

Diorge Brognara S.

**Extração automática de relações semânticas a
partir de dados ruidosos**

São Carlos

2020

Diorge Brognara S.

**Extração automática de relações semânticas a partir de
dados ruidosos**

Dissertação de mestrado
Orientador: Prof. Dr. Ricardo Cerri

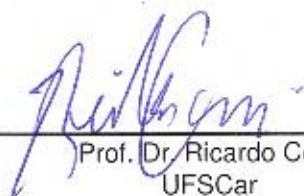
Universidade Federal de São Carlos — UFSCar
Departamento de Computação
Programa de Pós-Graduação

São Carlos
2020



Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Diorge Brognara Sardinha, realizada em 09/03/2020:



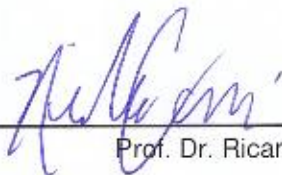
Prof. Dr. Ricardo Cerri
UFSCar



Profa. Dra. Helena de Medeiros Caseli
UFSCar

Profa. Dra. Veronica Oliveira de Carvalho
UNESP

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Veronica Oliveira de Carvalho e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.



Prof. Dr. Ricardo Cerri

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Extração de relações é uma tarefa realizada em sistemas de aprendizado contínuo a partir de fontes textuais, com o objetivo de encontrar relações semânticas entre categorias ou entidades. A NELL é um sistema desse tipo, que encontra problemas na extração de relações devido a suas propriedades de supervisionamento e rotulação. Um dos algoritmos para essa tarefa desenvolvidos para a NELL é o OntExt, que apresenta dificuldades de execução devido a entradas ruidosas e ao seu custo computacional. No entanto, o algoritmo possui propriedades interessantes para o contexto da aplicação, que não estão presentes em outros métodos. Neste trabalho, é proposta uma variante do algoritmo a fim de aliviar as principais deficiências identificadas, que utiliza uma estrutura de grafo, e é flexível para tratamento de *outliers*. O novo método proposto possui precisão comparável ao existente, e uma revocação maior. Também é apresentada uma forma eficiente de representar o problema através de uma estrutura esparsa, reduzindo o custo computacional da ordem de minutos para segundos.

Palavras-chave: extração da informação, extração de relações, aprendizado contínuo.

Abstract

Relationship extraction is a task performed in text-based continuous learning systems, aiming to find semantic relationships between categories or entities. NELL is such a system, which suffers from supervised labeling in its relationship extraction. One of the algorithms attempting to solve this task for NELL is OntExt, but it does not handle noisy input very well, and is computationally expensive. However this algorithm has interesting properties in the context of NELL's application, not available in other methods. In this work, it is proposed a variant of the algorithm to reduce the impact of its flaws, using a graph-based representation, which is flexible in the handling of outliers. This new method has a comparable precision and higher recall, compared to the existing method. It is also shown an efficient way to represent the problem using sparse structures, reducing the computational cost from minutes to seconds.

Keywords: information extraction, relationship extraction, continuous learning.

Lista de ilustrações

Figura 1 – Exemplo de aplicação do VerbKB, extraído de (WIJAYA, 2016)	18
Figura 2 – Consulta SQL gerando os valores das coordenadas da matriz de co-ocorrência	34

Lista de tabelas

Tabela 1 – Estrutura da entrada recebida pelo algoritmo Snowball	17
Tabela 2 – Performance dos métodos OntExt e HCSw, para os mesmos 100 pares de categoria, de acordo com dois humanos juízes (E1 e E2)	30
Tabela 3 – Variáveis relacionadas ao custo computacional	35

Lista de abreviaturas e siglas

IE	Extração de Informação (<i>Information Extraction</i>), subárea de aprendizado de máquina
ER	Extração de Relações, uma tarefa de IE
KB	Base de conhecimento (<i>Knowledge Base</i>)
SVO	Arquivo contendo uma coleção de sentenças no formato de triplas (<i>sujeito, sintagma verbal, objeto</i>)

Sumário

1	INTRODUÇÃO	12
2	DOMÍNIO DE APLICAÇÃO	14
3	EXTRAÇÃO DE RELAÇÕES	16
3.1	Extração supervisionada	16
3.2	Extração tradicional	17
3.3	Extração aberta	18
3.4	Extração não-supervisionada	19
3.5	OntExt	20
3.6	Avaliação de métodos de extração	24
4	GRAFO DE CO-OCORRÊNCIA	26
4.1	Algoritmo HCS	26
4.2	Algoritmo HCSw	27
4.3	Experimento	29
5	GERAÇÃO DA ESTRUTURA DE CO-OCORRÊNCIA	32
6	CONSIDERAÇÕES FINAIS	36
	REFERÊNCIAS	38

1 Introdução

Extração da Informação (*Information Extraction*, IE) é a área que estuda a aquisição automática de conhecimento a partir de fontes não-estruturadas de dados, como textos livres presentes na *Web* (será utilizado o termo *corpus* para designar fontes textuais de dados), e estruturar esse conhecimento em uma representação mais acessível (WILKS, 1997). Uma estrutura comumente utilizada para esse propósito é a ontologia, expressada através do uso de predicados, e frequentemente representada através de um banco de dados relacional ou de um grafo. Exemplos de tarefas de IE incluem a extração de entidades nomeadas e a extração de relações (ER).

A tarefa de ER consiste em encontrar relações semânticas entre diferentes categorias de entidades. Por exemplo, o padrão textual “<Instituição>, localizada em <Local>” caracteriza a relação entre o par de categorias “instituição” e “local”, que pode ser descrita pelo predicado binário *localizado_em(Instituicao, Local)*, geralmente utilizando iniciais maiúsculas para identificar variáveis. Quando as variáveis são substituídas por entidades, dizemos que a relação foi instanciada, como no predicado *localizado_em(ufscar, sao_carlos)*. Para identificar que uma entidade pertence a uma categoria, utilizam-se predicados unários, como *instituicao(ufscar)*, sendo assim uma instância de categoria. Não serão tratados predicados com outras aridades.

