PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESAMBIGUAÇÃO DE ANOTAÇÕES MORFOSINTÁTICAS FEITAS POR MTMDD

Área de Processamento de Linguagem Natural

por

Pablo Frederico Oliveira Thiele

Paulo Fernandes, Dr.
Orientador

Lucelene Lopes, Dra.
Co-Orientadora

Dissertação de Mestrado

Porto Alegre
2015

RESUMO

Atualmente as tecnologias de Processamento de Linguagem Natural (PLN) estão sendo utilizadas em análises de enormes quantidades de dados. Com o advento das novas mídias e a adoção em massa das redes sociais, o fluxo de informações geradas a cada segundo é o maior da história. Embora isso se concentre, em maior parte, por informações e arquivos de multimídia, uma grande parcela da informação produzida, principalmente nas redes sociais, é textual. Desta forma, as soluções de PLN necessitam ser mais robustas do que jamais foram, encontrando soluções de processamento que possam acompanhar esta geração constante de informações ou pelo menos apresentar resultados melhores se comparados aos procedimentos utilizados anteriormente.

Os etiquetadores ou taggers são um dos principais componentes da PLN. Sua função, explorada neste trabalho é a capacidade de observar e catalogar as palavras em um texto de acordo com suas funções morfossintáticas. O nome comumente dado a este processo é o de POST (Part-Of-Speech Tagging). Dentro do contexto Part-Of-Speech (POS) encontra-se a função de processar e identificar um grupo de palavras agrupando-as em tipos pré-definidos. Este agrupamento pode ocorrer em razão sintática, morfológica ou morfossintática. Embora a velocidade de processamento seja uma caraterística digna de nota, quando tratamos de etiquetadores, a acuidade obtida por seu processo deve ser a premissa.

O conceito da obtenção de etiquetas semânticas a partir de avaliações dos textos embora pareça simples em um primeiro momento, apresenta vários desafios. Um dos maiores desafios encontrado em PLN é o problema da ambiguidade. Esta situação que ocorre nas mais diversas etapas do processamento de linguagem natural é complexa, devido à necessidade de que a aplicação processadora tenha conhecimentos abrangentes que possam ser utilizados como ferramentas que colaborem no intuito de realizar as escolhas mais corretas. Devido ao fato de se tratar de um problema antigo, inerente à linguagem natural e existente desde o começo das pesquisas da área, diversas possibilidades de minimizar suas consequências foram propostas. O presente trabalho enumera algumas das propostas encontradas, adicionando a possibilidade de uso de estruturas do tipo MTMDD no processo, buscando um ganho substancial de desempenho.

Palavras Chave: Processamento de Linguagem Natural, etiquetadores, ambiguidade, MTMDD.

ABSTRACT

The Natural Language Processing technologies (PLN) are being used for analysis of huge amounts of data. With the advent of new media and mass adoption of social networking, the flow of information generated every second is the largest in history. The majority of that is multimedia files. Meanwhile, a large portion of the information produced, especially in social network, is textual. Thus, PLN solutions need to be more robust than they ever were, finding processing solutions that might accompany this constant information production or at least provide better results compared to procedures previously used.

The labelers or taggers are a major component of PLN. Its function, explored in this work is the ability to observe and catalog the words in a text according to their morphosyntactic functions. The name commonly given to this process is the POST (Part-Of-Speech Tagging). Within the context Part-Of-Speech (POS) is the function to process and identify a group of words by grouping them into pre-defined types. This grouping can occur due to syntactic, morphological or morphosyntactic. Although the processing speed is a worthy feature, when we deal with labelers, the accuracy obtained for its process should be the premise.

The concept of obtaining semantic labels from texts evaluations seems simple at first sight, although presents several challenges. One of the major challenges encountered in PLN is the problem of ambiguity. This situation, which occurs in several stages of natural language processing, is complex due to requires comprehensive knowledge from the processing application using that as tools to collaborate in order to implement the most correct choices. It is a classic problem, inherent to natural and existing language since the beginning of the researches of this area. Several possibilities to minimize its consequences have been proposed since then. This paper lists some of the proposals found on the literature by adding the possibility to use MTMDD structures during the process, looking for a substantial performance gain.

Keywords: Natural Language Processing, taggers, ambiguity, MTMDD.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CG	Constraint Grammar
CTMC	Continuous Time Markov Chains
DTMC	Discrete Time Markov Chains
HMM	Hidden Markov Models
PLN	Processamento de Linguagem Natural
POS	Part-of-Speech
POST	Part-of-Speech Tagging
SVM	Support Vector Machines

SUMÁRIO

1 INTRODUÇÃO

Atualmente as tecnologias de Processamento de Linguagem Natural (PLN) estão sendo utilizadas em análises de enormes quantidades de dados. Com o advento das novas mídias e a adoção em massa das redes sociais, o fluxo de informações geradas a cada segundo é o maior da história. Segundo o SINTEF, 90% de todas as informações hoje armazenadas no planeta foram geradas nos últimos dois anos [1]. Embora isso se concentre, em maior parte, por informações e arquivos de multimídia, uma grande parcela da informação produzida, principalmente nas redes sociais, é textual. Desta forma, as soluções de PLN necessitam ser mais robustas do que jamais foram, encontrando soluções de processamento que possam acompanhar esta geração constante de informações ou pelo menos apresentar resultados melhores se comparados aos procedimentos utilizados anteriormente.

Tendo em mente a necessidade de velocidade de processamento necessário no caso de utilizarmos uma enorme quantidade de informação no procedimento, devemos também, manter a meta de obter a melhor taxa de acerto possível. Os etiquetadores ou *taggers* são um dos principais componentes da PLN, e como elemento crucial, antes de velocidade sua acuidade deve ser a premissa. Sua função, explorada nesse caso é a capacidade de observar e catalogar as palavras de um texto de acordo com suas funções morfossintáticas. O nome comumente dado a este processo é o de POST (Part-Of-Speech Tagging).

Dentro do contexto Part-Of-Speech (POS) encontra-se a função de processar e identificar um grupo de palavras agrupando-as em tipos pré-definidos. Este agrupamento pode ocorrer em razão sintática, morfológica ou morfossintática. Assim sendo, uma palavra pode ser definida como verbo, adjetivo ou pronome citando apenas algumas opções. Bem como, utilizando uma observação de contexto de sua frase, pode-se definir se uma determinada palavra atua como sujeito, objeto direto ou qualquer outra possibilidade adequada.

O conceito da obtenção de etiquetas semânticas a partir de avaliações dos textos embora pareça simples em um primeiro momento, apresenta vários desafios. Um dos maiores desafios encontrado em PLN é o problema da ambiguidade. Esta situação que ocorre nas mais diversas etapas do processamento de linguagem natural é complexa, devido à necessidade de que a aplicação processadora tenha conhecimentos abrangentes que possam ser utilizados como ferramentas que colaborem no intuito de realizar as escolhas mais corretas. Devido ao fato de se tratar de um problema antigo, inerente à linguagem natural e existente desde o começo das pesquisas da área, diversas possibilidades de minimizar suas consequências foram propostas. O presente trabalho

enumera algumas das propostas encontradas, adicionando a possibilidade de uso de estruturas do tipo MTMDD no processo, buscando um ganho substancial de desempenho.

1.1 Motivação

De acordo com Silva [2], a ambiguidade é uma das maiores dificuldades a serem devidamente administradas nos processos de linguagem natural. Suas peculiaridades recaem nas suas diversas formas de apresentação que podem ocorrer em cada etapa do processo. Suas variações podem ser do tipo que representa ambiguidade semântica, sendo esta mitigada com um conhecimento do mundo real do seu redor. Há também as ambiguidades de baixo nível, sendo estas exemplificadas como o reconhecimento correto do ponto final em uma frase, que pode ser tanto para marcar o final desta ou simplesmente indicar a presença de uma palavra que representa uma abreviatura.

Em um passo anterior à desambiguação, existe a necessidade das anotações POS das palavras de um texto de uma forma correta, essas etiquetas são obtidas após um pré-processamento do texto. O processo de etiquetagem que já foi em seus primórdios uma tarefa estritamente manual executada basicamente por especialistas do idioma do texto, hoje pode ser obtido com o processamento automatizado. As formas e objetivos dos atuais processamentos automáticos variam, alguns se utilizam de capacidades de aprendizagem de máquina para que após algumas fases de treino com grandes quantidades de texto, estes componentes que "aprendam" como os textos em um determinado domínio se comportam. Isso permite que esses programas possam ser mais efetivos quando a eles é apresentado um texto novo a fim de que se sejam anotadas todas as suas palavras.

Na literatura podem ser encontradas propostas de desambiguação que entremeiam os processos de etiquetação dos textos processados. Citando exemplos, temos os trabalhos Aduriz e Illarraza [3], Brill [4], Giménez e Màrquez [5] e Segond et.al.[7]. Basicamente cada proposta utiliza-se de um horizonte único, alguns trabalham com a noção de utilizar métodos estritamente probabilísticos no momento de definir as desambiguações adequadas. No entanto outros exemplos dão conta de utilizarem uma forma gramatical, criando regras específicas que são utilizadas como guias no momento da catalogação das palavras. Nos trabalhos mais modernos pode-se observar o uso de propostas híbridas que visam unir o melhor dos dois mundos fazendo-se valer das acuidades e velocidades das regras e utilizando as estatísticas para uma validação confirmadora. Os trabalhos citados utilizam diversos idiomas, e algumas das soluções apresentadas podem ser aplicadas na língua portuguesa.

Verifica-se também que nenhuma das propostas encontradas na literatura faz uso de anotações advindas de estruturas *Multi-Terminal Multi-valued Decision Diagrams* (MTMDD)

estruturas essas que permitem o uso de grande quantidade de dados de pesquisa, em formato de dicionários, por vezes multilíngue para a classificação de palavras de uma maneira extremamente rápida [6].

Com isto, este trabalho tem como motivação o fato de: (i) o problema de a ambiguidade ser um problema complexo e a proposta de novas técnicas para minimizar sua atuação é bem vinda; (ii) não existir propostas de desambiguação que utilizem estruturas MTMDD durante o *tagging* para a língua portuguesa; (iii) a possibilidade de se obter uma solução flexível, que por utilizar MTMDD, possa ser facilmente adaptada para novos idiomas, bastando que novos dicionários sejam criados, catalogados e adicionados à ferramenta.

1.2 Objetivos

O objetivo geral do trabalho é propor um processo de desambiguação de classes gramaticais encontradas nos textos em língua portuguesa anotados. Visando um processo iterativo que busca melhorar seu desempenho, a solução foi divida em objetivos específicos:

I. Programar um sistema computacional que execute a desambiguação de classes gramaticais em um texto previamente anotado, valendo-se de uma escolha aleatória.

II. Obter estatísticas de um *corpus* corretamente anotado e utilizar esses dados de maneira a desambiguar utilizando cálculos de probabilidade ponderada melhorando o desempenho em relação ao objetivo anterior.

III. Melhorar ainda mais os resultados utilizando o conhecimento de qual classe gramatical cada palavra vizinha possui em um *corpus* corretamente anotado. Este agrupamento em triplas permite a observação de palavras contíguas a fim de filtrar de maneira mais acurada as melhores opções de etiqueta do texto analisado.

Para que os objetivos traçados nesta proposta sejam concluídos, na seção seguinte será descrita a metodologia que será utilizada.

1.3 Metodologia

Observou-se nos trabalhos similares que já havia uma definição prévia a respeito de qual formato de *tagger* que seria o escolhido na proposta. Neste trabalho, no entanto, o enfoque está na constatação da evolução dos resultados obtidos nos protótipos criados em cada fase. Partindo de uma solução simples, aumentando sua complexidade ao longo do tempo com a expectativa de melhoria no desempenho, podemos ter o embasamento necessário de como construir um desambiguador desde o

começo. Esse tipo de abordagem nos permite uma ter uma visão clara e concreta, baseada em fatos, de quais serão as melhores escolhas para adicionar a este trabalho.

A primeira etapa do trabalho consistirá em criar ferramentas próprias para ajustar um conjunto de *corpus* que será utilizado para a avaliação. Dentre diversas opções existentes, este trabalho irá focar exclusivamente no Mac-Morpho[1]. Aproveitando-se a característica do Mac-Morpho de ser um corpus revisado com etiquetagem correta, a avaliação da qualidade do *parser* será medida através do seu comparativo com o texto previamente anotado. O intuito do processo de melhoria contínua é adicionar novas funcionalidades e escopo aos poucos, assim para cada novo processo criado um melhor desempenho do *tagger* é esperado. Idealmente a ferramenta deve chegar ao patamar de agregar funcionalidades híbridas sendo estas estatísticas e baseadas em regras. Deverá ser criada uma lista de necessidades relevantes, para que os *taggers* criados possam ser melhorados ao decorrer do processo de criação.

Como soluções funcionais, os *taggers* serão testados com os *corpora* previamente obtidos a fim de que se possa criar uma classificação destes a partir do desempenho obtido através dos testes. Essa análise é crucial para rever configurações, realizar ajustes e adequações pontuais.

Após os diversos testes e obtendo valores adequados para cada tipo de abordagem, realizaram-se novos ajustes e melhorias eventuais a fim de gerar uma proposta que consiga trazer uma desambiguação adequada para textos que foram previamente anotados através de dicionários MTMDD existentes.

1.4 Organização

A organização da dissertação apresenta-se da seguinte maneira:

- O Capítulo 2 apresenta uma revisão breve sobre conceitos fundamentais relacionados ao PLN, *Part-Of-Speech Tagging* além de apresentar formas diversas de etiquetagem e conceitos de desambiguação. No final deste capítulo é mostrado um conjunto de trabalhos similares e uma análise de seus resultados.

- O Capítulo 3 apresenta o processo proposto nessa dissertação, detalhando as etapas trabalho, que se iniciam na criação de um desambiguador trivial, melhorando-o na segunda parte, culminando em um terceiro ajuste que forma um desambiguador

[1] Mac-Morpho é um corpus formado a partir de nóticias em português brasileiro anotado com POS tags.

capaz de observar relevância das classes gramaticais baseado na localização da palavra na sentença e classe dos termos vizinhos.

- O Capítulo 4 demonstra dados obtidos nos experimentos bem como o processo de criação desses dados e os testes realizados. A metodologia de análise e comparação dos resultados obtidos nos *taggers* deste trabalho em comparação a outros também é descrita nesta seção.

- O Capítulo 5 encerra o trabalho com as conclusões obtidas no estudo bem como propostas para futuras melhorias da atual solução.

