CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA CELSO SUCKOW DA FONSECA
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE PESQUISA
COORDENADORIA DE PESQUISA E ESTUDOS TECNOLÓGICOS

RELATÓRIO FINAL DE INICIAÇÃO CIENTÍFICA

PROCESSAMENTO DE LINGUAGEM NATURAL EM TEXTOS DA FARMACOVIGILÂNCIA

Edital PIBIC 2018

Aluno:

André Alves de Araújo

Ciência da Computação / 7º período

Bolsista CEFET/RJ.

Professores Orientadores:

Gustavo Paiva Guedes, Doutorado

Kele Teixeira Belozze, Doutorado

Rio de Janeiro, RJ - Brasil

08 / 2019

Rev: 09/2018

Resumo. *A farmacovigilância é uma área médica caracterizada pela detecção de eventos adversos em produtos farmacêuticos. Uma das técnicas utilizadas para detectá-los é o uso do processamento de linguagem natural em textos não estruturados da área médica. Um software que faz o processamento de linguagem natural na área médica é o Metamap. Esse software faz o mapeamento das palavras de um texto de entrada em conceitos médicos presentes no sistema médico UMLS. Entretanto, ele apenas funciona com textos em inglês, dado que mapeia os conceitos de vocabulários apenas na língua inglesa do UMLS. No presente trabalho explicamos as principais características do Metamap e apresentamos um modelo de dados para ser usado pelo servidor do Metamap, de maneira a permitir o mapeamento de textos em português e retornar conceitos dos vocabulários em português do UMLS. Esse modelo foi criado ao usar o Metamap Data File Builder. Apresentamos o uso do modelo de dados em uma aplicação desenvolvida na linguagem de programação Java, cuja entrada são textos em português. Isso foi possível ao usar as bibliotecas disponíveis do Metamap Java API. Os resultados indicam que o uso do nosso modelo tem muitas limitações. Dentre elas, o nosso modelo não permite o mapeamento de palavras com caracteres especiais fora da tabela ASCII e tem pouca possibilidade de mapear variações gramaticais. Conclui-se que o modelo de dados proposto precisa ser melhorado para poder auxiliar na farmacovigilância.*

1. Introdução

A farmacovigilância é a detecção, monitoramento, caracterização e predição de efeitos adversos em produtos farmacêuticos, também chamados de eventos adversos relacionados a medicamentos (EAM), com o objetivo de reduzir sua incidência e severidade [Luo et al., 2017]. As principais fontes de dados que contêm as EAM são literatura biomédica, relatórios periódicos, dados encontrados nas mídias sociais e os registros eletrônicos de saúde [Wong et al., 2018]. A fim de automatizar o processo de detecção de EAM nessas fontes são utilizados sistemas de processamento de linguagem natural (PLN) [Wong et al., 2018].

O Metamap é um programa de PLN que identifica palavras e frases de um texto não estruturado e as mapeia nos conceitos médicos presentes no sistema médico chamado *Unified Medical Language System* (UMLS) [Chiaramello et al., 2016]. O servidor do Metamap, por padrão, apenas mapeia textos em inglês para conceitos médicos em inglês [Metamap_FAQ]. Isso ocorre pois ele, por padrão, utiliza modelos de dados (*USABase, NLM, Base*) que contém apenas vocabulários do UMLS na língua inglesa [Metamap_FAQ].

Nesse cenário, o presente trabalho apresenta um novo modelo de dados que permite o mapeamento de textos em português em conceitos presentes em vocabulários em português do UMLS. Também exemplifica como esse modelo é utilizado tanto no servidor quanto em uma aplicação em Java e expõe suas principais limitações. Por fim, discute se é possível usar o Metamap com esse modelo para auxiliar na farmacovigilância, tendo como entrada *posts* de mídias sociais.

Além dessa introdução, o presente trabalho é separado em 5 seções. A seção 2 apresenta as principais características do Metamap e do novo modelo de dados. A seção 3 apresenta os resultados do uso do modelo com exemplos. A seção 4 discute sobre o

modelo e seu uso na farmacovigilância. Por fim, a seção 5 apresenta as considerações finais.