A literatura apresenta alguns métodos de ER, com diferentes características, particularmente em relação aos dados de entrada e saídas produzidas. Em geral, os métodos podem produzir um ou mais dos seguinte elementos: um conjunto de instâncias de relações, novos padrões textuais para uma relação (sinônimos), e/ou novas relações. No entanto, a principal diferença entre os algoritmos de ER refere-se à abordagem utilizada nos dados de entrada, divididos entre extração tradicional, extração aberta, e sistemas híbridos (BANKO; ETZIONI, 2008; MOHAMED; Hruschka Jr; MITCHELL, 2011). Métodos de extração tradicional recebem dados manualmente rotulados de uma relação específica, na forma de padrões textuais ou sentenças anotadas, enquanto na extração aberta é utilizado somente o *corpus*; os algoritmos híbridos trabalham com menos dados que os tradicionais, mas possuem algum tipo de informação rotulada (por exemplo, quais entidades pertencem a uma categoria).

Extração de relações é importante para aprendizes contínuos como a NELL¹, um sistema de descoberta do conhecimento autônomo, semissupervisionado, cujo objetivo é criar uma base de conhecimento (*knowledge base*, KB) sobre diferentes domínios, usando o *framework* de aprendizado sem fim, e (idealmente) somente dados provenientes da *Web* (CARLSON et al., 2010). Nesse contexto, um algoritmo de ER não é responsável somente pelo aumento da KB (através de novas instanciações), mas também por melhorar a performance de

¹ rtw.ml.cmu.edu/rtw/

outros componentes, pois há a vantagem do co-acoplamento entre as diferentes categorias e relações (CARLSON et al., 2009). Dado que existe uma KB disponível como entrada, intuitivamente é vantajoso utilizar métodos híbridos de ER que aproveitem essa informação adicional.

Foram desenvolvidos os algoritmos OntExt (MOHAMED; Hruschka Jr; MITCHELL, 2011) e VerbKB (WIJAYA, 2016) para serem aplicados na NELL. Ambos utilizam uma base de dados no formato SVO (*Subject-Verbal phrase-Object*), onde cada instância é uma sentença no padrão “Sujeito — Sintagma verbal — Objeto”, e integram o conhecimento já existente na KB para realizar um agrupamento semissupervisionado nestes dados. No caso do VerbKB, é utilizado um algoritmo de particionamento *Expectation-Maximization* (“hard EM”) para encontrar novos padrões textuais para relações existentes, enquanto o OntExt utiliza o algoritmo K-Means, também de particionamento, mas com o intuito de encontrar novas relações.

Atualmente, o OntExt não é utilizado dentro da NELL devido a problemas relacionados aos dados ruidosos e inconsistentes provenientes da *Web*, a escalabilidade do algoritmo em relação ao número de sentenças no *corpus*, entre outros problemas. Este componente, no entanto, tem certas propriedades interessantes para a NELL, que serão discutidas nos Capítulos 2 e 3, fazendo com que diversas melhorias fossem propostas (NAVARRO et al., 2016; BARCHI et al., 2014; BROGNARA; Hruschka Jr, 2016), a fim de viabilizar a execução do método.

Com o objetivo de melhorar o processo de extração de relações da NELL, foram analisados diversos métodos de ER, avaliando suas características, vantagens e desvantagens quando aplicados dentro da NELL. Dado que o OntExt mostrou ter propriedades interessantes para o sistema, este trabalho apresenta modificações do algoritmo, a fim de diminuir o esforço computacional do método e aumentar a qualidade das relações extraídas.

Assim, este trabalho é estruturado da seguinte maneira: no Capítulo 2, é apresentada a arquitetura do sistema NELL, e quais dados estão disponíveis para possíveis métodos de ER aplicados ao sistema, e quais são as saídas desejadas. No Capítulo 3, são apresentados alguns diferentes algoritmos de ER, discutindo as abordagens para solução do problema de extração de relações na literatura, e detalhando o funcionamento e deficiências do OntExt. Duas possíveis soluções foram estudadas para os problemas identificados no OntExt, uma abordagem baseada em grafos é apresentada no Capítulo 4, enquanto uma reestruturação da matriz de co-ocorrência é apresentada no Capítulo 5. Por fim, o Capítulo 6 apresenta as conclusões e próximos passos.

2 Domínio de aplicação

O objetivo desse trabalho é propor uma nova solução de ER a ser aplicada no sistema NELL, *Never-Ending Language Learner*, um sistema desenhado para ser executado continuamente, todos os dias e para sempre, executando duas tarefas: ler e extrair informações da *web*, construindo uma base de conhecimento; e aprender a ler melhor com o tempo, de forma que revisitar textos já lidos extraia informações novas e mais acuradas (CARLSON et al., 2010).

A arquitetura da NELL funciona através do compartilhamento de uma KB unificada, utilizada por diversos componentes independentes. A KB é dividida em duas partes, as *crenças* e os *fatos candidatos*. A cada iteração, todos os componentes acessam as crenças atuais, produzindo novos fatos candidatos. Um subsistema separado, o *Knowledge Integrator*, é responsável por analisar os fatos candidatos e promover aqueles com maior confiança para serem crenças.

Cada componente tem acesso a recursos externos, além da base de crenças. Diferentes componentes podem produzir o mesmo tipo de fato candidato, por exemplo instâncias de categorias (predicados unários instanciados) para as categorias já existentes na KB. Assim, diferentes módulos com erros não-correlacionados criando os mesmos fatos candidatos favorecem a chance desses serem promovidos a crenças.

A NELL utiliza um método de *bootstrapping*, isto é, é um sistema iterativo que se retroalimenta. A primeira iteração da NELL recebe uma ontologia construída manualmente, e como o resultado da iteração é um aumento dessa ontologia, as próximas iterações utilizam o conhecimento gerado pelas iterações anteriores. Isso permite que o sistema aprenda continuamente, e também causa que erros em uma iteração se propaguem para as iterações seguintes, aumentando a taxa de erros conforme o tempo, em um processo chamado desvio semântico. Um exemplo de desvio semântico na NELL é a categoria *sport*, onde por um erro em alguma iteração passou a considerar times esportivos como esportes, e hoje contém várias universidades e escolas, devido a times esportivos universitários, que tem o mesmo nome que suas instituições.