2 CENÁRIO E CONTEXTUALIZAÇÃO

Neste capítulo são apresentados conceitos fundamentais para a compreensão deste trabalho. Na seção 2.1 é apresentada a definição para *Part-Of-Speech Tagging*, sua importância e contribuição para a PLN. Na seção 2.2 são descritos alguns tipos de *tagging*, que auxiliam na construção automática ou semiautomática de textos anotados. Na seção 2.3 é apresentado o conceito de desambiguação propriamente dito e as formas já apresentadas para lidar com esse problema. Na seção 2.4 são apresentadas propostas de automatização de etiquetagem existentes, e suas diferenças. Por fim a seção 2.5 apresenta uma breve análise das propostas apresentadas.

2.1 Part-Of-Speech Tagging

O grande objetivo de uma pesquisa em PLN é analisar e entender a linguagem. Como tarefas intermediárias a este objetivo final, temos objetivos menores que não requerem um completo entendimento de uma linguagem para ser realizado. Uma tarefa que pode se encaixar nessa descrição é a ação de etiquetar as palavras de um texto, catalogando-as ou simplesmente *Part-Of-Speech Tagging* [8]. O termo *Part-Of-Speech* também pode ser entendido como a definição de classe gramatical. Assim, as palavras são anotadas de acordo com este conceito situado na morfologia que classifica cada palavra em um texto conforme sua distribuição sintática e morfológica. Essa classificação constitui na função executada por cada palavra em uma frase. Uma palavra anotada pode ser classificada como: substantivo, adjetivo, verbo, advérbio, preposição etc.

Na definição de Schmid [9], o *Part-Of-Speech Tagging* é a tarefa que consiste em determinar corretamente as constituintes de uma sequência de palavras. Neste prisma introduzem-se as dificuldades inerentes de palavras que podem agir de maneira distinta conforme o contexto empregado e também o complicado da ambiguidade típica que muitas palavras possuem. Um exemplo, em língua inglesa, que demostra as diversas facetas que podem ser observadas pela mesma palavra quando o contexto difere é apresentado a seguir. Utilizando-se a palavra "*back"* como exemplo, podemos entender as diferentes representações da mesma nas frases: "*They stabbed him in the back*", "*They will back the proposal*", "*The charity owes $400,000 in back taxes*", "*It's too late to put the genie back in the bottle*" onde constitui respectivamente um substantivo, um verbo no infinitivo, um adjetivo e por fim um advérbio.

Levando em consideração que a quantidade de *tags* assinaladas nos grupos de palavras embora possa flutuar bastante, de dúzias às centenas o mais comum é definir um grupo de etiquetas representativas que juntas somam algo entre 50 a 150 *tags*. Essa quantidade pode variar bastante de

acordo com o idioma a ser utilizado dependendo o quão rico o idioma em questão é de regras morfossintáticas [9].

Schmid [9] comenta que diversos métodos já foram aplicados no processamento de POS T*agging* ao longo dos anos. Entre outras opções, foram utilizadas por Church [10] Cutting et. al. [11], e Brants [12] soluções com base em *Hidden Markov Models (HMM)*. Na década de 90, Brill [13] usou *transformation-based-learning*, Daelemans et al [14] utilizou *memory-based learning*, enquanto Ratnaparkhi [15] se valeu de *maximum-entropy modelling*. Outras opções utilizadas foram as redes neurais de Benello et al. [16] e Nakamura et al. [17], bem como as árvores de decisão empregadas por Black [18], Màrquez e Padró [19]. Podem ser encontradas ainda soluções que empregam *support vector machines [5] e* propostas de regras de desambiguação escritas de forma manual [20, 21, 22]. Ainda existe a possibilidade de se utilizar analisadores estatísticos para essa análise, no entanto os *taggers* mais comuns costumam ser muito mais rápidos e tendem a ser a opção mais aprimorada.

A construção de um POS *tagger* eficiente, como pode se imaginar, permite que sejam trilhados os mais diversos caminhos. Diversos trabalhos feitos na área apresentam uma releitura de trabalhos anteriores, utilizando uma técnica já empregada e reconhecidamente eficiente, no entanto adicionando alguma peculiaridade que agregue desempenho ou capacidade de maior acerto. Dentro dessa gama de opções já citadas anteriormente a seção a seguir tem como objetivo apresentar os tipos de *tagging* que constituem em opções interessantes para serem adaptadas como ferramentas do presente trabalho. Consequentemente, a construção da solução final, se dará apenas com os resultados dos testes realizados com os diversos tipos de *taggers* avaliando suas peculiaridades e analisando seus prós e contras, encontrando por fim uma solução que carregue o melhor custo-benefício.

2.2 Tipos de Tagging

Entre os diversos trabalhos e propostas já apresentados na área, serão brevemente comentados exemplos de processadores POST que se baseiam em ideias distintas na hora de resolver o problema de desambiguação das palavras. O primeiro se utiliza de regras pré-definidas para fazer sua avaliação direta, o segundo utiliza estudos probabilísticos no momento de executar sua escolha pela melhor opção. Por último a forma que tem sido adotada por trabalhos recentes é a que se baseia em uma mescla dos dois tipos anteriores criando uma solução híbrida.

2.2.1 Part-Of-Speech Tagging Baseado em Regras

Jurafsky e Martin [24] mostram que as soluções mais antigas (década de 60 e 70) de etiquetagem de Part-Of-Speech que se baseavam no uso de regras executavam duas etapas distintas. No primeiro momento é usado um dicionário que é capaz de listar todas as possibilidades válidas encontradas para cada palavra no texto. Em uma segunda etapa acontece a desambiguação que se vale de uma grande lusa de regras escritas manualmente. Com a ajuda dessa lista o sistema consegue filtrar as possibilidades encontradas, mantendo listas de possibilidades cada vez menores até que a decisão por apenas uma etiqueta aconteça e a desambiguação termine.

O comportamento encontrado nas soluções mais modernas não difere muito de seus predecessores. Como diferença evidente temos o tamanho dos dicionários atuais, bem como na quantidade de regras nas listas criadas para desambiguações disponíveis para uso hoje em dia. Baseado nessas fases de processamento, Karlsson [25] criou o *Constraint Grammar* (CG) uma metodologia dependente de contexto que sendo compilada em uma gramática é capaz de realizar a anotação de palavras de um texto. Esse trabalho se tornou uma base para diversas propostas que buscam adicionar funcionalidades ao CG.

Como exemplo de *tagger* baseado em CG e que segue esse paradigma temos o *EngCG*. Esta implementação, seguindo o mesmo caminho de implementações CG mais antigas, etiqueta cada palavra no texto com todas as possibilidades encontradas em seu dicionário em um primeiro momento. Após essa listagem e devida etiquetagem, a segunda etapa sustentada por mais de 3700 regras de desambiguação começa com a mineração das opções existentes, eliminando as indicações inválidas culminando em um texto anotado adequadamente com apenas uma *tag* para cada palavra, tendo o objetivo de sustentar a melhor opção em cada desambiguação. As regras podem ser realmente complexas, levando em consideração diversas situações que encadeadas culminam em adições ou remoções de etiquetas durante o processamento. Na Figura 1 pode-se observar uma regra simplificada com alguns argumentos, utilizada quando no texto se encontra a palavra "*that*".

```
1  REGRA ADVERBIAL "THAT"
2  AO RECEBER "that"
3  SE
4      (+1 A/ADV/QUANT);/* Se a próxima palavra é um adj., advérbio ou quantificador */
5      (+2 SENT-LIM);   /* e a seguinte é um delimitador de sentença */
6      (NOT -1 SVOC/A); /* e a anterior não é um verbo que permita adj. como complemento. */
7  ENTÃO eliminar tags não-ADV
8  SENÃO eliminar tag ADV
```

Figura 1- Regra relativa à palavra "*that*" no idioma inglês em uma tradução livre.

Fonte: Jurafsky e Martin [24]

Nota-se que uma regra pode ser formada com diversas premissas necessárias e com variações de comportamento, para uma situação de casamento das premissas ou uma situação adversa. Os trabalhos de Aduriz e Illarraza [3] e Brill [4] se encaixam nessa área de estudo.

2.2.2 Part-Of-Speech Tagging Usando HMM

A intenção de se utilizar estatística como parte da solução de POST, não é uma ideia nova. O uso de probabilidades foi visto na década de 60 em um trabalho de Stolz [26]. Na década de 70, Bahl e Mercer [27] apresentaram um protótipo de *tagger* probabilístico que utilizada decodificação através do algoritmo de Viterbi[2]. Durante os anos 80 diversos *taggers* estocásticos foram criados, Church [10] e DeRose [29] foram alguns dos autores.

Uma das soluções mais vistas em trabalhos de POST baseado em probabilidade é o uso de *Hidden Markov Models* como algoritmo estocástico de etiquetagem. Jurafsky e Martin definem a utilização de HMM como sendo um caso especial de inferência Bayesiana também conhecida por classificação Bayesiana, paradigma que foi originalmente conhecido através do trabalho de Bayes [30]. Utilizando uma explicação simplificada, essa inferência busca encontrar em uma sequência de palavras, como em uma frase, quais são as *tags* que são as mais corretas, partindo de uma etapa inicial onde todas as *tags* são válidas. Diversas fórmulas são utilizadas para encontrar o objetivo através desse paradigma, no entanto a utilização HMM traz consigo duas premissas facilitadoras.

- A probabilidade de uma palavra aparecer é determinada somente por sua própria POS *tag*, isto é, essa possibilidade é independente das palavras e *tags* ao seu redor.

- A probabilidade de uma *tag* aparecer no texto é dependente apenas da *tag* imediatamente anterior, e não leva em consideração a sequência completa das *tags.*

Com as premissas apresentadas, uma nova fórmula baseada nas teorias de Bayes pode ser obtida. A fórmula em questão, que pode ser observada na figura 2, define uma equação onde um etiquetador de bigramas, é capaz de estimar a mais provável sequência de *tags.*

[2] Viterbi é um algoritmo de programação dinâmica que tem como objetivo encontrar o "Viterbi Path" que seria uma determinada sequência de estados ocultos. Possui esse nome devido ao seu criador Andrew Viterbi [28].

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1})$$

Figura 2 - Equação de HMM com dois tipos de probabilidades, transição de *tags* e possibilidade de palavras.
Fonte: Jurafsky e Martin [24].

Como exemplos desse método podem ser citadas as propostas de Giménez e Màrquez [5] e Segond et.al. [7].

Explicando um pouco melhor a solução apresentada por Gimenez e Marques [5] temos um pacote de ferramentas que se baseiam no uso de *Support Vector Machines* (SVM). Esse pacote consiste em três componentes principais divididos a partir das funções por eles executadas: SVMlearn (componente de aprendizado), SVMTagger (o etiquetador) e por fim o SVMeval, componente este responsável pela avaliação dos resultados obtidos pelos outros dois. Focando exclusivamente na função *tagger* temos uma solução robusta que percorre linha a linha do corpus anotando as palavras, levando em consideração para a palavra seguinte, a informação obtida sobre a palavra atual. Demonstrando flexibilidade, o SVMtagger pode ser configurado para avaliar as palavras em contexto reduzido, ou a nível de sentença. As direções que o *tagger* percorre também podem ser ajustadas, resolvendo da "direita para a esquerda" e vice-versa. Entre outras configurações uma opção importante é a utilização de múltiplos passos de etiquetagem o que embora envolva mais tempo, normalmente realiza uma desambiguação de melhor qualidade final.

O trabalho de Segond et.al. [7], no entanto, buscou utilizar o HMM em seu formato clássico formulando uma desambiguação de palavras totalmente baseada em probabilidade. Este trabalho usou como contexto o formato apresentado anteriormente baseado em bigramas. Desta forma as cadeias são montadas de maneira onde apenas o estado imediatamente anterior possui relevância. Para ajudar a compreender e visualizar como isso funciona, é interessante comentar como a criação das Cadeias de Markov (*Markov Chains*) ocorre. Como explicação trivial, pode-se dizer que a cadeia de Markov pode ser vista como um autômato finito, que possui transições acionadas pela ocorrência de processos estocásticos [32]. Embora os modelos markovianos possam ser encontrados nas mais diversas combinações (Stochastic Automata Networks, HMM, etc.) existem apenas dois tipos de cadeias de Markov:

- CTMC - *Continuous Time Markov Chains*

- DTMC - *Discrete Time Markov Chains*

Sua principal diferença consiste na forma em que cada tipo descreve uma transição entre os estados possíveis da cadeia. Enquanto o CTMC descreve cada transição entre estados como taxas de ocorrência, o DTMC apresenta a transição como probabilidade. Partindo do pressuposto que as cadeias de Markov tiveram como base as ideias de Bayes, sabemos que a soma das N possibilidades existentes de transições em um DTMC deve ter seu valor somado igual ou inferior a um.

Exemplificando graficamente uma simples cadeia de Markov DTMC, temos a figura 3.

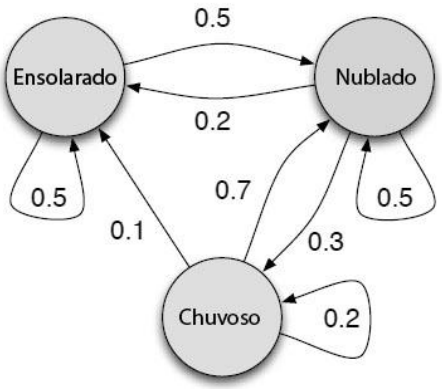

Figura 3 - Cadeia de Markov DTMC simples representando possibilidades climáticas.

Condicionado à ideia de que cada aresta representa em valor probabilístico de cada estado transacionando para outro, podemos obter disto, inferências. Um clima chuvoso possui uma probabilidade de 0.2 de continuar assim, enquanto tem 0.7 de chance de tornar-se nublado. Desta forma resta a probabilidade 0.1 de que o clima atualmente chuvoso torne-se ensolarado. Seguindo essa ideia podem-se averiguar as possibilidades validas de cada estado e a chance de cada transição acontecer. Uma grande característica das cadeias markovianas representada no modelo é a ausência de memória. Também conhecida como propriedade markoviana essa condição faz com que apenas o estado atual seja relevante no momento do cálculo da possibilidade futura, não interessando também a quantidade de tempo atualmente utilizado no estado corrente.