2. Metodologia

Nessa sessão, primeiramente explica-se sobre a relação do Metamap e o UMLS. Em seguida são descritas as etapas do processo de mapeamento do Metamap. Então explico sobre os modelo de dados do Metamap e necessidade de um criar um novo modelo de dados. E no final é apresentado o uso do novo modelo em uma aplicação em Java.

2.1. Metamap e UMLS

O UMLS é um sistema que proporciona dados para auxiliar no desenvolvimento de sistemas que agem como se entendessem o significado da linguagem presente na biomedicina e na saúde [UML_tutorial]. Ele é o sistema que fornece ao Metamap informações médicas necessárias para seu processamento. Ele consiste de três fontes de conhecimento, presentes na figura 1, que podem ser usadas separadas ou em conjunto [UML_tutorial].

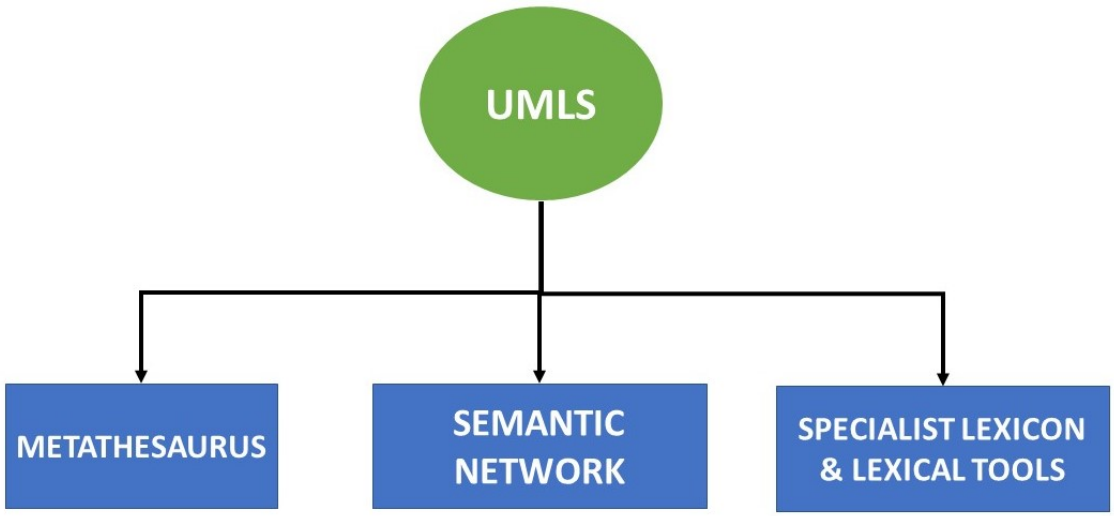

Figura 1. Fontes de Conhecimento do UMLS [UML_tutorial]

O *Metathesaurus* é um amplo e multilíngue conjunto de dados de vocabulários que contém informações de conceitos específicos da área da biomedicina e da saúde [UML_tutorial]. O *Metathesaurus* é contruído a partir de versões eletrônicas de classificações, lista de termos controlados usados na assistência de pacientes, em estatísticas de saúde pública e outras fontes de vocabulário biomédicos [UML_tutorial].

Cada palavra que pertence ao *Metathesaurus* é chamada de *string Metathesaurus* e ela tem um identificador único e pertence a pelo menos uma fonte de vocabulário e pode estar relacionada a outras *strings* [UML_tutorial]. Como tem muitas *strings* que são parecidas na escrita e significado, é definida uma *string* principal, preferida e importante chamada conceito [UML_tutorial]. Por exemplo têm-se as *strings* “Dor” do vocabulário genérico A, “DOR” do vocabulário B e “dor” do vocabulário C. A partir delas é escolhido o conceito ou *string* preferida que é a “DOR” do vocabulário B.