A principal ferramenta que a NELL utiliza para evitar desvios semânticos é o *Knowledge Integrator*. Como o *Knowledge Integrator* pode esperar “evidências suficientes” para que um fato candidato seja promovido a crença, desde que os componentes cometam erros não-correlacionados, a maior parte dos erros não seriam promovidos a crença (CARLSON et al., 2010). A NELL ainda permite uma iteração limitada com humanos, a fim de corrigir possíveis erros, idealmente antes destes erros causarem um desvio semântico. No entanto, os componentes cometem erros correlacionados, por exemplo os componentes CMC e CPL ambos promovendo *persistent cookies* para a categoria *bakedGoods* (CARLSON et al., 2010).

Em (CARLSON et al., 2009), são discutidas as vantagens do acoplamentos de categorias e relações. A ideal principal desse acoplamento é que aprendizes que tentam aprender separadamente cada categoria ou relação possuem performance inferior a aprendizes que tentam aprender um conjunto de categorias e relações simultaneamente. A justificativa é que existe conhecimento intrínseco do uso de múltiplos domínios, com o exemplo “*tecnico(X)* implica em *pessoa(X)* e \neg *esporte(X)*” (CARLSON et al., 2009). O mesmo se aplica a relações, por exemplo, novas relações podem ser geradas a partir de transitividade, como o conhecimento de que *atleta_joga_em_time(Atleta, Time)* e *time_de_esporte(Time, Esporte)* implica em *atleta_joga_esporte(Atleta, Esporte)*.

Sendo assim, é interessante para um sistema como a NELL que haja acoplamento dos subsistemas aprendizes. Naturalmente, para que haja o maior acoplamento possível, é necessário que sejam identificadas novas categorias e relações, sobre as quais novas regras de acoplamento possam ser criadas. Ou seja, a identificação de novas relações não somente é interessante como um fim, mas também é um meio para uma maior precisão na identificação de instâncias para as relações já existentes.

3 Extração de relações

O processo de extração de relações visa aprender novas instâncias, novas relações ou padrões textuais para relações existentes. Uma relação pode ser escrita utilizando mais de um padrão textual, e um padrão textual pode representar mais de uma relação. Por exemplo, a relação *localizado_em*(*Instituicao*, *Local*) pode ter os padrões “<Instituição>, localizada em <Local>” e “<Instituição> (<Local>)”, e a relação *ocorreu_em*(*Evento*, *Ano*) também pode ter o padrão “<Evento> (<Ano>)” (a troca do nome das variáveis não altera o padrão textual).

A vantagem de se obter padrões textuais é poder fazer buscas simples no *corpus* para encontrar novas instâncias da relação. Uma quantidade maior de padrões textuais para uma mesma relação acarreta assim um aumento na cobertura das instâncias, ou seja, são encontradas mais instâncias entre o total possível de instâncias válidas. Diferentes padrões textuais para a mesma relação também permitem utilizar a redundância a fim de aumentar a precisão e confiança do algoritmo, não promovendo todas as sentenças que combinam com o padrão (*match*¹) mas requerendo um certo limiar de *matches*.

Neste capítulo, serão discutidos os principais métodos de extração de relação presentes na literatura. No entanto, o levantamento desses trabalhos relacionados tem um foco não nos resultados obtidos pelos métodos em suas respectivas aplicações, mas na afinidade que a abordagem do método possui com a NELL, considerando os tipos de dados e os objetivos do projeto.

3.1 Extração supervisionada

Uma forma de tratar a extração de relações é utilizando um modelo estritamente supervisionado de aprendizagem. Nesse caso, as possíveis relações são fixas e pré-definidas. Ou seja, define-se um conjunto de relações de interesse R , e é dado como entrada um conjunto de sentenças, onde cada sentença é rotulada com uma das relações definidas. O objetivo da extração supervisionada é construir um classificador capaz de prever o rótulo de uma sentença, com imagem no conjunto $R \cup n$, onde n representa a classe “não há relação”.

Para construir tal classificador podem ser utilizadas abordagens de extração de atributos, onde elementos de nível léxico e sintático são utilizados para extrair atributos relevantes e treinar um classificador tradicional, ou através da construção de funções de *kernel* específicas para o domínio de aplicação. Existe ainda uma variação nesses métodos considerando se as entidades nomeadas já são rotuladas ou não. Alguns algoritmos

¹ Será utilizado o termo técnico *match* em inglês para identificar que uma sentença combina com um determinado padrão textual.

fazem ambos os processos simultaneamente e tendem a ter melhores resultados que uma aplicação sequencial (PAWAR; PALSHIKAR; BHATTACHARYYA, 2017). No entanto, o modelo de extração supervisionada não é capaz de atender aos requisitos da NELL, principalmente por não encontrar novas relações, mas somente instâncias de relações.

3.2 Extração tradicional

Em (BANKO; ETZIONI, 2008), são discutidas as diferenças entre as abordagens chamadas de extração de relações tradicional e extração de relações aberta. Define-se como extração tradicional o processo onde o sistema recebe como entrada, além do *corpus*, exemplos rotulados de uma relação-alvo específica, na forma de sentenças, predicados ou padrões textuais, e então são encontrados novos padrões textuais e instâncias dessa relação no *corpus*. Logo, para que o sistema considere outra relação é necessário um processo manual para rotular os exemplos específicos àquela relação. Esse tipo de algoritmo pode ser considerado como um aprendiz semissupervisionado.

Um sistema que utiliza a abordagem tradicional é o *Snowball* (AGICHTEIN; GRAVANO, 2000). Os exemplos utilizados para treinar esse sistema são instâncias de uma relação. A Tabela 1 mostra uma possível entrada para o sistema, manualmente escolhida para encontrar padrões textuais e instâncias da relação *localizada_em(Cidade, Pais)* (embora o sistema desconheça a relação). Utilizando esses dados, o sistema é responsável por buscar no *corpus* padrões textuais onde um dos pares dos exemplos ocorre. Os padrões encontrados são utilizados para realizar o *match* no restante do *corpus*, encontrando novos pares que são agregados à tabela de exemplos, e recomeçando o processo iterativamente. O sistema ainda trata de graus de confiança, tanto em relação à confiança de um padrão textual, como em relação à uma instância da relação, baseando-se em medidas de similaridade no cálculo do *match*.