No caso específico do trabalho de Segond et. al [7] foi utilizado um *tagger* mais antigo, de 1992 tendo base em HMM e descrito formalmente no trabalho de Cutting et. al. [11]. Este por sua vez apresenta o formalismo do modelo e quais algoritmos são geralmente utilizados para a obtenção

de estimativas, sendo comumente usados ou a proposta de Viterbi ou o trabalho de Baum Welch, como mostra solução apresentada por Baum [33].

Entrando no detalhe de ambas as soluções percebemos que esses aplicativos visam resolver o problema da desambiguação através de um grupo de HMMs que são geradas automaticamente na fase de aprendizagem do *tagger*, fase essa que ocorre normalmente como uma prévia dos testes reais. Neste momento é que os ajustes finos da aplicação podem ser efetuados resultando assim em uma solução mais correta dentro de um determinado tema textual. Para manter um bom nível de eficiência os *taggers* costumam ser treinados com textos de mesma área de conhecimento daqueles que serão utilizados nos futuros testes. Isso ajuda a catalogar as virtudes e deficiências de uma determinada solução bem como facilitar a comparação desta com as demais que também se focam em um determinado nicho textual.

2.2.3 Part-Of-Speech Tagging Híbrido

Para melhorar o desempenho de acerto (palavras corretamente etiquetadas), normalmente menor dos métodos estatísticos e também conseguir ser mais específico quando trabalhando com um idioma mais complexo, diversos autores partiram para a utilização de um método híbrido. Este método que visa pegar o melhor dos dois mundos, o da estatística e o das regras, concentra as melhores facetas de cada método tentando assim elaborar uma solução mais eficiente daquela alcançada apenas com um paradigma [31].

2.3 Desambiguação

No momento que lemos um texto, podemos nos deparar com diversas situações onde uma determinada palavra ou conjunto de palavras podem possuir significados distintos. Essa desambiguação acontece normalmente durante uma leitura já que estamos com o contexto das palavras em nossa memória. No entanto se apenas um conjunto de palavras com ambiguidade forem apresentadas sem um contexto adequado, escolher a melhor opção de significado se torna uma tarefa árdua [23].

Dentro de cada formato de POS *tagger* utilizado, uma maneira diferente é encontrada para lidar com esse problema. No caso dos *taggers* probabilísticos, estes fazem uso basicamente do algoritmo de Viterbi para desambiguar as palavras que acabam possuindo mais de uma *tag* após a análise superficial do texto (*shallow parsing*). Os *taggers* que possuem um processamento baseado no uso de regras pré-determinadas, aprendidas ou inseridas manualmente, utilizam-se dessa mineração de possibilidades relevantes que são filtradas em cada iteração. Essa etapa de mineração

dos dados, através das regras termina quando apenas uma *tag* é escolhida para ser anotada em cada palavra do texto. Existem ainda os formatos utilizados pelas soluções híbridas que fundem algumas etapas para tentar obter uma taxa de acerto mais elevada do que as soluções originais. Estudos como o de Tapanainen e Voutilainen [21] já alcançaram até 98% de taxa de acerto utilizando métodos híbridos.

2.4 Trabalhos Similares

Os trabalhos similares buscados foram aqueles que conseguiram em seus diversos métodos bons resultados em relação às técnicas mais conhecidas. Ainda, para ter uma boa visão das possibilidades que essa área de estudo apresenta, foram escolhidos trabalhos que usem maneiras de POS *tagging* baseadas em probabilidades ou que utilizem regras de desambiguação.

Na área de desambiguação através de regras, o trabalho de Brill [4] pode ser considerado um artigo base. Essa proposta descreve um algoritmo que é capaz de realizar aprendizagem sem supervisão, aprendendo assim a partir de corpus não anotados manualmente. Neste trabalho ele também apresenta uma integração das possibilidades de treinamento para a obtenção de regras, combinando algoritmos de treinamento supervisionado e não supervisionado criando assim um *tagger* de alto desempenho com uma necessidade pequena de textos pré-anotados.

Seguindo na linha de desambiguação baseada em regras, o trabalho de Aduriz e Illarraza [3], aparece como um trabalho onde uma análise de ambiguidades morfossintáticas é realizada, sendo focado especialmente no idioma basco. Essa proposta utiliza como *parser* o já comentado *Constraint Grammar* (CG) [25] que já foi diversas vezes utilizado na criação de gramáticas para vários idiomas diferentes. Sua colaboração foi apresentar um resultado de melhoria nas tarefas de identificação e ambiguidades em funções sintáticas e morfossintáticas utilizando uma análise superficial. Para em seguida realizar as desambiguações através das mil regras geradas para este fim.

Exemplo de um representante da linha de pesquisa de *taggers* estatísticos, o trabalho de Giménez e Màrquez [5] apresenta uma proposta de POS tagger baseado em *Support Vector Machines* chamado de *SVMTool*. Essa ferramenta se apresenta como uma opção simples, flexível, e eficiente para as necessidades atuais do processamento de linguagem natural. De acordo com os autores a *SVMTool* simples de utilizar necessitando de poucos parâmetros para funcionar em seu formato na linguagem *Pearl*. O contexto a ser adotado na ferramenta também é passível de ajuste, definindo tamanhos dos N-gramas (bigramas, trigramas, etc.), podendo ajustar também o tempo de *tagging* a ser executado. Outra vantagem apresentada no trabalho é que a ferramenta tende a se

comportar bem não importando o idioma utilizado. Nos testes, utilizando tanto o inglês quanto o espanhol foi possível observar marcas consideráveis nas taxas globais, (96,16% e 96,89% respectivamente). Para tanto além de uma etapa de aprendizagem não supervisionada de textos no idioma pretendido, é necessário adicionar às suas configurações dicionários morfossintáticos compatíveis com estas linguagens.

Mantendo as propostas baseadas em probabilidades, o trabalho de Segond et.al.[7] apresenta um experimento que através de HMM consiste em uma formatação clássica de um *tagger* deste estilo. A preparação de dados se dá com a obtenção de todas as *tags* semânticas possíveis obtidas na *WordNet*[3]. Em seguida, tendo como origem o *Brown Corpus,* foi gerado um *corpus* de treinamento e um corpus para os testes propriamente. Nos testes o modelo HMM utilizado apenas era capaz de interpretar bigramas, isto é cada palavra apenas leva em consideração a palavra imediatamente anterior. Por fim realizado o processamento do *corpus* de treino a fim de realizar os ajustes necessários no algoritmo de *tagging,* foi processado o *corpus* de teste. Na sequência foi realizada uma comparação das *tags* encontradas nos no processamento, com as já existentes no conjunto de palavras do treinamento inicial que já haviam sido etiquetadas manualmente. Três testes distintos foram realizados, conseguindo uma taxa máxima de acerto global de 89%. Como possibilidade de melhoria o trabalho conclui que adicionar a capacidade de consulta a dicionários sintáticos poderia incrementar o desempenho obtido através da formatação clássica executada.

Como exemplo a ser destacado, com um formato já ajustado e adequado à utilização do idioma português encontra-se a tese de Domingues [34] que propõe uma abordagem completa para o desenvolvimento de um etiquetador de alta acurácia para o português do Brasil. Esse estudo exploratório apresenta solução que foi idealizada como uma visão híbrida combinando etiquetagem probabilística e etiquetagem baseada em regras. Foram utilizados quatro versões dos seguintes corpora CETENFolha, Bosque CF 7.4, Mac-Morpho e Selva Científica. A solução de *tagging* foi utilizar ferramentas *open source* já existentes, entre elas o gerador de regras automatizado μ-TBL (*Micro transformation-based learning*) e o TreeTagger como solução de tagging. Sua solução final apresenta diversas fases de processamentos, que começa com a tokenização do texto, passando pelo *parser* estatístico. Após essa etapa ocorre a consulta por nomes próprios no texto e por fim a etiquetagem baseada em regras. Sendo esta última etapa diferenciada, pois se utiliza de três grupos

[3] WordNet é um grande banco de dados com dados léxicos no idioma inglês. Cada um expressando um conceito, substantivos, verbos, adjetivos e advérbios são agrupados em conjuntos de sinônimos cognitivos, os *synsets.*

de regras. Todo esse processo permitiu que os experimentos com melhor desempenho atingissem uma taxa de acerto global superior a 98%.

Para sintetizar as informações coletadas nos trabalhos similares apresentados, as principais características relevantes para este trabalho foram organizadas e são apresentadas na Tabela 1. Esta análise, todavia não possui um caráter de escolha entre as propostas a seguir apresentadas. Servem, no entanto para obter um entendimento sistemático das opções existentes para o problema. Desta forma a contínua observação dos trabalhos já utilizando as ferramentas adequadas para este problema, colabora na compreensão ao produzir subsídios necessários para a escolha de um método de desambiguação próprio. Esse método busca trazer uma nova contribuição ao, aplicar soluções realizadas em outros idiomas tornando-o útil para textos em português, ou ainda conseguir uma mescla destas possibilidades.

Tabela 1 - Comparação entre trabalhos similares encontrados na bibliografia

Identificação da Proposta	**Tipo de *tagging***	**Máxima acurácia obtida em testes**	**Em qual idioma é aplicada**	**Tipo de desambiguação utilizado**
Aduriz e Illarraza [3]	Baseado em regras, usando uma gramática própria.	97.51%	Basco	Regras de desambiguação
Brill [4]	Baseado em regras, com aprendizagem sem supervisão. (Transformation-Based Learning)	96%	Inglês	Regras de desambiguação
Giménez e Màrquez [5]	Utilizando *Support Vector Machines* (SVM)	97.16%	Inglês, Espanhol	Estatístico
Segond et.al.[7]	Baseado em HMM	89%	Inglês	Estatístico
Domingues [34]	Processo híbrido baseado em probabilidade e regras	98,30%	Português do Brasil	Estatístico/Regras

3 DESENVOLVIMENTO

Neste capítulo é apresentada a proposta para desambiguação de classes gramaticais em texto de língua portuguesa, objetivo final deste trabalho. Também são apresentadas as ferramentas utilizadas para a obtenção da solução final e seu processo de estruturação arquitetural evolutivo.

3.1 Arquitetura da Solução

Como visão geral do processo proposto neste trabalho para a desambiguação morfossintática de textos da língua portuguesa. A fonte de dados utilizada, no caso o conjunto de palavras corretamente anotadas foram as extraídas do corpus Mac-Morpho.

Embora o Mac-Morpho já possua suas regras de anotação própria, bem como ter sido validado por especialistas em gramática, a solução apresentada utiliza o WAGGER[4] como *tagger*. Utilizando o mesmo conjunto de palavras o WAGGER por sua característica pode anotar corretamente uma palavra, entendo uma opção única de classe gramatical na situação ótima. No entanto, em caso de múltiplas possibilidades ele acaba por etiquetar a mesma palavra com diversas classes. Essa situação corriqueira de ambiguidade das palavras do texto representa o mote para executar a segunda fase do processo, desambiguando as classes de cada palavra etiquetada mais de uma vez.

Devido ao fato de atualmente o WAGGER utilizar uma quantidade menor de etiquetas válidas para análise das palavras, foi necessário um mapeamento das classes gramaticais identificadas pelo Mac-Morpho a fim de propiciar um comparativo verossímil. Desta forma, temos uma correlação onde algumas *tags* do Mac-Morpho são representadas na definição do WAGGER por uma única etiqueta.

Esse mapeamento entre as etiquetas apresentado na Tabela 2 demonstra as classes ou grupos de classes existentes em cada caso, bem como a etiqueta representativa usada nessas situações.

[4] Word clAss taGGER é um aplicativo que etiqueta as palavras de um arquivo texto, utilizando uma estrutura Multi-Terminal Multi-valued Decision Diagram (MTMDD) para armazenar o conjunto de palavras e suas classes gramaticais.

Tabela 2 - Mapeamento entre etiquetas do WAGGER com as usadas no Mac-Morpho

MacMorpho Classe Gramatical	MacMorpho Tag	Wagger Tag	Wagger Classe Gramatical
ADJETIVO	ADJ	ADJ	Adjetivo
ARTIGO (def. ou indef.)	ART	DET	Artigo
ADVÉRBIO RELATIVO SUBORDINATIVO	ADV_KS_REL	ADV	Advérbio
ADVÉRBIO CONECTIVO SUBORDINATIVO	ADV_KS	ADV	Advérbio
ADVÉRBIO	ADV	ADV	Advérbio
SÍMBOLO DE MOEDA CORRENTE	CUR	NUM	Número
INTERJEIÇÃO	IN	INTERJ	Interjeição
CONJUNÇÃO SUBORDINATIVA	KS	CONJ	Conjunção
CONJUNÇÃO COORDENATIVA	KC	CONJ	Conjunção
NOME PRÓPRIO	NPROP	N	Nome
NUMERAL	NUM	NUM	Número
NOME	N	N	Nome
PARTICÍPIO	PCP	V	Verbo
PALAVRA DENOTATIVA	PDEN	ADV	Advérbio
PREPOSIÇÃO	PREP	PREP	Preposição
PREPOSIÇÃO	PREP	PREP+PREP	Preposição+Preposição
PREPOSIÇÃO+ARTIGO	PREP+ART	PREP+DET	Preposição+Artigo
PREPOSIÇÃO+PRONOME SUBSTANTIVO	PREP+PROSUB	PREP+P	Preposição+Pronome
PREPOSIÇÃO+ADVÉRBIO	PREP+ADV	PREP+ADV	Preposição+Advérbio
PREPOSIÇÃO+PRONOME ADJETIVO	PREP+PROADJ	PREP+P	Preposição+Pronome
PREPOSIÇÃO+PRONOME PESSOAL	PREP+PROPESS	PREP+P	Preposição+Pronome
PREPOSIÇÃO+PRONOME-CONJUNÇÃO SUBORDINATIVA	PREP+PRO-KS	PREP+P	Preposição+Pronome
PRONOME ADJETIVO	PROADJ	P	Pronome
PRONOME CONECTIVO SUBORDINATIVO	PRO_KS	P	Pronome
PRONOME RELATIVO CONECTIVO SUBORDINATIVO	PRO_KS_REL	P	Pronome
PRONOME PESSOAL	PROPESS	P	Pronome
PRONOME SUBSTANTIVO	PROSUB	P	Pronome
PRONOME SUBSTANTIVO	PROSUB	P+P	Pronome+Pronome
PONTUAÇÃO	PU	PU	Pontuação
VERBO AUXILIAR	VAUX	V	Verbo
VERBO	V	V	Verbo
NOME	N	ACR	Acrônimo
NOME	N	ABBR	Abreviatura
ADJETIVO	ADJ	PFX	Prefixo

A solução proposta no trabalho, em uma visão de alto nível, é apresentada na Figura 4 onde se observa sete fases distintas. (i) Em um primeiro momento as etiquetas do corpus original são removidas. (ii) Utiliza-se o WAGGER como ferramenta para realizar o *tagging* deste texto em formato de uma sentença por linha e puramente texto. (iii) Com o corpus já etiquetado pelo WAGGER, acontece uma análise para verificar a existência de palavras com múltiplas *tags*, o que indica a necessidade de desambiguação. (iv) Como opção trivial de desambiguação, há a escolha aleatória de alguma *tag* entre as indicadas para a palavra, no caso desta possuir múltiplas opções válidas na visão do WAGGER.