O *Semantic Network* é uma rede que fornece a categorização consistente em assuntos de alto nível, chamados de grupos semânticos, dos conceitos dos *Metathesaurus*, e

fornece o conjunto de relações úteis entre esses grupos semânticos [UML_tutorial]. Isso permite rotular o domínio biomédico [UML_tutorial]. No exemplo: *Clinical Drug treats Disease or Syndrome* os grupos semânticos são *Clinical Drug* e *Disease or Syndrome* e a relação é *treats* [UML_tutorial].

O *SPECIALIST Lexicon & Lexical Tools* é dividido em um léxico e as ferramentas de léxico. O léxico *SPECIALIST* é um dicionário em inglês que inclui termos biomédicos, assim como palavras comuns nessa língua [UML_tutorial]. E para cada palavra no dicionário tem informação sintática, morfológica e ortográfica [UML_tutorial]. As ferramentas de léxico são um conjunto de programas designados para auxiliar o PLN [UML_tutorial]. Elas são o gerador de variante léxico (LVG), gerador de string normalizada (Norm), gerador de índice de palavra (Wordind) [UML_tutorial].

O Metamap está relacionado ao UMLS da seguinte maneira. No processamento do Metamap ele identifica quais *strings Metathesaurus* estão relacionadas ao texto de entrada, avalia elas e retorna os conceitos médicos que mapeiam da melhor forma o texto de entrada [Aronson, 1996]. Na saída retorna informações de cada conceito como seu grupo semântico, o vocabulário a que pertence, qual a sua *strings Metathesaurus* principal entre outros. Além disso numa das etapas desse processamento é utilizado o Léxico *SPECIALIST* na etapa de geração de variantes que será explicado adiante [Aronson, 1996].

2.2. Processamento teórico do Metamap

Como dito anteriormente o objetivo do Metamap é identificar os conceitos médicos pertencentes a *UMLS* que mapeiam da melhor forma o texto de entrada [Liu and Chen, 2015]. Esse processo de mapeamento pode ser dividido nas etapas da figura 2 [Aronson, 1996].

Figura 2. Algoritmo do Metamap[Aronson, 1996]

Na primeira etapa é o pré-processamento na qual são definidas as tags sintáticas das palavras do texto de entrada [Aronson, 1996]. Em seguida tem a análise sintática das tags e expansão desse resultado com os *tokens* [Aronson, 1996]. No final ocorre a limpeza de informações dessa análise expandida, e isso resulta em palavras que são então correspondidas as do texto de entrada [Aronson, 1996].

Essa correspondência é necessária para a etapa de geração de variantes, que são todas as variações flexionadas e ortográficas das palavras de entrada [Aronson, 1996]. Primeiro são definidos os geradores de variantes, que são sequências de palavras que ocorrem no léxico do *SPECIALIST* do UMLS, a partir do texto de entrada [Aronson, 1996]. Então para cada gerador são produzidos todas as variantes gramaticais possíveis assim como seus sinônimos, acrônimos e abreviações [Aronson, 1996].

Em seguida têm-se a etapa de busca de candidatos em que são recuperadas todas as *strings Metathesaurus* do *UMLS* com suas informações principais (como ID, grupo semântico, nome do conceito) que começam com uma das variações criadas na etapa anterior [Aronson, 1996]. Posteriormente na etapa de avaliação dos candidatos é calculado o *score* de cada candidato encontrado. *Score* é um valor numérico que define se o candidato está mapeando bem o texto original [Aronson, 1996].

Na fórmula do *score* são utilizados os parâmetros a seguir *Centrality (CE), Variation (VA), Coverage(CO), Cohesiveness(CH)* [Aronson, 2001]. Existe também o *Involvement(IV)* que existe nos casos em que a ordem das palavras de entrada não importa, caso isso ocorra esse parâmetro fica no lugar do *Coverage* e *Cohesiveness* [Aronson, 2001]. Todas os parâmetros estão explicados na tabela 1.