São Paulo	Brasil
New York	Estados Unidos
Paris	França

Tabela 1 – Estrutura da entrada recebida pelo algoritmo Snowball

Outro método que pode ser considerado como extração tradicional é o VerbKB (WIJAYA, 2016). O VerbKB é capaz de utilizar a relação de co-ocorrência entre sintagmas verbais para alinhar duas representações de conhecimento. O método é aplicado a duas bases de conhecimento, utilizando um *corpus* como intermediário para encontrar relações sinônimas e permitir a troca de instâncias. Também é possível substituir uma KB por um pequeno conjunto de dados rotulados. A Figura 1 mostra uma aplicação dessa estratégia, onde o termo *web* representa o *corpus*, e *Verb Resource* representa a informação gerada pelo VerbKB utilizando os dados supervisionados disponíveis. Os verbos conhecidos são

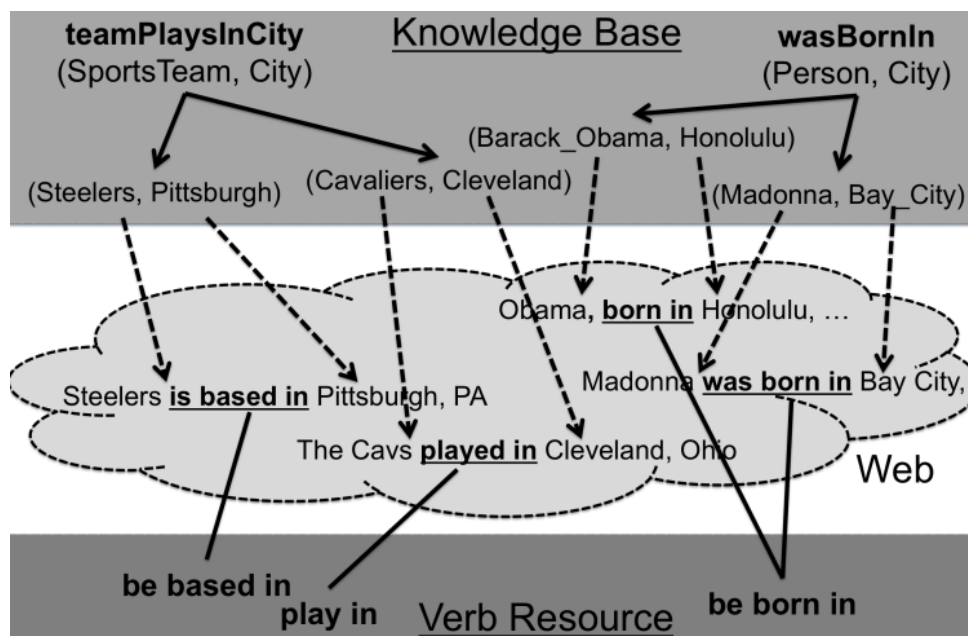


Figura 1 – Exemplo de aplicação do VerbKB, extraído de (WIJAYA, 2016)

associados às instâncias de relações já existentes na KB, mapeando padrões textuais com relações, e então utilizando tais padrões para encontrar novas instâncias no texto.

A maior desvantagem dos métodos de extração tradicional para a NELL é a necessidade de se conhecer a relação antes da execução. Tais sistemas produzem instâncias de relações, mas são incapazes de produzir novas relações. Como discutido no Capítulo 2, embora essa tarefa também seja importante, o escopo deste trabalho inclui promover novas relações.

3.3 Extração aberta

Em contraste com o modelo de extração tradicional, um sistema de extração aberto recebe como entrada somente o *corpus*. As relações a serem procuradas não são definidas previamente, e o sistema é responsável tanto por achar os padrões textuais da relação assim como instâncias das relações encontradas. É natural esperar que algoritmos que utilizem essa abordagem tenham uma performance menor que os de extração tradicional, mas como em quase qualquer tarefa de ER é possível ajustar o balanço entre precisão e revocação do sistema.

O primeiro método na literatura que se define como extração aberta de relações é proposto em (BANKO et al., 2007), chamado de *TextRunner*. O algoritmo possui três módulos. Primeiro é treinado um classificador Naive-Bayes para identificar extrações como confiáveis ou não, utilizando como dados de treinamento o resultado da aplicação de certas heurísticas na saída de um analisador morfossintático (*parser*). O segundo módulo é um extrator que utiliza as classes sintáticas de cada sentença para remover certos elementos como advérbios e construções preposicionais que, heurísticamente, não alteram a semântica

da sentença. Todas as frases extraídas pelo extrator que passam por um certo limiar de confiança são então classificadas pelo primeiro módulo, mantendo somente aquelas classificadas como confiáveis. Por último, um módulo de redundância utiliza medidas de similaridade para encontrar padrões textuais que representam a mesma relação.

Também é possível montar sistemas híbridos, que unem o modelo tradicional e o aberto, por exemplo utilizando *ensemble stacking*, uma técnica de coleção de classificadores (BANKO; ETZIONI, 2008). Neste modelo, são utilizados como classificadores básicos do *ensemble* algoritmos tanto da abordagem tradicional quanto da abordagem aberta, e treinados em conjunto. Dessa forma, é possível treinar modelos tradicionais com menos dados rotulados, utilizando a saída do modelo aberto, mas perde-se a flexibilidade de obter resultados para mais de um par de categorias, uma vez que os algoritmos tradicionais necessitam de exemplos rotulados para relações específicas.

Uma crítica a sistemas abertos de extração é de que “extração aberta é puramente textual e incapaz de relacionar as formas superficiais a uma ontologia” (SODERLAND et al., 2010)². É necessário o treinamento de aprendizes externos, com o conhecimento de domínio desejado, para que o resultado de uma extração aberta seja convertido para uma forma estruturada, como uma ontologia. Um processo capaz de fazer tal conversão é apresentado por (SODERLAND et al., 2010), mas o requerimento de conhecimento de domínio supervisionado remove uma das principais vantagens da extração aberta. Considerando tal desvantagem, esses métodos perdem sua relevância para um sistema aprendiz autossupervisionado como a NELL, que utiliza o nome da relação como informação, ou seja, é importante para a NELL que a relação seja nomeada *trabalha_em(pessoa, companhia)* e não algo sem significado como *REL005(pessoa, companhia)*.