(v) Em uma opção com maior análise, a definição da melhor etiqueta para uma palavra a ser desambiguada ocorre após a análise estatística das classes mais utilizadas por ela no corpus. Ex. no corpus original do Mac-Morpho temos a palavra “modelo” sendo um substantivo em 80% dos casos e um nome próprio em 20%, desta forma a escolha da *tag* baseia-se num cálculo ponderado sobre esses valores. (vi) Utilizando um grafo orientado como base de dados, este método de desambiguação analisa as classes gramaticas das palavras vizinhas. A análise do conjunto de palavras e suas classes influencia sua possibilidade de escolha se este conjunto de vizinhos acontece com frequência no corpus original devidamente anotado. (vii) Por fim temos o texto desambiguado com apenas uma etiqueta para cada palavra, com o qual podemos realizar comparativos com o corpus anotado em seu formato original.

Todo esse processo é devidamente explicado nas seções seguintes trazendo informações relevantes a respeito das escolhas realizadas em nível de programação utilizada na ferramenta de desambiguação quanto algumas opções técnicas escolhidas em conjunto com a orientação. Essas escolhas guiaram o processo de construção do projeto bem como se propuseram a focar nas soluções factíveis de serem realizadas durante o período do curso.

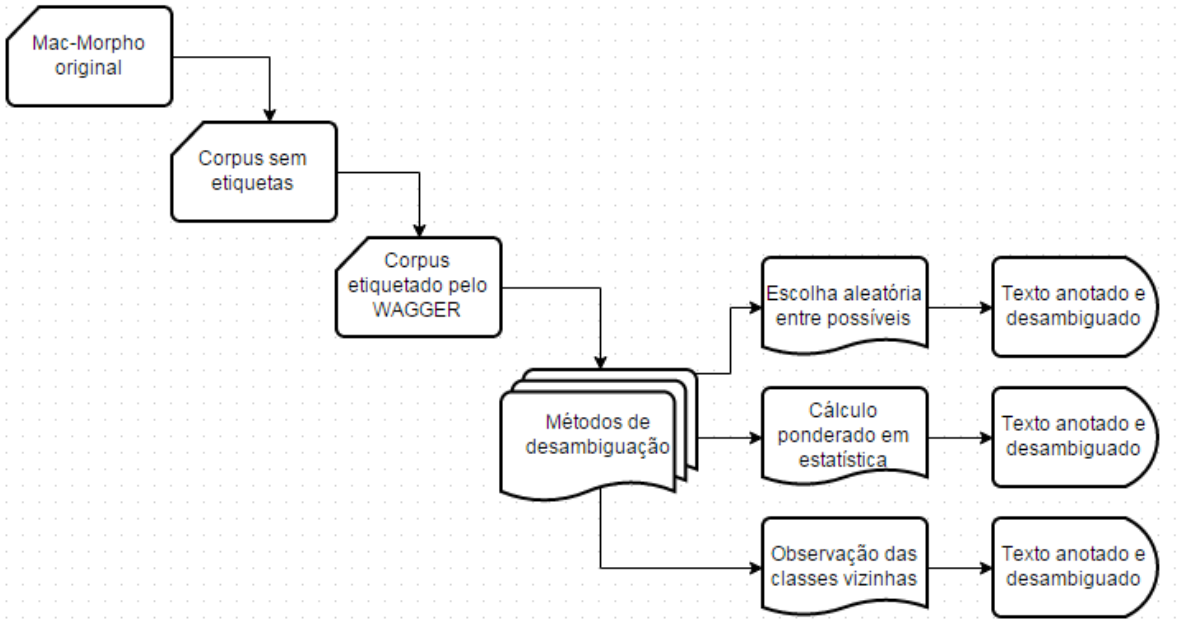

Figura 4. Estrutura de fases do processo de desambiguação

3.1.1 Mapeamento de Remoção de Tags

Como primeira etapa desse processo de desambiguação conforme citado anteriormente, acontece fase de adequação do corpus Mac-Morpho com o objetivo de remover suas etiquetas e ajustar o texto. Assim sendo, o texto consistindo em basicamente por uma sentença por linha do arquivo, num total de cerca de 39950 linhas contendo mais de 760 mil itens para serem etiquetados. A remoção propriamente dita é executada por um código em *Groovy*[5] que percorre o arquivo, linha a linha removendo todas as etiquetas existentes. Na sequência é criado um novo arquivo utilizando o a codificação ISO8859-1, formato necessário para o arquivo ser corretamente interpretado e analisado pelo WAGGER. Esse novo arquivo, com as sentenças finalmente ajustadas bem como livre de etiquetas será utilizado como corpus base durante a avaliação deste trabalho.

3.1.2 Etiquetagem Utilizando WAGGER

A segunda etapa do processo visa utilizar o etiquetador WAGGER tendo como parâmetro de entrada o arquivo de sentenças refinado resultante do passo anterior. Este etiquetador foi

[5] Groovy é uma poderosa linguagem dinâmica, opcionalmente tipada com capacidade para tipagem e compilação estática, para a plataforma Java possuindo uma sintaxe concisa, familiar e fácil de aprender. [35]

configurado com um dicionário de palavras na língua portuguesa contendo mais de 878600 entradas. Cabe lembrar que esse dicionário foi modificado manualmente, adicionando-se a ele algumas palavras. Este dicionário apenas possui palavras em idioma português, fazendo com que o WAGGER não seja capaz de identificar termos e palavras nos demais idiomas. Para termos uma ideia de cobertura, o dicionário em questão possui 94,17% das palavras encontradas no corpus de teste. Levando-se em conta tanto palavras do idioma português ainda não existentes no dicionário bem como termos de outros idiomas que não são reconhecidos como dito anteriormente. Outro fato relevante sobre esse dicionário é a quantidade de palavras inseridas múltiplas vezes. Isso acontece quando uma determinada palavra pode ser utilizada por diversas maneiras em uma frase, dependendo do seu contexto. Um exemplo seria a palavra "que" encontrada em três maneiras distintas. Ela foi catalogada como preposição, conjunção e advérbio. Essa multiplicidade, por certo influencia os resultados dos testes, levando-se em consideração que essas palavras normalmente são as amplamente usadas em textos. Em números absolutos essa duplicidade em mais de quarenta mil casos o que resultaria em 4,63% de duplicações no dicionário. Usando esse dicionário o aplicativo é capaz de montar um diagrama MTMDD especifico englobando essa lista de palavras, com o qual será o conteúdo avaliado durante seu processamento de etiquetagem. O processo de como se comporta e funciona o WAGGER em uma visão de maior profundidade não faz parte do contexto deste trabalho, contudo pode ser analisado em detalhes no trabalho de Paulo et. al.[6]. O WAGGER tem como pré-requisito básico de funcionamento algumas bibliotecas do ambiente Linux, desta forma essas configurações bem como a etiquetagem propriamente dita devem ser executadas em sistemas operacionais UNIX. Para este trabalho, foi utilizado um ambiente virtualizado do Linux Ubuntu 14.10, que permitiu o funcionamento correto do WAGGER sem que tenha sido necessária qualquer configuração extra.

Como resultado da execução dessa fase de etiquetagem, temos um arquivo separado em mais de 9400 sentenças. Onde cada sentença possui suas palavras, tendo ao seu lado uma ou mais classes gramaticais encontradas pelo dicionário do WAGGER. A separação das palavras em sentenças é feita de maneira automática durante a etiquetagem, basicamente observando as pontuações como ponto final, ponto de interrogação, exclamação como sinal de fim de uma frase. Outro arquivo também é gerado informando todas as palavras não reconhecidas pelo etiquetador. Estas são aquelas não existentes no dicionário, em grande maioria nomes próprios ou palavras em idioma estrangeiro.

Com o texto das sentenças criado e etiquetado, o passo seguinte consiste em analisar o arquivo e identificar quais palavras necessitam de desambiguação, isto é, palavras que foram anotadas com duas ou mais etiquetas.

3.1.3 Análise e Desambiguação

O arquivo obtido após a utilização do WAGGER será peça fundamental na realização de nossas diversas desambiguações. Todos os três processos de desambiguação formulados possuem o mesmo texto como parâmetro basal além de possuírem objetivos equivalentes, no entanto atribui-se funcionalidades distintas que os fazem percorrer esse caminho. De um modo geral, o algoritmo de desambiguação percorre linha a linha do texto, esperando encontrar em cada linha uma palavra ou definição de sentença e suas etiquetas relacionadas, quando houver.

No momento em que se encontra uma palavra com mais de uma etiqueta, inicia-se o método de desambiguação, levando em consideração as possibilidades elencadas anteriormente através do etiquetador WAGGER. Tomando por exemplo a palavra “nova” encontrada na primeira frase do texto analisado, encontramos entre as opções possíveis de etiqueta: N, ADJ e V. Assim sendo, no processo de desambiguação desta palavra nesta localidade, devemos decidir se no contexto de sua sentença, a palavra “nova” encaixa-se como substantivo (N), adjetivo (ADJ) ou verbo (V). Isso deve acontecer para todas as palavras do texto, excetuando as linhas que delimitam as sentenças, bem como as palavras que não possuam múltiplas etiquetas assinaladas, não requerendo enfim uma desambiguação.

Nos trabalhos similares acontece uma etapa similar de identificação de multiplicidade e desambiguação, entretanto algumas soluções de software mascaram essa etapa, traduzindo o texto diretamente para sua versão final com apenas uma etiqueta por palavra. Outros casos existentes se valem da utilização de processos baseados em regras pré-definidas que são aplicadas ao texto, a fim de realizar uma desambiguação padronizada.

A desambiguação apresentada nesse trabalho ocorre após pesquisas diretas ao texto, formatado em grafo utilizando uma ferramenta de banco de dados baseados em grafos. Essa opção permite com que todos os dados relativos à desambiguação como palavras, etiquetas e sua distribuição estejam, em um único ambiente conciso, inter-relacionado da maneira que for mais pertinente. Na sequência é apresentada a ferramenta e quais benefícios ela trouxe ao projeto.

3.2 Visualização das Anotações Morfossintáticas

A ferramenta escolhida para visualizar e colaborar no decorrer do processo que define as etiquetas corretas nas palavras foi o *Neo4j* [6] Essa ferramenta consiste em uma suíte completa para se trabalhar com bancos de dados baseados em grafos. Tendo em vista a uma ideia semelhante aplicada no WAGGER e seus mapas MTMDD a utilização de um ferramental que também se baseia em grafos foi natural para seguir a mesma linha de raciocínio por todo o trabalho.

Sua utilização propriamente dita ocorre como repositório de todas as palavras inclusas no texto, em sua formação original, bem como completa lista de etiquetas compreendidas pelo WAGGER além de um formato de dados que define um mapeamento de qual etiqueta foi utilizada por determinada palavra e às palavras próximas.

Atendo-se ao fato de que um banco de dados baseado em grafos foge das estruturas fixas, não há necessidade de preocuparmo-nos com tabelas e formatações estritas de dados, tudo é baseado em duas unidades básicas nodo e relacionamento. O nodo nesse caso é quem carrega as propriedades, e o relacionamento é o que interliga os diversos nodos.

No primeiro passo realizado, as palavras do texto foram adicionadas como nodos, possuindo seu valor na propriedade "word". As palavras não foram duplicadas no banco de dados. Como não há problemas de haver múltiplos relacionamentos, a mesma palavra pode se relacionar com diversas etiquetas simultaneamente.

Com o intuito de tornar as consultas mais rápidas, o Neo4j oferece uma solução semelhante ao índice dos bancos de dados relacionais, um *label*. Essa opção permite definir um *alias* comum a determinado grupo de nodos ou relacionamentos, fazendo com que esse grupo seja também indexado, buscando diminuir a demora no retorno das consultas realizadas. O nome do *label* utilizado como índice dos nodos que representam todas as palavras do *corpus* é "Palavra".

Em um segundo momento, a lista completa de etiquetas conhecidas pelo MacMorpho e consequentemente todas as possibilidades de desambiguação foi adicionada ao grafo. Essas etiquetadas foram adicionadas como nodos, tendo como propriedade *tag* contendo o nome da

[6] Neo4j é uma solução em banco de dados baseado em grafos que utiliza arquitetura Java. Possui versões open source e comercial [36].

etiquetada em questão. Esse conjunto de nodos também foi indexado, recebendo o *label* nomeado “MacMorphoTag”.

Por fim o grafo recebeu um novo grupo de nodos, que tinha como valores de definição a informação da etiqueta correta da palavra, bem como as etiquetas de seus vizinhos. Assim sendo esse grafo de *label* “TagGroup” tinha como propriedade a informação morfossintática de um trigrama. A formatação específica, e como essa informação foi construída é detalhada na seção seguinte que aborda os experimentos realizados nesse trabalho.

4 EXPERIMENTOS

O Capítulo 4 apresenta os resultados obtidos através de três experimentos realizados com o processo proposto. Além disso, também apresenta os resultados obtidos durante a avaliação do trabalho. Este Capítulo está organizado em Estudo de Caso e Avaliação e Análise.

4.1 Estudo de Caso

Com o objetivo de verificar o funcionamento e desempenho do processo proposto, foi desenvolvido um sistema computacional em *Groovy* para operacionalizar as etapas de ajustes do corpus, análise dos arquivos obtidos através do WAGGER bem como desambiguar as palavras eventualmente efetuando consultas ao banco de dados em formato de grafo.

O corpus utilizado durante os experimentos foi o Mac-Morpho [37], criado pelo NILC (Núcleo Interinstitucional de Linguística Computacional) da USP de São Carlos. É um corpus totalmente construído a partir de textos noticiais em português. Foi utilizada a última versão disponível para download em seu site, efetuado em Agosto de 2014. Possuindo em seu texto de treinamento um total de 39945 linhas contendo mais de 760 mil palavras.