Tabela 1. Parâmetros para o cálculo do Score

Paramêtro	Explicação
Centrality	Valor que indica se o candidato contém escrito o núcleo da frase original [Aronson, 2001].
Variation	Valor que indica a diferença entre as variantes do candidato e as palavras da frase original [Aronson, 2001].
Coverage	Valor que indica a cobertura do candidato na frase original [Aronson, 2001].
Cohesiveness	Valor que indica quão suave o candidato cobre a frase original [Aronson, 2001].
Involvement	Valor alternativo a Cobertura e Coesão que indica quanto a frase está envolvida na correspondência entre a frase e o candidato [Aronson, 2001].

A fórmula do *score* no processamento normal :

$$1000 * (CE + VA + 2 * CO + 2 * CH)/6$$

A fórmula *score* quando a ordem das palavras de entrada não importam no processamento:

$$1000 * (CE + VA + 4 * IV)/6$$

Por fim ocorre a construção do mapeamento que consiste em escolher a melhor combinação de candidatos que formam o mapeamento mais completo possível [Aronson,

1996]. O resultado final por padrão mostra apenas o mapeamento com maior *score*, caso exista mais de um mapeamento com esse valor então aparece ambas as combinações [Aronson, 1996].

O resultado final do processamento do Metamap informa, para cada candidato do mapeamento, suas informações principais. Por exemplo na figura 3 o texto de entrada no retângulo laranja teve o mapeamento dos termos *"forget"*, *"to"*, *"a diabetes screening"*. O *"forget"* e *"a diabetes screening"* tem 1 candidato encontrado enquanto o *"to"* tem 3. Para cada candidato tem em vermelho o nome do seu conceito (preferido), em azul o nome da *string Metathesaurus* e em verde o grupo semântico que pertence.

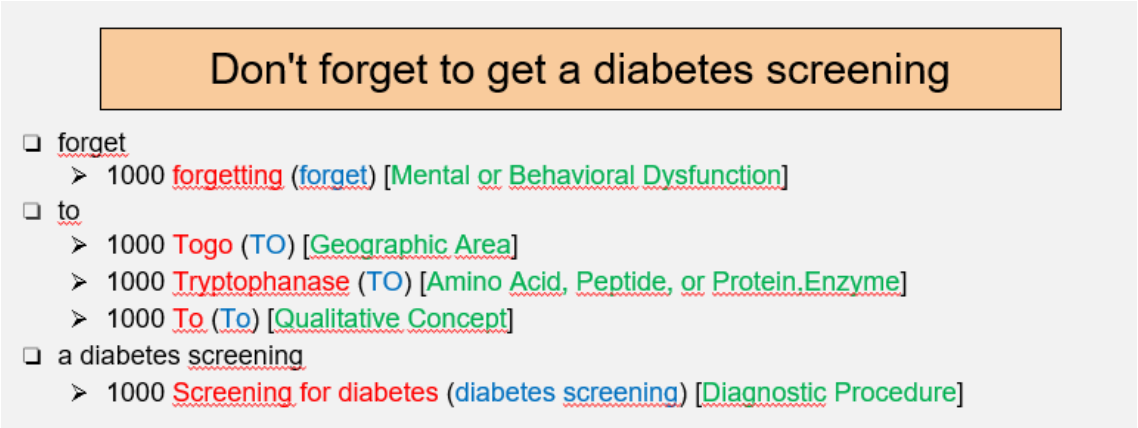

Figura 3. Exemplo do resultado do Metamap

Abaixo estão as estruturas de dados resultantes das etapas do processamento do texto de entrada "Anti-gastroesophageal reflux implantation": [Aronson, 1996].

1. "[implantation,noun]" [Aronson, 1996].
2. "head([lexmatch([implantation]),inputmatch([implantation]),tokens([implantation])])" [Aronson, 1996].
3. "v(implantation,[noun],0,[],implantation,1)" [Aronson, 1996].
4. "v(implant,[verb],3,"d",implant,1)" [Aronson, 1996].
5. "csc([implantation],Implantation,Ovum Implantation)" [Aronson, 1996].
6. "ev(-812,Implantation,Ovum Implantation,[implantation],[Organism Function], [[[4,4],[1,1],0]],yes,no)" [Aronson, 1996].