3.4 Extração não-supervisionada

Existe uma diferenciação na literatura sobre métodos de extração de relações não-supervisionada e o paradigma de extração aberta de relações, embora tal diferença não seja clara. Os métodos denominados não-supervisionados, assim como os de extração aberta, não recebem instâncias de relações manualmente rotulados como entrada. A principal diferença é que os métodos considerados como não-supervisionados recebem como entrada as possíveis categorias que as entidades pertencem, ou seja, instâncias de categorias, ou automaticamente geram tal informação utilizando um classificador de entidades nomeadas previamente treinado.

Um dos primeiros exemplos de um sistema não-supervisionado é apresentado em (HASEGAWA; SEKINE; GRISHMAN, 2004). O método consiste em encontrar sentenças onde há pares de entidades nomeadas lexicamente próximas, e usar as palavras entre as duas ocorrências para criar um contexto do par. É utilizado um classificador de entidades

² No original: “Open IE is purely textual and is unable to relate the surface forms to an ontology”

nomeadas externo para determinar as sentenças válidas. Esses contextos são vetorizados utilizando a transformação Tf-Idf na representação de contagem (*bag-of-words*). Com essa forma matricial, os pares são agrupados utilizando um algoritmo hierárquico de ligação completa. Os grupos são rotulados de acordo com as palavras mais frequentes que são comuns aos membros do grupo.

O algoritmo OntExt é proposto em (MOHAMED; Hruschka Jr; MITCHELL, 2011) como um sistema de extração aberta, embora seu funcionamento seja mais próximo dos métodos não-supervisionados. De fato, o método é muito similar estruturalmente ao apresentado em (HASEGAWA; SEKINE; GRISHMAN, 2004), alterando a forma de representação da matriz, e o algoritmo de agrupamento utilizado. Neste modelo o algoritmo extrator recebe como entrada o *corpus* e exemplos de instâncias de duas categorias. Não é necessário, no entanto, nenhum exemplo de instâncias de relações, ou qualquer informação sobre as relações, como acontece para os algoritmos de extração tradicional. O resultado são novas relações entre o par de categorias, os padrões textuais encontrados para essas relações, instâncias iniciais para as relações, e um nome relevante para cada relação. Diferente de outros algoritmos não-supervisionados, as categorias das entidades (conjunto de entrada) são provenientes de uma ontologia, e não de um extrator de entidades nomeadas, o que permite maior flexibilidade nas relações que podem ser encontradas. Em (NAKASHOLE; WEIKUM; SUCHANEK, 2012) são levantadas as questões da escalabilidade do método, e sua incapacidade de inferir a hierarquia semântica das relações encontradas. O funcionamento desse método é detalhado na Seção 3.5.

Devido às características apresentadas neste capítulo, os métodos de extração não-supervisionada se mostram mais adequados para a aplicação de encontrar novas relações dentro da NELL. Outros métodos são utilizados em KBs existentes, como em TextRunner (BANKO et al., 2007) e PATTY (NAKASHOLE; WEIKUM; SUCHANEK, 2012). No entanto, essas bases tem objetivos distintos da NELL; especificamente, o TextRunner não dá grande consideração ao valor semântico das relações, enquanto o PATTY se atenta somente às relações e não aos fatos (instâncias de relações).

3.5 OntExt

Nessa seção será descrito o funcionamento do algoritmo OntExt, na sua versão mais recente, onde há integração com outro componente da NELL, o Prophet (NAVARRO et al., 2016). O sistema que integra ambos os algoritmos é chamado de PrOntExt. A vantagem de tal construção é que o Prophet é capaz de gerar uma lista pares de categorias onde há confiança de que haja relações desconhecidas, e essa informação é fornecida ao OntExt, evitando assim a necessidade de entrada humana no processo.

O Prophet é um algoritmo de mineração em grafos, que opera sobre a KB da NELL convertendo as categorias em vértices e as relações em arestas. A etapa relevante para

o OntExt são os triângulos abertos encontrados nesse processo de mineração. De forma geral, o Prophet encontra pares de categorias onde não há relação, mas com certo grau de confiança de que pode existir uma relação entre as categorias. Como subproduto do processo, também são geradas as listas de instâncias de cada uma das categorias consideradas, para que o OntExt não precise acessar a KB diretamente. Esses dados são então utilizados para alimentar o OntExt.

Tendo como entrada o par de categorias, os conjuntos de entidades dessas categorias (de acordo com a KB), e um conjunto de sentenças no formato SVO, o OntExt primeiro cria uma matriz de co-ocorrência entre os padrões textuais, cruzando o sujeito e o objeto da sentença. Essa é uma matriz de tamanho $|V| \times |V|$, onde $|V|$ representa a quantidade de sintagmas verbais diferentes presentes nos dados. Ou seja, são contados quais V nos dados compartilham os exatos mesmos S e O. Por exemplo, se estão presentes no SVO as sentenças “João, conversou com, Maria”, “João, falou com, Maria” e “João, é casado com, Maria”, a matriz seria incrementada nas células onde tanto a linha como a coluna representam um dos três sintagmas verbais.

Após a contagem de todas as co-ocorrências entre as sentenças, ocorre uma etapa de normalização. Primeiro os números são convertidos para o intervalo $[0, 1]$ utilizando a transformação:

$$M'_{ij} = \frac{M_{ij}}{\sum_{j=0}^{|V|} M_{ij}}$$

Esse valor é então alterado novamente, utilizando o valor G_i , a quantidade de células diferentes de zero na linha i :

$$M''_{ij} = M'_{ij} \frac{|V|}{G_i}$$

Por fim, a matriz M'' é utilizada como entrada em um algoritmo K-Means com $K = 5$, resultando em um particionamento dos sintagmas verbais em cinco grupos. O valor cinco foi encontrando empiricamente para o conjunto de dados onde o OntExt foi inicialmente testado. O resultado é interpretado da seguinte maneira: cada grupo representa uma nova relação entre as categorias, o medoide do grupo (elemento mais próximo do centroide) é o nome dessa nova relação, e cada elemento do grupo é um padrão textual para a relação, ou seja, um sinônimo do medoide.