A proposta deste trabalho desde sempre foi a revisitar o problema da desambiguação partindo de soluções simples para evoluir em opções que possam atingir uma taxa de acerto relevante. Desta maneira, iniciaram-se os trabalhos com uma proposta de desambiguação que se utiliza unicamente de métodos de escolha pseudoaleatórios a fim de escolher a etiqueta para uma palavra com ambiguidade. Essa metodologia foi abordada no item 4.1.1. Seguindo no passo de evolução o novo formato de desambiguação escolhido faz uso de estatística, baseando-se em dados obtidos previamente (em uma análise do corpus), que norteiam um cálculo estatístico utilizado para definir a etiqueta mais adequada. O método ponderado através dos valores e quantidades relativos ao determinado uso das etiquetas, é apresentado no item 4.1.2. Como último método apresentado nesse trabalho, apresenta-se uma fórmula que agrega à ideia de calculo estatístico anterior, uma informação a respeito das palavras vizinhas à ambígua. Assim sendo, as palavras vizinhas e suas anotações eventuais afetam o comportamento do etiquetador, em uma tentativa de buscar uma maior acuidade em suas escolhas. A opção que apresenta uma análise estatística com informações agregadas relativas às palavras vizinhas às desambiguadas é mostrada no item 4.1.3.

4.1.1 Desambiguação Aleatória

Como o próprio nome diz, esse experimento tem como o objetivo testar a arquitetura da solução enquanto entrega um texto totalmente desambiguado. Levando em consideração o processo utilizado para a desambiguação a expectativa de acerto é a mais comedida nessa situação.

Esse experimento iniciou-se com o ajuste do corpus Mac-Morpho, removendo suas etiquetas e ajustando a codificação de seu arquivo para ISO8859-1. Com esse arquivo foi executado o etiquetador WAGGER configurado com o dicionário de palavras já citado anteriormente.

O arquivo resultante possui milhares de palavras a serem desambiguadas. Na figura 5 é apresentado um fragmento do texto resultante da execução do WAGGER no corpus. Pode ser observada a definição de uma sentença (linha com S# ao lado do número referente à sentença) bem como um grupo de palavras que foram anotadas com diversas etiquetas, sendo então passíveis de desambiguação. Nota-se também que há uma palavra, "semana", que não possui mais do que uma etiqueta, nesse caso considera-se essa palavra como adequadamente definida.

```
S#1
        salto N V
        sete N NUM
        o N DET P
        grande N ADJ
        assunto N V
        da PREP+DET PREP+P
        semana N
        ...
```

Figura 5 - Fragmento do resultado do WAGGER, palavras a desambiguar.

Tendo como objetivo a leitura de textos nesse formato, que possui apenas uma palavra com uma ou mais etiquetas por linha foi criado um script em Groovy que realiza esse parse e executa comandos de avaliação após cada linha observada. O código efetua a leitura do arquivo e para cada linha repassa a uma rotina de avaliação básica. Basicamente essa avaliação pode ser descrita em três etapas:

- Etapa um: Verifica se a linha representa um início de sentença: as linhas do texto que possuem somente um valor "S#X" onde X representa o contador de sentenças começado em um.

- Etapa dois: Verifica se a linha possui uma palavra com apenas uma etiqueta vinculada: nesse caso a solução é trivial e a etiqueta vinculada é a que deve ser adicionada no resultado final.

- Etapa três: Verifica se a linha possui uma palavra com duas ou mais etiquetas vinculadas: essa situação é onde acontece o repasse desses dados para o método de desambiguação utilizado nesse momento.

Entendendo que as duas primeiras etapas possuem desdobramentos simples, manter-se-á o foco na situação apresentada pela terceira etapa.

O contexto atual é de estarmos com uma determinada palavra e um conjunto de duas ou mais etiquetas que serão passíveis de avaliação pelo desambiguador. Dessa maneira o script agrupa os valores da etiqueta em uma lista, enviando essa lista para uma desambiguação aleatória. O processo de desambiguação é rápido e como não leva em consideração nada fora a própria lista e métodos randômicos possui valores que podem se modificar a cada execução. O método de geração de aleatoriedade se vale da classe *Random* que faz parte da linguagem Groovy retornando um valor inteiro de zero ao tamanho da lista de etiquetas. O numero obtido é usado como índice na recuperação do valor da lista de possibilidades, produzindo assim uma desambiguação pseudoaleatória.

Os resultados esperados para esse tipo de abordagem são relativamente módicos, cerca de metade dos casos efetuando a desambiguação definida como mais correta. Na seção 4.2 os números obtidos serão expostos e analisados em conjunto com os encontrados nos experimentos subsequentes.

4.1.2 Desambiguação Ponderada

Com o intuito de melhorar os valores obtidos no processo de desambiguação pseudoaleatória, bem como seguir com as tarefas visando uma melhoria continua, esse formato utiliza o histórico como base de decisão. Histórico nesse caso advém da lista de palavras e suas etiquetas mais usadas, calculadas em números absolutos. Nesse formato de desambiguação, o mapeamento entre as palavras e suas definições mais ocorridas constitui a estrutura chave de todo o processo, dessa forma, deve ser executado com antecedência.

A listagem de palavras é obtida através da leitura e compilação de todas as palavras constituintes do corpus Mac-Morpho com suas *tags* originais e consequentemente corretas. A solução encontrada foi criar um novo script que seja capaz de organizar as palavras encontradas no texto, armazenando a quantidade de vezes utilizada, quais etiquetas que ela possuía e a quantidade de situações em que ocorreu. Visando uma solução mais abrangente que também pode trabalhar com outros corpora no futuro, esse script gera um compilado de informações no formato JSON [38] que é um formato de transferência de dados em modo texto, largamente utilizado nas tecnologias atuais.

Essa leitura do texto percorre o arquivo completamente agregando as palavras e seus contadores relacionados a fim de gerar um arquivo JSON que será parâmetro de entrada necessário para o método de desambiguação com cálculo ponderado. Um fragmento do arquivo em seu formato final pode ser observado na Figura 6.

Como se pode observar, a palavra “e” foi encontrada milhares de vezes no corpus, possuindo um total de sete etiquetas distintas. A composição em formato JSON se dá conforme apresentado, onde temos a palavra como item principal do objeto atuando como índice, que se relaciona a um objeto que define seu numero global de aparições como “*totalHits*” e na sequência uma lista de objetos.

Dentro dessa lista cada objeto apresenta a definição da etiqueta (*tagName*), quantas vezes aquela palavra apresentou-se com essa *tag* (*hitNumber*), um identificador numérico para a *tag* (*tagId*) e por fim, o percentual relativo de utilização da etiqueta em proporção ao numero total de situações em que ocorre a palavra (*percentageUsed*). Na tarefa atual a informação crucial desse conjunto de dados é o valor informado no percentageUsed pois é através de um cálculo utilizando-o que a definição de etiqueta acontece.

```
},
"e": {
  "totalHits": 14615,
  "tagStatistic": [
    {
      "tagName": "PREP",
      "hitNumber": 1,
      "tagId": 8,
      "percentageUsed": 0.01
    },
    {
      "tagName": "NUM",
      "hitNumber": 1,
      "tagId": 11,
      "percentageUsed": 0.01
    },
    {
      "tagName": "N",
      "hitNumber": 361,
      "tagId": 2,
      "percentageUsed": 0.03
    },
    {
      "tagName": "CONJ",
      "hitNumber": 14243,
      "tagId": 64,
      "percentageUsed": 0.98
    },
    {
      "tagName": "ADJ",
      "hitNumber": 3,
      "tagId": 61,
      "percentageUsed": 0.01
    },
    {
      "tagName": "P",
      "hitNumber": 1,
      "tagId": 36,
      "percentageUsed": 0.01
    },
    {
      "tagName": "ADV",
      "hitNumber": 5,
      "tagId": 302,
      "percentageUsed": 0.01
    }
  ]
},
```


Figura 6 - Descrição completa de "e" encontrada no corpus Mac-Morpho em JSON

Terminada essa tarefa necessária, pode-se iniciar a etapa de análise e desambiguação. A fim de possibilitar a comparação entre os números obtidos para comprovar um aumento gradual no desempenho em relação ao método anteriormente utilizado, o mesmo conjunto de dados foi utilizado como corpus. Desta maneira, o WAGGER foi o etiquetador mais uma vez, realizando seu processamento e gerando um novo arquivo que deveria ser desambiguado.

O script anterior foi atualizado para que pudesse, nesse momento, efetuar a leitura do arquivo em formato JSON contendo as estatísticas previamente calculadas. Esse código também recebeu melhorias que propunham uma definição de escolha do tipo de desambiguação a ser executada, neste momento as opções são aleatória e ponderada.

Como o enfoque da solução é sempre desambiguar, de modo à nunca haver palavras no relatório final que possuam mais de uma etiqueta, o modo de desambiguação pseudoaleatório é utilizado se por ventura ocorrer algum problema durante a desambiguação ponderada de alguma palavra. Exemplo desse caso seria uma palavra nunca vista anteriormente, não existente no corpus modelo, por consequência sem dados estatísticos suficientes para efetuar o cálculo.

Comentando o algoritmo propriamente, temos uma situação inicial semelhante ao desambiguador anterior, diferindo-se na presença dos dados relativos à palavra que são encontrados no JSON de estatísticas. Essas informações são avaliadas por um método que ajusta essa gama de possibilidades a fim de adicionar possibilidades relevantes que por acaso o WAGGER não adicionou em seu resultado final. O método utiliza-se de um valor inicial de percentual máximo a ser utilizado por essas etiquetas “esquecidas” e não consideradas durante o parse do etiquetador. Essa manobra é efetuada no caso de palavras que possuam apenas uma *tag* em suas estatísticas, assim mesmo que o WAGGER tenha definido apenas uma etiqueta, a palavra pode terminar utilizando outra. A Figura 7 apresenta um trecho do código que define etiquetas que serão usadas em casos de palavras com apenas um valor estatístico no JSON usado como parâmetro.

```
def getAdjustedStatistics(statistics){
    if(statistics.size > 1){
        return statistics
    } else {
        if(statistics[0].tagName == 'N'){
            tagsToUse = ['V', 'ADJ']
        } else if(statistics[0].tagName == 'V'){
            tagsToUse = ['N', 'ADJ']
        } else if(statistics[0].tagName == 'ADJ'){
            tagsToUse = ['V', 'N']
        } else {
            tagsToUse = ['V', 'N', 'ADJ']
        }

        defaultStats = getDefaultTagStatistics(tagsToUse)

        statistics[0].percentageUsed = statistics[0].percentageUsed - defaultPercentageValue
        return statistics + defaultStats
    }
}
```

Figura 7 - Método que adiciona etiquetas possíveis para palavras com apenas uma *tag*

Conforme o código apresenta, esse método não atua nos casos de estatísticas compostas por mais de uma etiqueta. No entanto, baseado no tipo de etiqueta encontrada, um novo grupo é formado e adicionado às possibilidades. O valor "*defaultPercentageValue*" é definido previamente, e fornece o máximo valor percentual que o grupo de novos entrantes pode possuir. A variável *defaultStats* contém a lista do grupo de *tags*, cada um possuindo um percentual divido igualmente entre seu pares. Equilibra-se os valores subtraindo *defaultPercentageValue* do valor estatístico original (100%). Por fim a concatenação desses valores será utilizado no método de desambiguação.

Possuindo todos esses dados em tempo de execução, a desambiguação fica facilitada. O método recebendo a lista de etiquetas possíveis realiza um laço que calcula, em cada iteração, um valor percentual. Sempre que esse valor estiver de acordo com o percentual calculado previamente para a palavra, a *tag* pertencente à iteração atual é escolhida. No caso de não haver dados relativos a alguma palavra, este método utiliza a desambiguação aleatória descrita anteriormente como último recurso. Com a possibilidade de usar o histórico das palavras encontradas em um corpus, esse método tem expectativa de obter números melhores que o método pseudoaleatório. Os valores encontrados serão expostos na seção 4.2 deste documento.

4.1.3 Desambiguação Ponderada Através de Estatística

Seguindo com a ideia de melhorar continuamente os resultados obtidos, um novo método criado buscando utilizar como informação de apoio à desambiguação, os valores de etiquetas existentes

nas palavras adjacentes. Para que isso fosse realizado, um novo script foi criado que é capaz de ler as palavras existentes no corpus original com suas *tags* corretas e inserir essas informações em uma base de dados utilizando o Neo4j. Nesta situação as informações em formato de grafo, permitem uma série de pesquisas no momento em que cada palavra for desambiguada. O resultado dessas consultas, realizadas em tempo de execução permitem uma definição gramatical mais acurada do que as opções anteriores.

O primeiro passo desta proposta é a criação de uma base de dados com todas as informações relevantes disponíveis. Embora o conceito de massa de dados proposto pela ferramenta Neo4j apoiar a definição de que qualquer informação pode se correlacionar livremente, essa organização que separa em três tipos básicos de informação será útil no entendimento da solução.

Os nodos foram organizados em três grupos distintos:

- Palavras encontradas no corpus não havendo repetições. Esse grupo foi indexado usando um *label* com o nome “Palavra”

- Lista unificada de etiquetas utilizadas pelo Mac-Morpho e utilizada ao menos uma vez no corpus. A lista foi adicionada ao índice através do *label* “MacMorphoTag”. A esse grupo foram adicionados dois valores de controle: “START” e “END” que foram utilizados como delimitadores das sentenças nos trigramas.

- Informações de quais as classes gramaticais utilizadas, em grupos de três palavras no formato de um *array* de três elementos. Esse *array* utiliza como valores, os identificadores das etiquetas presentes no grupo anterior.

Após a criação dos nodos, seus relacionamentos serão cruciais para que as consultas possam retornar informações relevantes para o processo de desambiguação. Novamente os relacionamentos foram agrupados a fim de obter um melhor desempenho devido à indexação. Os relacionamentos possuem os tipos:

- **HAS_A**: Tipo utilizado para definir o relacionamento partindo de um nodo “Palavra” para algum nodo “TagGroup”. Nesta situação, uma palavra pode fazer parte de uma série de trigramas, já que a mesma palavra pode ser encontrada diversas vezes em situações distintas. Desta maneira uma única palavra acaba possuindo muitos relacionamentos deste tipo, conforme pode ser visto na Figura 8. Nessa figura aparece o resultado de uma

consulta no Neo4j, em sua interface web, buscando pela palavra “mas” e seus TagGroup” relacionados, limitando-se a 25 itens de cada tipo.