Os itens 1 e 2 pertencem ao pré-processamento e mostram uma tag encontrada no texto e depois o resultado da análise sintática expandida com token. O itens 3 é o gerador de variante(que também conta como variante) e 4 é uma variante gerada por ele. O item 5 mostra um candidato encontrado do UMLS que é ligado a variante 3. E o item 6 é a pós avaliação do candidato que, nesse exemplo, é escolhido na construção do mapeamento.

2.3. Modelo de dados em português do Metamap

O modelo de dados do Metamap é uma estrutura de dados que permite ao servidor acessar os conceitos do UMLS de vocabulários específicos durante o processo de mapeamento [Metamap_FAQ]. Por padrão o Metamap possui três modelos de dados (*USABase, NLM, Base*) que contém em quantidades diferentes apenas vocabulários em inglês [Metamap_FAQ]. Devido a isso, o mapeamento do Metamap sempre é para conceitos na língua inglesa.

A fim de permitir apenas o mapeamento para conceitos na língua portuguesa criou-se um novo modelo de dados para ser acessado pelo servidor. Ele é feito ao utilizar uma ferramenta adicional do Metamap chamada de *Data File Builder* [Willie Rogers, 2015]. Essa ferramenta contêm um conjunto de *scripts* que permite criar um novo modelo do Metamap, desde que haja a escolha quais vocabulários UMLS deseja utilizar. Nesse caso, foram escolhidos os vocabulários em português do UMLS a seguir: ICPCPOR, LNC-PT-BR, MDRPOR, MSHPOR, WHOPOR [Site_principal_UMLS].

2.4. Modelo de dados na aplicação em Java

Para criar uma aplicação em Java que utiliza o Metamap é necessário instalar a ferramenta adicional Metamap Java API [Metamap_Java_API]. Quando instalada, só precisa importar as suas bibliotecas em Java na aplicação. A lista abaixo mostra os nomes das principais classes disponibilizadas pela API e suas características:

- MetaMapApi: Interface que acessa o servidor do Metamap [Metamap_API_Javadoc]. Ela possui métodos que fazem o processamento do Metamap com diferentes entradas de textos(*String*, *File*) e que permitem adicionar as opções extras do Metamap como escolher o modelo de dados,restringir o grupo semântico entre outros [Metamap_API_Javadoc].
- Result: Um contêiner para a representação de saída da máquina do resultado do MetaMap[Metamap_API_Javadoc].
- Utterance: Representação de uma estrutura de dados que contém o texto de entrada, id próprio, tamanho total do texto e outras informações [Metamap_API_Javadoc].
- PCM: Interface que contém os conjuntos das classes *Phrase, Candidates* e *Mapping* [Metamap_API_Javadoc].
- Candidates: Estrutura de dados que representa um possível candidato do *UMLS* para o mapeamento [Metamap_API_Javadoc].
- Phrase: Estrutura de dados que representa uma parte do texto de entrada geralmente separadas por pontuações [Metamap_API_Javadoc].
- Mapping: Estrutura de dados que representa o resultado da construção do mapeamento [Metamap_API_Javadoc].
- Ev: Instância de avaliação do candidato que contém as informações de nome de conceito,a *string Metathesaurus* encontrada, valor do *Score*, tipo semântico, fonte de vocabulários entre outras [Metamap_API_Javadoc].

Um algoritmo em Java utilizado nos testes está abaixo:

```
MetaMapApi api = new MetaMapApiImpl();
api.setOptions("-V UMLS_PORT");
String entrada="Estou com sinusite e muitas dores.";
List<Result> resultList = api.processCitationsFromString(entrada);

for(Result r : resultList){
    for(Utterance ut : r.getUtteranceList()){
        for(PCM pcm: ut.getPCMList()){
        int cont=1;
        System.out.println("Phrase:" + pcm.getPhrase().getPhraseText())
```

```
            if(pcm.getMappingList().size() != 0){
                for(Mapping m :pcm.getMappingList()){
                    System.out.println("MAPEAMENTO " + cont);
                    for(Ev v : m.getEvList()){
                    System.out.println("Score:" + v.getScore());
                    System.out.println("Conceito:"+ v.getPreferredName());
                    System.out.println("G.Semântico:"+v.getSemanticTypes())
                    System.out.println("Fontes" + v.getSources());
                    System.out.println();
                    }
                    cont++;
                    }
            }else{
                    System.out.println("Não foi mapeado");
            }
            System.out.println("-------------------------------------");
            }
        }
}
```