No exemplo mostrado no trabalho original (MOHAMED; Hruschka Jr; MITCHELL, 2011), a base de dados (em inglês) procura por relações entre o par de categorias “remédio” e “doença”, gerando dois grupos, um com os sintagmas verbais “trata”, “é tratamento para” e “medicação”, e outro com “pode causar”, “podendo causar” e “acarreta em”.

O OntExt utiliza ainda uma métrica para ranqueamento das instâncias mais relevantes para cada relação. Cada par $(Sujeito, Objeto)$, s , recebe uma nota que é a soma das ocorrências de s com cada contexto c , dividido pelo desvio padrão do vetor da diferença

entre o contexto c e o centroide de seu grupo³. A Equação 3.1 mostra essa relação como originalmente descrita, onde Occ é a contagem de ocorrências, e sd é o “desvio padrão” mencionado. A justificativa para essa métrica é que a ocorrência com contextos mais próximos do centroide “fortaleçam” o par, embora nesse caso a norma ou outra métrica de distância entre o vetor do contexto e o centroide pareçam mais adequadas que o desvio padrão da diferença dos vetores. Os 50 pares de $(Sujeito, Objeto)$ com a maior nota T de cada relação são promovidos pelo OntExt como instâncias da relação⁴.

$$T(s) = \sum_{c \in cluster} \frac{Occ(c, s)}{1 + sd(c)} \quad (3.1)$$

Dentro do contexto da NELL, esse processo de ER é interessante por alguns motivos. Primeiramente o método utiliza mais dados além do *corpus*, caracterizando a abordagem híbrida de extração de relações, que intuitivamente deveria ter desempenho melhor que as outras duas abordagens, como discutido nas seções anteriores. Ainda assim, todos esses dados estão presentes na KB ou são gerados pelo próprio sistema, sem que haja a necessidade de intervenção humana.

Outro motivo é que o algoritmo gera não somente novos padrões textuais e instâncias de relação, mas também novas relações. A NELL utiliza métodos de co-acoplamento que permitem que um determinado conceito seja aprendido mais facilmente utilizando-se outros conceitos (CARLSON et al., 2009). Ou seja, não somente a NELL passa a aprender sobre conceitos diferentes que ainda não haviam sido considerados, o que por si só já é de interesse no projeto, mas também permite que conceitos já conhecidos tenham um ganho de desempenho.

Como terceiro motivo, o OntExt produz instâncias iniciais da relação (*seeds*). Como diversos componentes da NELL são baseados em aumentar um conjunto de instâncias já existentes, geralmente por processos iterativos incrementais, é importante que haja um conjunto de instâncias iniciais para que esses outros algoritmos passem a considerar a nova relação e buscar novas instâncias e padrões textuais para ela.

Além disso, o OntExt não utiliza informações da língua natural sendo utilizada no *corpus*, embora seja assumido que existe somente uma língua. Isso pode ser considerado uma vantagem no futuro, caso haja a necessidade de aplicação do algoritmo para línguas que não o inglês.

Essas propriedades fazem com que o OntExt seja um dos componentes de interesse para ser executado nas iterações da NELL. No entanto, a implementação atual do algoritmo é inviável dentro do sistema, principalmente em termos de escalabilidade.

Os dados utilizados, apesar de serem filtrados mantendo somente as sentenças no formato SVO, são provenientes da *Web*, uma fonte de dados naturalmente ruidosa. Além

³ O texto original menciona “standard deviation of the context from the centroid of the relation contexts cluster”.

⁴ Não é justificada a escolha do número 50, e nenhum experimento foi realizado sobre a qualidade das instâncias promovidas.

disso, esses dados passam por processos automáticos de analisadores léxico-sintáticos, que embora muito evoluídos em tempos recentes, ainda adicionam uma certa quantidade de erros. Uma forma comumente utilizada quando se trata desse tipo de fonte ruidosa é fazer uso da redundância da informação, requerendo uma certa quantidade de ocorrências ou um grau de confiança alto.

Esse problema não é considerado na implementação atual do OntExt, embora seja descrito no trabalho original (MOHAMED; Hruschka Jr; MITCHELL, 2011), e causa problemas não só no desempenho mas também no custo computacional do método. Em um experimento realizado na base de dados da NELL em inglês, foram encontrados 135.418 contextos (sintagmas verbais) diferentes para o par de categorias “*person*” e “*room*” (BROGNARA; Hruschka Jr, 2016). A matriz de co-ocorrência nesse caso teria $135.418^2 = 18.338.034.724$ células de números reais, o que contabilizaria ao menos 136 GB de memória RAM⁵, se armazenados em uma matriz densa. É notado ainda que grande parte desses dados são nulos, mas a implementação não faz uso de matrizes esparsas. Não só isso mostra que existe uma quantidade grande de dados, o que indica que muitos desses contextos podem ser errôneos, também inviabiliza a execução em máquinas de pequeno porte, além de dificultar uma execução paralelizada em diversos pares de categorias concorrentemente sem que a memória seja um fator limitante.

Depois que a matriz de co-ocorrência é criada e normalizada, é executado o algoritmo de particionamento K-Means, especificamente sua variante “K-Means++”. Esse é um algoritmo de agrupamento “rígido”, que particiona todos os elementos para um dos grupos, e nenhum elemento pertence a mais de um grupo. Outra propriedade é que a quantidade de grupos é um hiper-parâmetro do algoritmo, sendo empiricamente ou heurísticamente escolhido, no caso do OntExt com o valor 5. Embora em alguns casos isso seja desejável, para o OntExt essas características são contestáveis. Como discutido anteriormente, existe uma quantidade grande de contextos a serem analisados, e deixar de agrupar um elemento considerado como *outlier* pode aumentar a performance do algoritmo. O K-Means ainda é conhecido por sofrer com o fenômeno “maldição da dimensionalidade”, por ser um algoritmo baseado em distância, e é comum matrizes de co-ocorrência com dimensões altas.