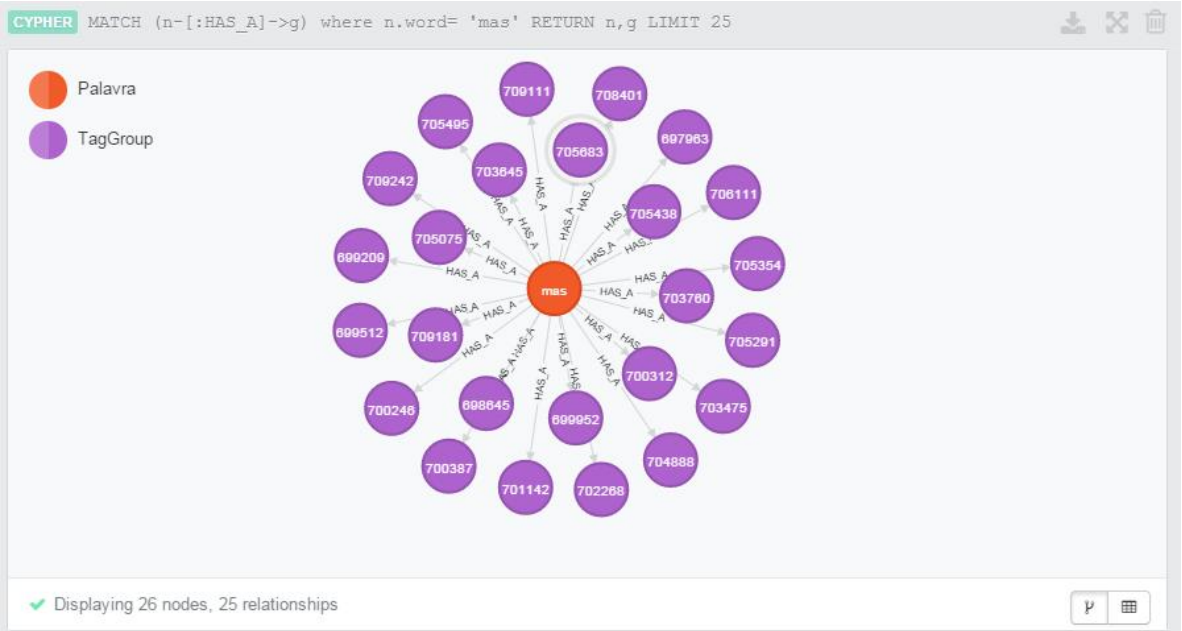

Figura 8 – Resultado em forma de grafo de uma consulta, na ferramenta web do Neo4j.

- **TO:** Tipo utilizado para apresentar o relacionamento entre nodos “TagGroup”, possuindo uma direção de antecessor de sucessor. Assim, o nodo “TagGroup” vinculado à primeira palavra de uma sentença possui um relacionamento “TO” para o “TagGroup” da palavra seguinte, prosseguindo desta maneira até o final da sentença. Desta forma cada sentença possui sua própria lista de nodos do tipo “TagGroup” relacionando-se apenas em seu escopo.

- **FROM:** Tipo que define um relacionamento com orientação contrária entre os nodos “TagGroup”. Com ela a última palavra de uma sentença pode encontrar as informações de suas palavras anteriores de maneira rápida. Seguindo a documentação do Neo4j, essa ligação foi realizada para melhorar o desempenho em consultas que façam uma busca na sentença de maneira reversa.

A Figura 9 apresenta o resultado de uma consulta que busca um conjunto de nodos do tipo “TagGroup” interligados por ambos os relacionamentos “TO” e “FROM” representando duas sentenças distintas. Cabe ressaltar que as sentenças não se interligam, tendo seu contexto formado apenas pelas palavras que constituem essa frase. Limitando-se a quantidade de nodos em vinte e

cinco, pode-se validar a quantidade de relacionamentos existente, sendo quarenta e seis relacionamentos no total, vinte e três do tipo “TO” e a mesma quantidade do tipo “FROM”.

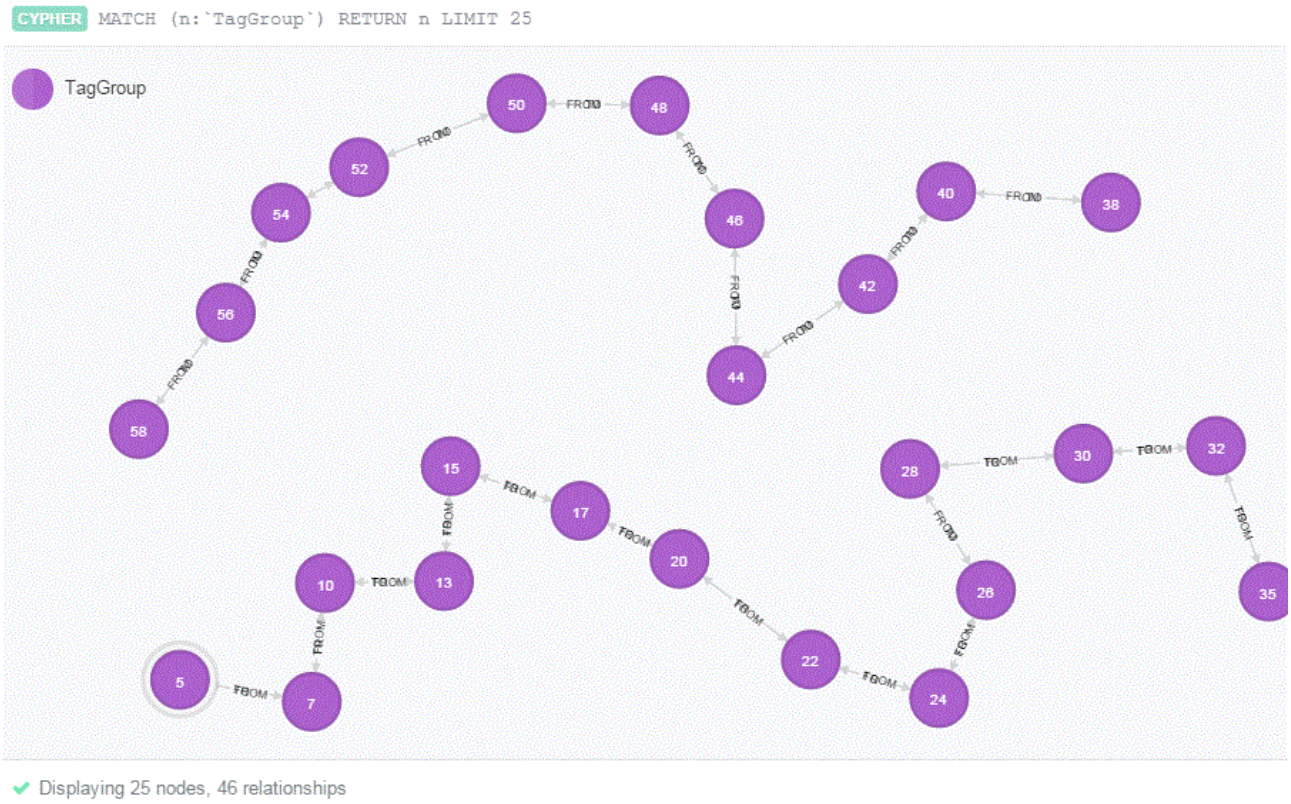

Figura 9 - Conjunto de nodos do tipo "TagGroup" representando duas sentenças

Tendo essa massa de dados finalmente ajustada e processada na ferramenta, pode-se executar o novo script de desambiguação que irá realizar consultas no grafo, a fim de obter informações que colaboram na escolha da melhor etiqueta. A busca começa com a leitura do arquivo, da mesma maneira que os métodos anteriores. No momento em que é lida uma linha onde há uma palavra que possa ser desambiguada, o novo processo começa. Logo antes da consulta ao grafo, é gerada a tríade que informa as palavras vizinhas. Como já comentado anteriormente são utilizados os indicadores de limite de sentença a fim de obter em qualquer situação uma palavra com dois vizinhos. Em conjunto das palavras, obtém-se a lista de etiquetas existentes, adicionadas pelo WAGGER para cada palavra. Ocorrendo situações onde uma definição de limite da sentença (“START” ou “END”) compõe a tríade, ela simplesmente não recebe etiqueta alguma.

Possuindo esses dados, um método que executa a busca no grafo é instanciado, essa busca é realizada através da API de consulta disponível no Neo4j. A busca se dá utilizando como

parâmetro base a palavra em questão, localizada no centro da tríade. Essa consulta retorna uma lista com valores no formato de tríade e um contador de quantas vezes cada item ocorreu no corpus. Essa listagem é ordenada pelo contador, de forma decrescente, deixando a situação que ocorreu em mais oportunidades com maior prioridade. Na figura 10 podemos observar a lista de tríades existentes no corpus e que possuem a palavra "notícia" como membro central. A lista possui um formato definido, usando *aliases* para as etiquetas que são definidas por seus identificadores numéricos (propriedade id de cada nodo). Os *aliases* definidos para as tuplas são:

- **prefix:** Identifica a etiqueta utilizada pelo primeiro elemento da tríade, pode eventualmente possuir o valor delimitador de sentença "START".

- **actual:** Aponta a etiqueta existente para a palavra utilizada como parâmetro na consulta, é neste local que as opções de etiqueta são obtidas.

- **suffix:** Armazena o identificador da etiqueta subsequente à palavra pesquisada. Em alguns casos possui o valor delimitador de sentença "END".

- **counter:** Apresenta a quantidade de vezes que determinado grupo de etiquetas aconteceu no *corpus*. Baseado nesse valor, a lista é ordenada a fim de permitir uma avaliação daquilo que mais aconteceu de maneira prioritária.

O conjunto completo de dados é retornado para avaliação posterior, embora seja uma consulta indexada, um cache tendo a palavra buscada como índice é salvo em memória para evitar buscas repetidas ao grafo.

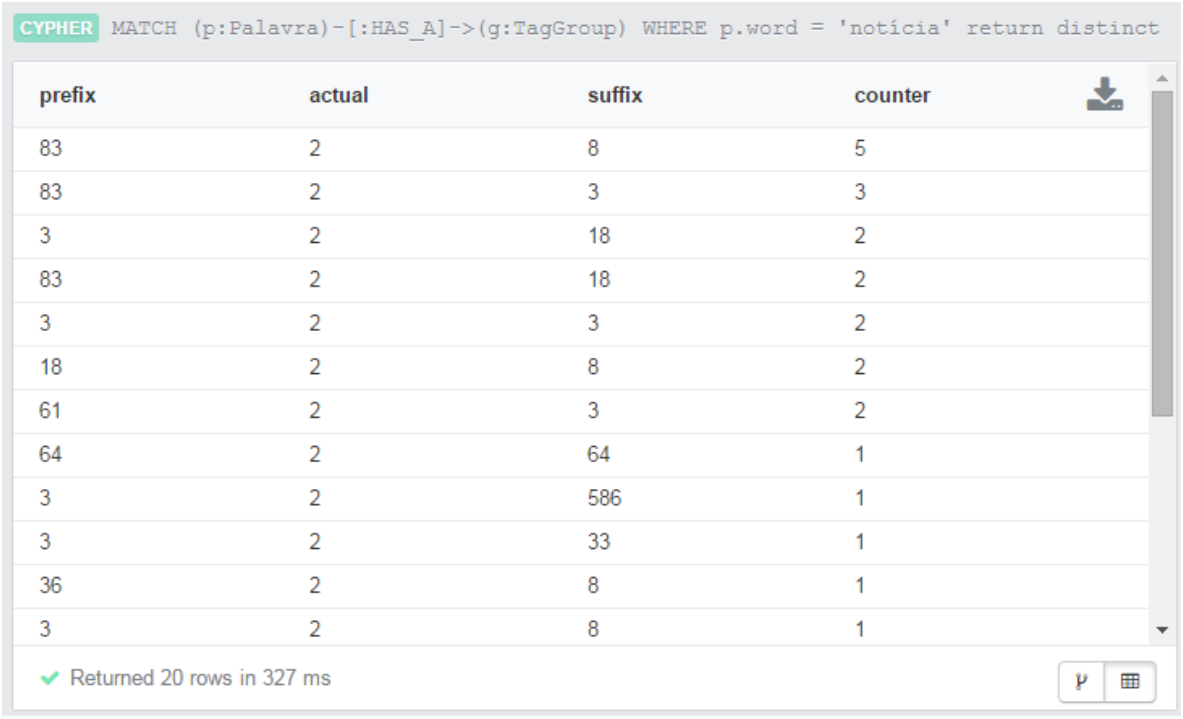

Figura 10 - Resultado da consulta pela palavra "notícia", retornando as tríades encontradas.

Com a lista de possibilidades encontradas no grafo, consequentemente no corpus o método pode finalmente desambiguar. Para que seja relevante a quantidade de vezes que cada grupo de etiquetas aconteceu, o processamento ocorre percorrendo a lista ordenada, desambiguando pela primeira possibilidade que for avaliada com sucesso. Assim sendo, para cada item da lista, as avaliações acontecem, em uma ordem pré-definida. Dentre as diversas possibilidades de utilização das informações foram definidas três formas distintas avaliação. Estas possibilidades são descritas como:

- ***Full-Match***: Definida quando a palavra a ser desambiguada encontrou no corpus alguma situação exatamente igual à atual. Isso significa que o corpus possui a situação onde as três palavras estão etiquetadas pelo mesmo conjunto de etiquetas existente na tríade atual. Exemplo: O corpus possui a tríade: “O menino caiu [...]”. Sabendo que esse conjunto é composto por um artigo seguido de um substantivo e um verbo, ocorrerá “*Full-Match”* se a tríade “Um menino subiu” for desambiguada.

- ***Prefix-Only***: Ocorre quando apenas a palavra atual e seu antecessor possuem as mesmas tags encontradas na listagem obtida após consulta ao grafo. Essa opção é avaliada após a avaliação de "*Full-Match"*. Nesta situação o último elemento da tríade é ignorado.

- ***Suffix-Only***: Acontece após a validação da "*Prefix-Only*", sendo avaliada com sucesso quando a palavra atual e sua vizinha subsequente possuem a mesma dupla de tags encontrada no corpus.