Nesse algoritmo tem como exemplo o mapeamento da frase de entrada “Estou com sinusite e muitas dores” que utiliza o modelo de dados em português chamado “UMLS_PORT”. E as seguintes informações de saída para cada conceito médico encontrado são: o valor *Score* só que negativo, nome do seu conceito, grupo semântico que pertence e as fontes de vocabulário em que está presente o conceito. Nesse exemplo foi identificado o conceito “Sinusite” com valor -632 do grupo semântico “Disease or Syndrome”(dsyn) presente nos vocabulários “MDRPOR, MSHPOR, WHOPOR”.

3. Resultados

Todas as informações técnicas(versões dos programas, sites de *download*, sistema operacional utilizado) sobre a instalação do Metamap padrão, a criação do modelo de dados em português e seu uso na aplicação em Java foram escritas em outro documento. Nele está incluso detalhes do uso das ferramentas Metamap Data File Builder e Metamap Java API. Assim como os principais comandos no terminal e a explicação detalhada do algoritmo em Java.

Através dos testes do Metamap com o modelo de dados em português proposto descobriu-se algumas limitações no seu uso. Uma delas é o fato do servidor do Metamap não aceitar como entrada palavras com acentuações ou caracteres especiais não presentes na tabela ASCII [Metamap_FAQ]. Mesmo que seja utilizado um modelo de dados que contenha conceitos com acentos nos seus vocabulários. Devido a isso o servidor aborta o processo quando encontra qualquer tentativa de mapeamento com esses caracteres.

Além disso quaisquer variações gramaticais em substantivos e verbos no texto de entrada tem altas chances de não serem mapeadas corretamente na saída. São apenas mapeadas aquelas que são totalmente iguais em escrita aos conceitos que estão nos vocabulários. Por exemplo, a entrada “dor” é mapeada corretamente ao conceito biomédico

"DOR", entretanto não são mapeadas suas variações como "dolorido" ou "dores", pois não existem os termos biomédicos "dolorido" ou "dores" nos vocabulários em português. O correto seria que essas entradas fossem mapeadas para "DOR" por exemplo.

4. Discussão

A limitação acerca dos caracteres especiais fora da tabela ASCII é uma regra do servidor do Metamap que independe do tipo de modelo de dados utilizado [Metamap_FAQ]. Isso afeta negativamente a análise do modelo proposto já que a língua portuguesa é composta, em grande parte, por acentuações.

Outra limitação sobre as variações gramaticais ocorre por causa do léxico inglês *SPECIALIST* nas etapas de geração de variantes e análise sintática [Chiaramello et al., 2016]. Essas etapas utilizam regras da língua inglesa em cima do texto de entrada em português [Chiaramello et al., 2016]. Isso resulta em palavras que são misturas de português e inglês que não podem ser mapeadas. Entretanto a variação gramatical da frase original em português também é utilizada nas etapas posteriores. Pois ela conta como uma variante inalterada e a partir disso pode encontrar um candidato em português para ela desde que o nome do seu conceito seja totalmente igual a essa variação.

A motivação inicial era de utilizar o Metamap em português para auxiliar a farmacovigilância na busca de eventos adversos relacionados a medicamentos em *posts* de mídias sociais. Isso pode ser feito em uma aplicação em Java que utiliza as frases desses *posts* com o modelo de dados proposto. Para cumprir isso, seria necessário especificar os grupos semânticos a que esses eventos pertencem, como por exemplo o grupo *Disorders* [Liu and Chen, 2015]. Além disso a aplicação precisaria mapear eventos que podem ser escritos como gírias [Liu and Chen, 2015]. Isso é difícil para o Metamap devido a sua limitação sobre acentuações e falta de vocabulários médicos que relacionam gírias a conceitos médicos [Liu and Chen, 2015].