A escolha do hiper-parâmetro K com o valor fixo 5 foi empiricamente baseada no conjunto de dados originalmente testado. Tais escolhas empíricas podem ser justificadas em muitos casos, especialmente quando há auxílio visual para um humano determinar o valor. Mas neste caso o valor será utilizado em mais de um processo de agrupamento, e sem nenhuma supervisão humana. Para alguns pares de categorias podem ser encontrados somente dez contextos, que representem todos a mesma relação, e em outros mais de 100.000 contextos, que dificilmente formam somente cinco relações. Dada a natureza do

⁵ Assumindo que cada célula ocupe 8 bytes de memória, como é comum para implementações modernas de ponto-flutuante.

problema, não parece ser justificável a escolha de um valor fixo para o número de relações em um determinado par de categorias.

3.6 Avaliação de métodos de extração

A literatura dos métodos de extração de relações semânticas não apresenta uma forma clara de medir a performance de um extrator. Na maioria dos trabalhos é apresentada a precisão do algoritmo, calculada ao se rotular manualmente o resultado dos métodos. Outras formas de avaliação também são utilizadas, e serão discutidas nessa seção.

Em (ETZIONI et al., 2005), o termo “revocação” é re-definido da seguinte forma: “Já que não podemos computar a ‘verdadeira revocação’ da Web, esse trabalho usa o termo ‘revocação’ para se referir ao tamanho do conjunto dos fatos extraídos.”⁶ O termo também é re-definido em (HASEGAWA; SEKINE; GRISHMAN, 2004) como: “Quantos pares corretos são detectados dentre todos os pares-chave? A quantidade de pares-chave é definida como o total de pares manualmente classificados em grupos de dois ou mais pares.”⁷, e utilizado para calcular o F-Score.

No OntExt, são relatadas 115 relações corretas entre 252 promovidas, caracterizando assim uma precisão de 45.6%. No entanto essa precisão não é discutida; em vez disso, é construído um classificador como etapa de pós-processamento, utilizando essas 252 relações como instâncias rotuladas, e são discutidos os números de precisão, revocação, e área sob a curva ROC desse classificador. A precisão deste classificador pode ser considerada como a precisão real do método, pois contabiliza o total de instâncias promovidas. A revocação, por outro lado, é em relação somente aos pares já promovidos pela etapa de agrupamento, e não é representativa da revocação real do método. Similarmente, o cálculo da curva ROC é feito em relação à revocação do classificador, e não do método de ER como um todo.

Essas limitações são causados por métricas tradicionais utilizadas não se aplicarem no contexto de extração de relações. Como mencionado por (ETZIONI et al., 2005), não é possível calcular a revocação, e a redefinição do termo em função de algo que é possível ser calculado não altera o fato que não se responde a pergunta “de todas as relações possíveis, quantas o método encontrou”? Em especial, calcular métricas derivadas, como a média harmônica F-score, a partir de um termo redefinido carrega pouco significado.

É discutível se tal pergunta, a definição real da revocação, é possível de ser respondida em primeiro lugar. Embora seja possível, em um determinado ponto do tempo e para uma língua específica, listar todos os sintagmas verbais existentes, mesmo que tal tarefa seja um esforço além do factível, ao se considerar neologismos, estrangeirismos, e outras possíveis evoluções da língua, tal esforço seria em vão.

⁶ No original: Since we cannot compute “true recall” on the Web, the paper uses the term “recall” to refer to the size of the set of facts extracted.

⁷ No original: How many correct pairs are detected out of all the key pairs? The key pair count, N_{key} , is defined as the total number of pairs manually classified in clusters of two or more pairs.

Para calcular a precisão, é necessário esforço humano para classificar cada relação promovida como correta ou incorreta, mas é uma métrica apropriadamente calculada, e bastante relevante para a aplicação da NELL, dado a arquitetura iterativa apresentada no Capítulo 2. Apesar de não ser possível calcular a revocação de cada método, se vários métodos são aplicados sobre os mesmos dados de entrada, tanto pares de categoria como *corpus* e listas de instâncias de cada categoria, é possível fazer afirmações sobre a revocação relativa deles, ou seja, qual deles tem a maior revocação. Uma vez que as relações promovidas são classificadas como correta ou incorreta, a quantidade absoluta de relações classificadas como corretas é diretamente proporcional à revocação; já que o denominador da revocação é o mesmo para todos os métodos, podemos compará-los utilizando somente o numerador.

No caso específico em que todos os métodos de ER sendo utilizados dêem uma nota para cada relação, ou seja, são ranqueadores, ainda é possível utilizar uma métrica mais especializada, a “precisão @ N ”, comumente utilizadas em sistemas de recomendação. Essa métrica é interessante, pois quando avaliadas em diferentes valores de N permite escolher um balanço apropriado entre a precisão e a revocação. Os dois métodos utilizados nesse trabalho, o OntExt e o HCSw, apresentado no Capítulo 4, não são ranqueadores, portanto somente as métricas de precisão e total de relações corretas são utilizadas.

4 Grafo de co-ocorrência

A primeira abordagem para resolver os problemas identificados na OntExt foi tratar a matriz de co-ocorrência como uma matriz de adjacência. De fato, a matriz de co-ocorrência antes da etapa de normalização, como descrita em (MOHAMED; Hruschka Jr; MITCHELL, 2011), é uma matriz quadrada, simétrica, preenchida com números inteiros, pouco densa, e onde a diagonal principal não tem uma definição clara. Essas propriedades fazem com que essa matriz possa ser considerada a representação de um grafo através de uma matriz de adjacência. As arestas desse grafo não possuem direção mas possuem pesos associados. A presença de *loops* (arestas que ligam um nó a ele mesmo) são opcionais dependendo do método a ser aplicado. Para o experimento deste capítulo, não foram gerados *loops*.

A principal vantagem de se considerar um grafo, nesse caso, é a localidade. Como a matriz de co-ocorrência pode ser grande, métodos baseados em métricas de distância, como o K-Means, podem sofrer da chamada “maldição da dimensionalidade”, um fenômeno em que métricas de dissimilaridade tornam-se menos eficientes conforme o número de dimensões aumenta (AGGARWAL; HINNEBURG; KEIM, 2001).