No momento em que uma das três avaliações é realizada com sucesso, a etiqueta utilizada pela palavra é definida, sendo esta a mesma encontrada no *alias* "*actual*" da tupla usa pela análise. Em casos de não encontrar informações de uma determinada palavra no grafo, o código realiza uma desambiguação calculada, descrita na seção anterior como plano alternativo. A visão de todas as palavras e o inter-relacionamento de suas etiquetas aliada à possibilidade de consultar essa massa de dados de maneira dinâmica propicia um desempenho superior às soluções anteriores como é mostrado na seção 4.2.

4.2 Avaliação e Análise

A análise da ferramenta proposta será apresentada levando-se em consideração sua própria evolução em relação aos resultados obtidos durante seus experimentos. Os valores encontrados na execução de testes mostram uma curva ascendente de melhoria, aumentando o percentual de acerto para etiquetas desambiguadas. Os três experimentos foram testados diversas vezes, no entanto para a obtenção dos valores apresentados a seguir foi idealizada uma bateria de testes.

A fim de proporcionar um ambiente de comparação justo, todos os testes foram executados no mesmo equipamento, um Notebook Latitude E7440, possuindo um processador Intel Core I5 4300U (1,9 GHz), 8 GB de memória RAM utilizando Windows 8.1 de 64 bits. Cada experimento foi executado três vezes com os mesmos parâmetros, em seguida foi calculada uma média simples entre os resultados para obter os valores finais de cada método de desambiguação.

4.2.1 Avaliação de Desambiguação Aleatória

Por se tratar de uma primeira versão da solução, utilizando um algoritmo simples é esperado um percentual de acerto na desambiguação relativamente mais baixo que os demais. Entretanto, mesmo simples essa solução traz o código que é usado como base para as gerações seguintes, sendo também nesse momento que o comparador de textos desambiguados com o corpus anotado foi

testado realmente. Esse comparador foi criado em Groovy para ser o código que verifica as etiquetas desambiguadas com as corretas, originalmente anotadas no corpus em seu formato original. Este é o método utilizado para a obtenção de percentuais de acerto de cada aplicação deste trabalho.

A desambiguação aleatória usa como única base dados o arquivo texto criado pelo WAGGER conforme citado na seção 4.1.1. Desta maneira os resultados não conseguem ultrapassar as possibilidades apontadas durante a etiquetagem, limitando-se à lista de etiquetas possíveis para cada palavra presente no texto.

A bateria de testes, em três etapas realizadas permitiu o cálculo do valor mediano entre as desambiguações obtidas. Os resultados são apresentados com seus valores percentuais de cada execução bem como o valor obtido na média final. O universo de dados encontrado no arquivo do WAGGER contém cerca de 156600 palavras únicas. Essas informações podem ser conferidas na Figura 11.

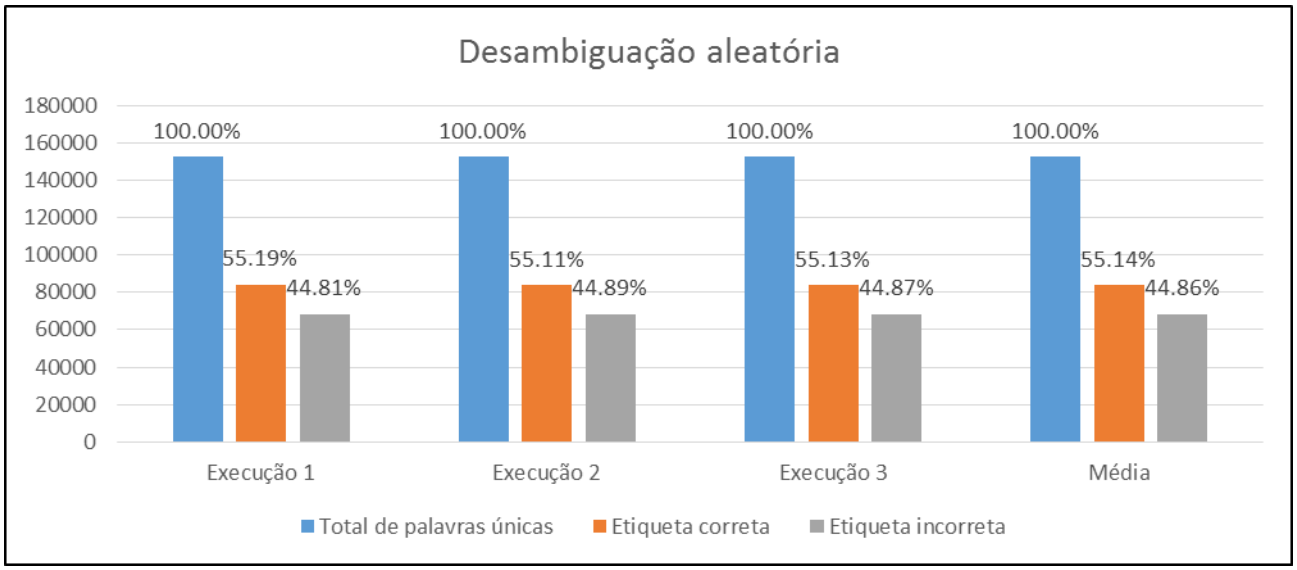

Figura 11 - Gráfico com os resultados obtidos através da desambiguação aleatória

Conforme já esperado, o resultado desse primeiro experimento ficou muito aquém do ideal percentual de acertos de um desambiguador. No entanto a estrutura base de código utilizada nos testes bem como na criação dessa versão se provou útil para o prosseguimento dos experimentos.

4.2.2 Avaliação de Desambiguação Ponderada

Resultado do segundo experimento, a solução que utiliza o histórico de etiquetas encontrada no corpus e baseado nele efetua cálculos probabilísticos surgindo como uma evolução da implementação original. Neste formato além da listagem das *tags* provenientes do WAGGER temos uma inserção proporcional de algumas etiquetas extras, a fim de melhor abrangência, inserindo uma nova possibilidade. Desta maneira, palavras com apenas uma etiqueta, ganham novas possibilidades em um pequeno percentual que será também agregado durante o processo de desambiguação.

Executando os mesmos passos durante a bateria de testes, essa solução foi executada por três vezes utilizando os mesmos parâmetros em todos os casos. Nesta situação as diferenças que podem acontecer devem-se ao fato de que alguma das possibilidades de etiquetas inseridas, com um pequeno percentual, fosse escolhida. Outra situação é a de palavras que não possuam informações no conjunto de estatísticas, isso ocorre já que o WAGGER pode eventualmente separar palavras de uma maneira particular que provoca, em último caso, uma desambiguação através de aleatoriedade.

Seguindo o formato de apresentação de dados usado anteriormente, foi calculada a média das execuções de forma a complementar o resultado. A Figura 12 apresenta os resultados demonstrando que a utilização do histórico de uso das palavras contribui para melhores percentuais de acerto.

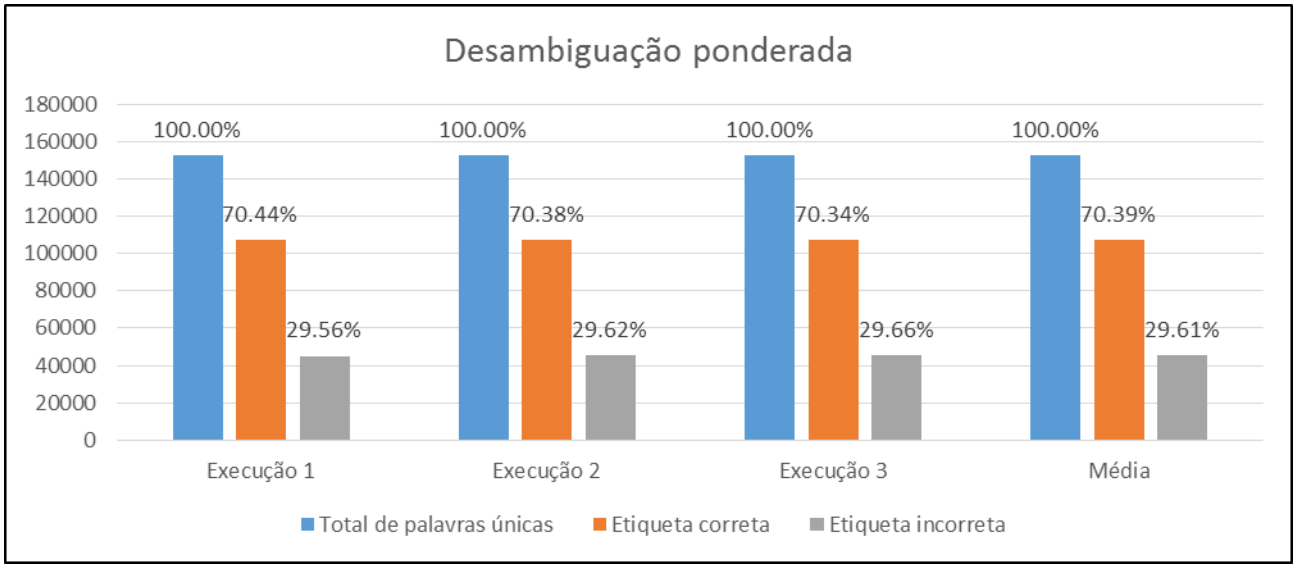

Figura 12 - Resultados obtidos através da desambiguação ponderada, usando histórico.

Embora a melhoria seja evidente, a proposta ainda se mostra com valores distantes dos métodos de melhor desempenho. Por ser uma solução que não se importa com o contexto da palavra, focando-se unicamente no uso desta ignorando sua localização na sentença o que acarreta desvantagens como a limitação de acertos e a execução de deduções equivocadas onde o conhecimento contextual poderia evitar.

4.2.3 Avaliação de Desambiguação Ponderada Através de Estatística

A última avaliação acontece juntamente com a análise da solução final proposta nesse trabalho, esta solução consegue utilizar tanto desambiguação calculada como análise de contexto. Como já comentado anteriormente é nessa solução que ocorre a busca no banco de dados em formato de grafo do Neo4j, procurando por informações de contexto toda a vez que uma palavra deve ser desambiguada.

Esse teste também utilizou o mesmo grupo de dados e foi repetido para obter-se um valor final da média de desempenho. Como esperado, por possuir mais formas de obter informações no momento de desambiguar, essa solução conseguiu obter um desempenho melhor que a anterior. Embora não seja um valor similar aos obtidos em trabalhos similares, a validade desse formato de desambiguação pode ser atestada com um acerto percentual médio de cerca de 70%.

Levando em consideração que essa solução, por prover uma base dados facilmente extensível, pode no futuro receber outros *corpus*, melhorando seu poder de análise e incrementando consequentemente sua capacidade de acerto. Ainda essa proposta pode ser eventualmente montada para trabalhar em cluster, utilizando processamento paralelo para que uma grande massa de dados possa ser depurada com velocidade reduzida.

Como problemas comuns encontrados, principalmente os validados na comparação entre *tags* existentes temos as diferenças de tratamento das palavras entre o etiquetador e o corpus. Mesmo tendo o mapeamento entre as *tags*, por diversas oportunidades o formato diferenciado de separação de fragmentos em palavras diferentes do WAGGER acaba por tornar palavra e consequentemente a etiqueta diferente daquela esperada pelo corpus anotado corretamente. Exemplos disso são palavras compostas que utilizam hífen, que no corpus original são anotados como apenas uma entidade, entretanto é separada em uma dupla de palavras pelo WAGGER podendo cada uma ter uma etiqueta distinta.

Situações semelhantes acontecem em casos como “PUC/SP” onde o etiquetador WAGGER compreende “PUC” e “SP” como palavras distintas removendo “/” de seu resultado final. Outro evento semelhante acontece no caso de valores monetários, eles são novamente separados em cada vírgula encontrada. No entanto essa situação dificilmente compromete o acerto, pois costumam todos os lemas relacionados serem do tipo numeral.

5 CONCLUSÃO

Neste Capítulo serão apresentadas as contribuições e conclusões obtidas nessa dissertação. Além disso, também serão apresentados trabalhos futuros que venham a complementar o processo proposto ou que possam fazer uso de alguma parte do código criado.

5.1 Contribuições

O objetivo central dessa dissertação foi propor um método de desambiguação de classes gramaticais encontradas nos textos em língua portuguesa anotados. Para isso, diferente dos trabalhos encontrados na literatura, foi utilizado um processo de melhoria iterativa de uma solução inicial com algoritmo mais simples, evoluindo para um software capaz de importar informações externas. O software se mostrou capaz de observar estatísticas históricas do *corpus* estudado bem como construir uma base de conhecimento relativa ao contexto das palavras desses textos. Como produto do processo proposto foram desambiguadas de forma automática milhares de palavras encontradas pelo etiquetador WAGGER, que por sua vez apenas tinha informado um grupo de etiquetas relacionando as classes gramaticais prováveis destes termos.

O objetivo foi alcançado e o processo foi aplicado observando uma melhoria contínua entre as soluções propostas de acordo com a sua evolução. Sobre os resultados das aplicações foram realizadas três avaliações que tinham como objetivo verificar a taxa de acerto para a desambiguação das palavras usando como base de retidão as etiquetas existentes originalmente no Mac-Morpho.

Conforme pode ser visto através da análise dos resultados obtidos pelos desambiguadores criados, o processo proposto nessa dissertação definiu com correção as classes gramaticais da maior parte das palavras encontradas nos textos analisados. Principalmente em relação à última proposta apresentada, há espaço para melhorias, tanto no mapeamento de etiquetas utilizadas pelo WAGGER com aquelas presentes em outros corpora, bem como na expansão da base de conhecimento. Esta pode receber informações em maior quantidade a fim de ser capaz de informar um número maior de contextos.

O foco dessa dissertação é identificar, desambiguando e definindo, apenas classes gramaticais encontradas nas palavras etiquetadas pelo WAGGER. Levando em consideração que o etiquetador utilizado possui um grupo de etiquetas próprio, um mapeamento entre os diferentes tipos utilizados entre *corpora* foi realizado. Outra tarefa foi a inserção de lógicas que visam

automatizar essas definições. Dito isto, pode-se concluir que foi proposto um processo relevante para desambiguação automática de classes gramaticais. A Figura 14 apresenta uma visão geral dos resultados obtidos nas propostas em um único grafo que demonstra a evolução obtida em cada fase. Uma continuação nessa pesquisa pode continuar a trajetória de melhoria ao seguir alguma das ideias de melhoria apresentadas na seção 5.2.