5. Conclusão

Com o objetivo de apoiar a farmacovigilância analisou-se as principais características e a estrutura do Metamap. Esse trabalho apresentou as suas etapas do processo de mapeamento da figura 2, os modelos de dados originais dele e sua relação com as fontes de conhecimento do UMLS da figura 1. E criou-se um modelo de dados que permite o mapeamento no idioma português assim como uma aplicação em Java que utiliza esse modelo num processo igual a do Metamap.

Criou-se outro documento que engloba os aspectos técnicos da criação do modelo em português, comandos principais e explicação detalhada do algoritmos em Java. Os testes mostram um desempenho razoável do uso do modelo mas que também existem certas limitações. Entre elas o Metamap não permite textos que contenham acentos e caracteres fora da tabela ASCII e as variações gramaticas tem baixa chance de serem mapeadas.

Baseado no que foi discutido, uma forma de resolver as limitações seria analisar a ferramenta do Metamap chamada Replace UTF8 que converte caracteres não ASCII para ASCII [Site_Principal_Metamap]. E poderia utilizar um novo léxico em português para acabar com a mistura de inglês e português resultantes das etapas de geração de variantes

e análise sintática que diminui as chances de mapeamento correto. E para auxiliar a farmacovigilância deve-se analisar e testar melhor o Metamap junto do modelo de dados e verificar se deve utiliza-los junto de outros softwares.

Referências

Aronson, A. R. (1996). Metamap Technical Notes. *https://ii.nlm.nih.gov/Publications/Papers/metamap.tech.pdf*.

Aronson, A. R. (2001). Metamap evaluation. *https://ii.nlm.nih.gov/Publications/Papers/mm.evaluation.pdf*.

Chiaramello, E., Pinciroli, F., Bonalumi, A., Caroli, A., and Tognola, G. (2016). Use of "off-the-shelf" information extraction algorithms in clinical informatics: A feasibility study of MetaMap annotation of Italian medical notes. *Journal of Biomedical Informatics*, 63:22–32.

Liu, X. and Chen, H. (2015). A research framework for pharmacovigilance in health social media: Identification and evaluation of patient adverse drug event reports. *Journal of Biomedical Informatics*, 58:268–279.

Luo, Y., Thompson, W., Herr, T., Zeng, Z., Berendsen, M., Jonnalagadda, S., Carson, M., and Starren, J. (2017). Natural Language Processing for EHR-Based Pharmacovigilance: A Structured Review. *Drug Safety*, 40(11):1075–1089.

Metamap_API_Javadoc. Javadoc da java api do metamap. *https://metamap.nlm.nih.gov/javaapi/javadoc/index.html*.

Metamap_FAQ. Metamap usage Frequently Asked Questions. *https://metamap.nlm.nih.gov/Docs/FAQ/MM_FAQ.pdf*.

Metamap_Java_API. Metamap java api. *https://metamap.nlm.nih.gov/JavaApi.shtml*.

Site_Principal_Metamap. Site principal do Metamap. *https://metamap.nlm.nih.gov/*.

Site_principal_UMLS. UMLS. *https://www.nlm.nih.gov/research/umls/*.

UML_tutorial. The Unified Medical Language System (UMLS). *https://www.nlm.nih.gov/research/umls/new_users/online_learning/OVR_001.html*.

Willie Rogers, Fran¸cois-Michel Lang, C. G. (2015). Metamap data file builder. *https://metamap.nlm.nih.gov/Docs/datafilebuilder.pdf*.

Wong, A., Plasek, J., Montecalvo, S., and Zhou, L. (2018). Natural Language Processing and Its Implications for the Future of Medication Safety: A Narrative Review of Recent Advances and Challenges. *Pharmacotherapy*, 38(8):822–841.

6. Agradecimentos

O autor agradece ao CEFET-RJ pelo apoio no desenvolvimento desta pesquisa. E também agradece aos orientadores Kele Belloze e Gustavo Paiva Guedes pelo suporte, direcionamento e inspiração. E a minha família que deu forças e auxílio.