resultados obtidos são coerentes, pois já esperávamos que estes fossem os termos mais importantes desta porção de sugestões.

7 CONCLUSÃO

Este trabalho permitiu a exploração e estudo sobre algumas teorias, aplicações e ferramentas da extensa área de Processamento de Linguagem Natural. Os recursos estudados e que possibilitaram aplicação ao problema real utilizado como cenário para este trabalho, foram de fundamental importância para o entendimento do potencial deste ferramental. Alguns recursos como o Reconhecimento de entidades nomeadas e a metodologia para análise de complexidade do texto, devido suas características mais específicas não permitiram um estudo mais profundo e aplicação destes recursos na base utilizada, não se encaixou no escopo deste trabalho.

Os resultados obtidos com a aplicação dos algoritmos estudados de detecção de cópias, inferência de estruturas textuais importantes e análise morfo-sintática através da plataforma PALAVRAS foram de grande valia para a compreensão dos estudos realizados sobre esta temática bem como serão muito úteis para a continuidade desta pesquisa, que sem dúvidas tende a alcançar objetivos mais audaciosos no decorrer do desenvolvimento do projeto.

O objetivo deste trabalho, que era estudar e compreender recursos de PLN que compõe uma Metodologia para análise de qualidade em textos, além de aplicar a pesquisa realizada a uma base de dados textuais com informações reais e gerar resultados importantes para contribuir com o desenvolvimento da pesquisa de doutorado do pesquisador Alex Hudson (University of Texas), foi alcançado de forma satisfatória. Além disso, o estudo serviu de estímulo para a continuidade da pesquisa nesta área e, em particular, na obtenção de recursos e ferramental para a resolução do problema em questão, no decorrer dos próximos anos.

REFERÊNCIAS

ALI, A. M. E. T.; ABDULLA, H. M. D.; SNÁSEL, V. Overview and comparison of plagiarism detection tools. In: CITESEER. **DATESO**. [S.l.], 2011. p. 161–172.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python**. [S.l.]: "O'Reilly Media, Inc.", 2009.

BRILL, E. A simple rule-based part of speech tagger. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the workshop on Speech and Natural Language**. [S.l.], 1992. p. 112–116.

DAS, A. et al. Neural net model for featured word extraction. In: **Unifying Themes in Complex Systems IV**. [S.l.]: Springer, 2008. p. 353–361.

DUARTE, J.; MILIDIÚ, R. Machine learning algorithms for portuguese named entity recognition. **Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio**, 2007.

GALASSO, J. **Analyzing English Grammar: An Introduction to Feature Theory**. CSUN, 2000. Table 12: Functional Categories. Disponível em: <<https://www.csun.edu/galasso/completehandbook.htm>>.

RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. **Proceedings of the IEEE**, IEEE, v. 77, n. 2, p. 257–286, 1989.

SALTON, G.; BUCKLEY, C. Term-weighting approaches in automatic text retrieval. **Information processing & management**, Elsevier, v. 24, n. 5, p. 513–523, 1988.

SOUZA, R. R. Uma proposta de metodologia para escolha automática de descritores utilizando sintagmas nominais. **Perspectivas em Ciência da Informação**, v. 10, n. 2, 2008.

TUMP, D. Text complexity analysis of news articles: Forming an objective view of news events by introducing the nack system. **University of Amsterdam**, Institute for Language and Logic Faculty of Science, 2014.

VENTURA, J.; SILVA, J. F. da. **Ranking and extraction of relevant single words in text**. [S.l.]: INTECH Open Access Publisher, 2008.

APÊNDICE A – PYTHON CODE AND RESULT: SEQUENCEMATCHER

```
1 import difflib
2 from difflib import SequenceMatcher
3 import pandas as pd
4 import itertools
5 import numpy as np
6 import networkx as nx
7 import matplotlib.pyplot as plt
8 import urllib.request as urllib2, simplejson
9
10 df = pd.DataFrame().from_csv('Sugestao-1-9999.tsv', sep='\t')
11 df = df.set_index("COD_DOCUMENTO")
12 df = df.dropna(subset=['SUGEST 0'])
13
14 G = nx.Graph()
15
16 for par in itertools.combinations(df.index[:1000], 2):
17     r = SequenceMatcher(None, df.SUGEST 0[par[0]], df.
18         SUGEST 0[par[1]]).ratio()
19     if r > 0.65:
20         G.add_edge(par[0], par[1], weight=r)
21
22 for line in nx.generate_edgelist(G):
23     print(line)
```

APÊNDICE A – CÓDIGO PYTHON: TF-IDF

```

1  #Compute the frequency for each term.
2  vocabulary = []
3  docs = {}
4  all_tips = []
5  for tip in df.SUGESTO[:100]:
6      tokens = tokenizer.tokenize(tip)
7
8      bi_tokens = bigrams(tokens)
9      tri_tokens = trigrams(tokens)
10     tokens = [token.lower() for token in tokens if len(token) >
11                2]
12     tokens = [token for token in tokens if token not in
13               stopwords]
14
15     bi_tokens = [' '.join(token).lower() for token in bi_tokens]
16     bi_tokens = [token for token in bi_tokens if token not in
17                  stopwords]
18
19     tri_tokens = [' '.join(token).lower() for token in
20                  tri_tokens]
21     tri_tokens = [token for token in tri_tokens if token not in
22                   stopwords]
23
24     final_tokens = []
25     final_tokens.extend(tokens)
26     final_tokens.extend(bi_tokens)
27     final_tokens.extend(tri_tokens)
28     docs[tip] = {'freq': {}, 'tf': {}, 'idf': {},
29                 'tf-idf': {}, 'tokens': []}
30
31     for token in final_tokens:
32         #The frequency computed for each tip
33         docs[tip]['freq'][token] = freq(token, final_tokens)
34         #The term-frequency (Normalized Frequency)
35         docs[tip]['tf'][token] = tf(token, final_tokens)
36         docs[tip]['tokens'] = final_tokens
37
38     vocabulary.append(final_tokens)
39
40 for doc in docs:

```

```

36     for token in docs[doc]['tf']:
37         #The Inverse-Document-Frequency
38         docs[doc]['idf'][token] = idf(token, vocabulary)
39         #The tf-idf
40         docs[doc]['tf-idf'][token] = tf_idf(token, docs[doc]['tokens'], vocabulary)
41
42 #Now let's find out the most relevant words by tf-idf.
43 words = {}
44 n = 0
45 for doc in docs:
46     sum = 0
47     n += 1
48     for token in docs[doc]['tf-idf']:
49         if token not in words:
50             words[token] = docs[doc]['tf-idf'][token]
51         else:
52             if docs[doc]['tf-idf'][token] > words[token]:
53                 words[token] = docs[doc]['tf-idf'][token]
54
55     #print(doc)
56     for token in docs[doc]['tf-idf']:
57         sum += docs[doc]['tf-idf'][token]
58         #print(token, docs[doc]['tf-idf'][token])
59         #print("Ratio de %i = %f" %(n, (sum/len(docs))))
60 for item in sorted(words.items(), key=lambda x: x[1], reverse=True):
61     print("%f <= %s" % (item[1], item[0]))

```