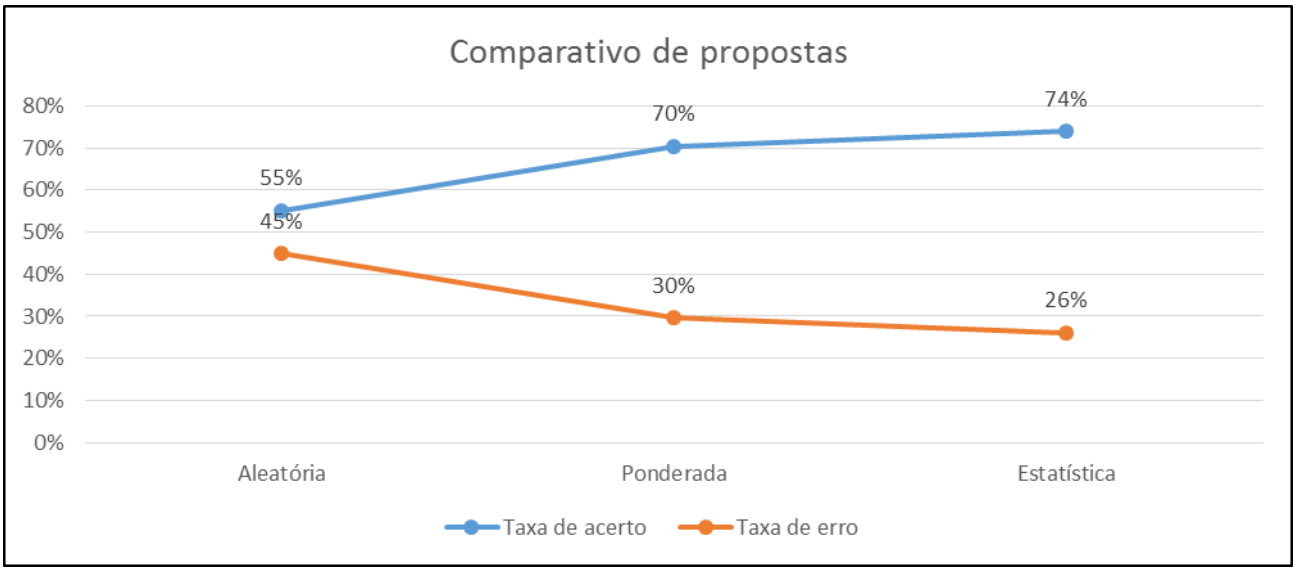

Figura 13 – Gráfico mostrando as taxas médias de acerto e erro das propostas deste trabalho.

Dentre as contribuições dessa dissertação pode-se citar o processo para desambiguação automática de classes gramaticais em corpus da língua portuguesa etiquetados pelo WAGGER. Conforme já citado, assim como o etiquetador, essa proposta pode ser facilmente portada para outros idiomas. Ainda nesse contexto, nenhuma das propostas encontradas na literatura utiliza como fonte de dados os conceitos e estatísticas advindas de uma massa de dados em forma de grafo estruturado.

Ainda como contribuição dessa dissertação, pode ser citada a ferramenta para criação de grafos orientados na solução baseada em Neo4j que pode ser utilizada para adicionar novos *corpora* aumentando a quantidade de palavras conhecidas e seus inter-relacionamentos. Essa possibilidade de fácil escalabilidade utilizando produtos focados na indústria acaba sendo o maior diferencial dessa proposta, onde máquinas de alto desempenho com múltiplos dicionários integrados poderiam analisar uma grande quantidade de informações em tempos aceitáveis.

Cabe ressaltar que o trabalho não conseguiu atingir o nível de excelência na taxa de acerto de trabalhos encontrados na literatura. Algumas possiblidades daquilo que pode ajudar a explicar esse desempenho são:

- Dicionário não completamente adaptado ao corpus utilizado houve uma relevante perda de assertividade em palavras que não puderam ser reconhecidas.
- Solução de busca em grafo e análise de cálculo ponderado não ter sofrido uma bateria de ajustes finos a fim de encontrar valores de parâmetros que pudessem atrair mais resultados corretos.
- Inserção de 10% de possibilidade de etiquetas mais comuns, mesmo se não existentes na lista encontrada pelo etiquetador.
- Não utilização de um conjunto de regras pré-definidas no idioma, item encontrado nas soluções com melhor desempenho dentro da literatura.

Embora não obtendo um desempenho compatível com seus similares na literatura, este trabalho contribuiu como uma proposta de utilização de modelos de dados em forma de grafo como repositório de informações relevantes para a desambiguação de anotações morfossintáticas. Essa abordagem não foi encontrada em outros trabalhos. A proposta trouxe consigo a possibilidade de utilizar ferramentas de alto desempenho que podem ter sua utilização facilmente escalável, abrindo possibilidades de uso em clusters.

5.2 Trabalhos Futuros

Os trabalhos futuros dessa dissertação podem ser divididos em quatro eixos de pesquisa: (i) aplicação de um conjunto de regras na etapa de desambiguação; (ii) aprimoramento do processo de inserção de relacionamentos na base dados e seus métodos de consulta; e (iii) construção de um desambiguador que preencha a base dados adicionando novas informações automaticamente. (iv) criar um formato de análise de contexto levando em consideração mais palavras vizinhas.

Dentro do eixo de aplicação de um conjunto de regras, podem ser exploradas as aplicações de regras estritamente focadas no idioma a ser desambiguado. No caso da língua portuguesa, existem diversos exemplos onde o uso de regras relacionadas à gramática pode aumentar a taxa de acerto durante a definição de uma etiqueta.

No eixo de aprimoramento do processo de relacionamentos na base dados e seus métodos de consulta destaca-se a possibilidade de criar novas maneiras de popular esse conjunto de dados. Ao criar métodos de busca e inserção de notícias da internet, por exemplo, resultaria em uma grande massa de informações que permitiria uma melhor cobertura dos termos utilizados cotidianamente pelos portais de informação. Outra grande melhoria que pode ser estudada é aprimorar os códigos de consulta utilizados na base em forma de grafo, bem como uma eventual mudança no formato desta base agregando diferentes tipos de nodos e relacionamentos que possam ser úteis no momento de desambiguar. Assim como consultas SQL utilizadas em bancos de dados relacionais, as *queries* utilizadas no presente trabalho são passíveis de ajuste fino também conhecido como "*tunning*", a fim de obter retornos de suas buscas com maior rapidez. O processo de normalização de domínio em uma estrutura de grafo também pode ser depurado, modificando assim o acesso à informação e permitindo, eventualmente, uma melhora em seu desempenho.

Estendendo ainda mais as atribuições do desambiguador, ele poderia, quando encontrar palavras desconhecidas, agrupá-las durante este processo. Com o fim da desambiguação essas palavras poderiam ser consultadas em diferentes dicionários online, que poderiam oferecer definições. Essas definições após uma validação humana poderiam ser inseridas no dicionário do etiquetador, bem como no banco de dados em formato de grafo, adicionando também os relacionamentos no caso destes existirem.

Expandindo o contexto utilizado nas pesquisas para avaliar mais do apenas uma palavra de cada lado durante a análise, seria a quarta opção de expansão deste trabalho. Esse aumento de visão para cada par de palavras ou mesmos três palavras de cada lado durante uma análise poderia permitir uma busca mais específica no grafo. A forma de inserção de dados no grafo também mudaria, eventualmente criando novos formatos de nodo que colaborariam em busca de uma melhor acurácia.

5.3 Considerações finais

Neste trabalho foi criado um desambiguador de anotações morfossintáticas em textos de língua portuguesa. Como base ferramental na análise do texto foi utilizado o parser WAGGER para, em seguida, utilizar algoritmos construídos utilizando Groovy em conjunto com um repositorio de dados em forma de grafo, graças ao uso do Neo4J. Um futuro pesquisador pode utilizar o presente trabalho como base, integrando mais funcionalidades e expandindo sua capacidade e acuidade.

REFERÊNCIAS BIBLIOGRÁFICAS

[3] Aduriz, I.; Illarraza, D. A. "Morphosyntactic disambiguation and shallow parsing in computational processing of Basque". In: Inquiries into the lexicon-syntax relations in Basque, 2004, pp. 1-23.

[27] Bahl L.; Mercer R. "Part-Of-Speech assignment by a statistical decision algorithm". In: Symposium on Information Theory, 1976, pp. 88-89.

[30] Bayes T. "An essay towards solving a problem in the doctrine of chances", Phil. Trans. of the Royal Soc. of London, vol. 53, Jan 1763, pp. 370-418.

[16] Benello J.; Mackie A. W.; Erson J. A. "Syntactic category disambiguation with neural networks", Computer Speech and Language, vol. 3, Abr 1989, pp. 203–217.

[18] Black E.; Jelinek F.; Lafferty J.; Mercer R.; Roukos S. "Decision tree models applied to the labeling of text with parts of speech". In: Proceedings of the DARPA Speech and Natural Language Workshop, 1992, pp. 117-121.

[12] Brants, T. "TnT - a statistical part-of-speech tagger". In: Proceedings of the Sixth Applied Natural Language Processing Conference, 2000, pp. 224-231.

[4] Brill E. "Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging". In: Natural Language Processing Using Very Large Corpora. Kluwer Academic Press, 1995, pp. 1-13.

[13] Brill, E. "A simple rule-based part of speech tagger". In: Proceedings of the Third Conference on Applied Natural Language Processing, 1992.

[10] Church, K. W. "A stochastic parts program and noun phrase parser for unrestricted text". In: Proceedings of the Second Conference on Applied Natural Language Processing, 1988, pp.136–143.

[11] Cutting D.; Kupiec J.; Pedersen J.; Sibun P. "A practical part-of-speech tagger". In: Proceedings of the Third Conference on Applied Natural Language Processing, 1994, pp. 133–140.

[14] Daelemans W.; Zavrel J.; Berck P.; Gillis S. "Mbt: A memory-based part of speech tagger-generator". In: Proceedings of the Fourth Workshop on Very Large Corpora, 1996, pp. 14–27.

[29] Derose S. "Grammatical Category Disambiguation by Statistical Optimization", Computational Linguistics, vol. 14, Fev 1988, pp. 31-39.

[34] Domingues, M. L. C. S. "Abordagem para o desenvolvimento de um etiquetador de alta acurácia para o Português do Brasil", Tese de Doutorado, Instituto de Tecnologia - Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Pará, 2011, 140p.

[38] ECMA International. "The JSON Data Interchange Format". Capturado em < http://www.ecma-international.org/publications/standards/Ecma-404.htm>, Agosto 2014.

[6] Fernandes P.; Lopes L.; Prolo C. A.; Sales A.; Vieira R. "A Fast, Memory Efficient, Scalable and Multilingual Dictionary Retriever". In: LREC, 2012, pp. 2520-2524.

[5] Giménez J.; Màrquez L. "Svmtool: A general pos tagger generator based on support vector machines". In: Proceedings of the 4th International Conference on Language Resources and Evaluation, 2004, pp. 43-46.

[20] Greene B.; Rubin G. "Automatic grammatical tagging of English", Technical report, Department of Linguistics - Brown University, 1971.

[24] Jurafsky D.; Martin J. "Speech and Language Processing: An Introduction to Natural Language Processing". Computational Linguistics, and Speech Recognition, Prentice Hall, 2008, 934p.

[25] Karlsson F. "Constraint Grammar as a Framework for Parsing Unrestricted Text". In: Proceedings of the 13th International Conference of Computational Linguistics, vol. 3, 1990, pp. 168-173.

[22] Klatt, S. "Combining a rule-based tagger with a statistical tagger for annotating German texts". In: KONVENS, 2002.

[33] Baum, L. E. "An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process", Inequalities, vol. 3, Fev 1972.

[8] Manning C.; Schütze H. "Foundations of Statistical Natural Language Processing", MIT Press, 1999, 680p.

[19] Màrquez L.; Padró L. "A flexible POS tagger using an automatically acquired language model". In: Proceedings of the 35th Annual Meeting of the ACL, 1997, pp. 238–245.

[23] Martins, P. L. M. "Desambiguação Automática da Flexão Verbal em Contexto", Dissertação de Mestrado, Lisboa University - Portugal, 2008, 96p.

[31] Nakagawa T.; Uchimoto K. "A hybrid approach to word segmentation and pos tagging". In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, 2007, pp. 217–220.

[17] Nakamura M.; Maruyama K.; Kawabata T.; Shikano K. "Neural network approach to word category prediction for English texts". In: Proceedings of the 13th International Conference on Computational Linguistics COLING 1990, 1990, pp. 213–218.

[36] Neo Technology. "Neo4j". Capturado em < http://neo4j.com/>, Agosto 2014.

[37] NILC - USP. "Mac-Morpho". Capturado em < http://www.nilc.icmc.usp.br/macmorpho/>, Agosto 2014.

[35] Pivotal. "Groovy Language". Capturado em <http://groovy-lang.org/>, Agosto 2014.

[15] Ratnaparkhi A. "A maximum entropy model for part-of-speech tagging". In: Proceedings of Conference on Empirical Methods in Natural Language Processing. University of Pennsylvania, 1996.

[9] Schmid, H. "Part-Of-Speech Tagging Reference". In: European Summer School in Logic, Language and Information. Capturado em: <http://www.coli.uni-sa ar lan d.de/~schulte /Tea ching/ESSLLI-06/Referenzen/Tagging/schmid-hsk-tag.pdf>, Dezembro 2013.

[7] Segond F.; Schiller A.; Grefenstette G.; Chanod J. "An Experiment In Semantic Tagging Using Hidden Markov Model Tagging". In: ACL/EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications, 1997, pp. 78-81.

[2] Silva, J. R. "Shallow Processing of Portuguese: From Sentence Chunking to Nominal Lemmatization", Dissertação de Mestrado, Lisboa University, Portugal, 2007, 194p.

[1] SINTEF. "Big Data, for better or worse: 90% of world's data generated over last two years" Capturado em: <http://www.sciencedaily.com/releases/2013/05/130522085217.htm>, Dezembro 2013.

[26] Stolz W.; Tannenbaum P.; Carstensen F. "A Stochastic approach to the grammatical coding of English". Communications of the ACM, 1965, pp. 399-405.

[21] Tapanainen P.; Voutilainen A. "Tagging accurately - don’t guess if you know". In: Proceedings of the 4th Conference on Applied Natural Language Processing, 1994, pp. 47–52.

[28] Viterbi A. J. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". IEEE Transactions on Information Theory, vol. 13, 1967, pp. 260–269.

[32] Stewart, W. J. "Probability, Markov Chains, Queues, and Simulation". Princeton University Press, 2009, 776p.