Métodos baseados em grafos, por outro lado, podem usar estruturas locais que não são afetadas pela quantidade de vértices no grafo, que seria equivalente ao número de dimensões na matriz de co-ocorrência. Um exemplo dessa localidade é calcular os componentes conexos do grafo assim que criado, permitindo uma separação altamente semântica de grupos. Foi visto na prática que diversas relações acabam criando múltiplos componentes conexos, e semanticamente instâncias de diferentes componentes não deveriam cair no mesmo grupo, propriedade que o K-Means não garante.

Um dos algoritmos mais simples de agrupamento em grafos é o algoritmo HCS (*Highly Connected Subgraphs*), descrito na Seção 4.1. Na Seção 4.2 é apresentada uma modificação do HCS, chamada de HCSw (*HCS weighted*), que visa contornar uma das limitações do algoritmo original, permitindo que o grafo seja ponderado, onde o HCS não considera pesos nas arestas do grafo.

4.1 Algoritmo HCS

O algoritmo HCS realiza um particionamento dos vértices de um grafo, por recursivos mínimos cortes (*mincuts*), até que os vértices pertencentes a uma partição sejam “altamente conectados” ou uma partição tenha somente um vértice. Esse algoritmo foi inicialmente proposto em (HARTUV; SHAMIR, 2000).

A conectividade de arestas de um grafo G , $k(G)$, é definida como o mínimo número de arestas que devem ser removidas de G para que G passe a ter dois componentes conexos; ou seja, o número de arestas no conjunto do *mincut* de G . Um grafo é dito altamente

conectado se sua conectividade de arestas é maior que metade do seu número de vértices, $k(G) > \frac{|V(G)|}{2}$.

Dada uma função de *mincut*, como Stoer-Wagner (STOER; WAGNER, 1997), assumindo que essa função de *mincut* retorne (H_1, H_2, k) , onde k é a conectividade de arestas de G , e H_1 e H_2 são os dois sub-grafos criados pelo *mincut*, o HCS pode ser descrito como no Algoritmo 1, onde a função *Compor* simplesmente une dois componentes conexos em um único grafo desconexo.

Algoritmo 1: Implementação básica do HCS

Dados: Grafo G conexo a ser particionado

Resultado: Grafo \bar{G} com um componente conexo por partição

```

1  $(H_1, H_2, k) \leftarrow \text{mincut}(G)$ ;
2 se  $k > \frac{|V(G)|}{2}$  então
3   |  $\bar{G} \leftarrow G$ ;
4 senão
5   |  $G_1 \leftarrow \text{HCS}(H_1)$ ;
6   |  $G_2 \leftarrow \text{HCS}(H_2)$ ;
7   |  $\bar{G} \leftarrow \text{Compor}(G_1, G_2)$ ;
8 fim
```

É notado que o HCS pode ser computacionalmente caro caso os vértices tenham um grau baixo (HARTUV; SHAMIR, 2000). Uma forma de aliviar o problema é iterativamente remover os vértices com o mínimo grau antes de aplicar o HCS, e somente aplicar o HCS recursivamente se uma partição de tamanho dois ou maior foi encontrada. Essa técnica evitaria a chamada da função de *mincut* múltiplas vezes para encontrar um corte de “singleton” (único vértice) repetidamente.

4.2 Algoritmo HCSw

Uma das limitações do HCS é não considerar arestas com pesos. Em (SHARAN; SHAMIR, 2000) é proposta uma variante chamada CLICK. Esse algoritmo apresenta uma abordagem estatística para o corte das arestas, assumindo uma distribuição gaussiana nos dados de um grupo.

A principal diferença do CLICK em relação ao HCS é a condição de parada da recursão. É definido um *kernel* como um conjunto de vértices que é um sub-conjunto de somente um grupo. Assim, a recursão do HCS é terminada quando os vértices daquele sub-grafo formam um *kernel*. Em um estágio posterior, vários *kernel* são unidos para formar um grupo.

Essa condição de parada, no entanto, é demasiadamente restrita, e assume uma determinada distribuição sobre os elementos de um grupo. Nesse trabalho foi implementada

uma outra variante, similar ao CLICK, que não atende todas as propriedades matemáticas que o CLICK possui, mas requer um esforço computacional menor.

Essa nova variante, chamada HCSw (*HCS weighted*), altera a condição de parada do HCS para considerar o particionamento como terminado quando o peso do *mincut* não excede a quantidade de vértices do grafo, multiplicado por uma constante, K . Essa nova constante funciona como um hiper-parâmetro, com um valor sugerido de $K = \frac{W(G)}{2|E(G)|}$, onde $W(G)$ é o peso total de todas as arestas do grafo; ou seja, o valor seria metade do peso médio das arestas do grafo. Esse valor sugerido faz com que no caso degenerado onde todas as arestas possuem o mesmo peso, o algoritmo funcione exatamente como o HCS. Nesse modelo, a nova definição de “altamente conectado” passa a ser $W(C) > |V(G)|K$, com $W(C)$ sendo a soma dos pesos das arestas do *mincut*.

Para reduzir o número de arestas no grafo, com o intuito de reduzir o tempo computacional, foi executado o algoritmo de chaves esparsas “Spanner” (BASWANA; SEN, 2007), utilizando o parâmetro de extensão igual a cinco. Esse é um algoritmo aleatório e em tempo linear que remove arestas pouco significativas para a conexão entre dois vértices; pouco significativas nesse contexto quer dizer que o peso total da caminhada entre os dois vértices não aumenta por um multiplicador maior que o parâmetro de extensão. Embora não seja garantida a otimalidade, como no caso do *mincut*, o algoritmo é executado somente uma vez por grafo, melhorando o tempo de execução, e intuitivamente não reduz significativamente a performance do HCSw, já que ambos visam remover arestas pouco significativas.

Uma das propriedades interessantes do OntExt é gerar nomes válidos para as relações promovidas. No HCSw, para definir o nome da relação é utilizada a métrica de grau de centralidade em cada componente conexo após a etapa de agrupamento. Ou seja, é utilizado como ponto central o nó que se conecta o máximo a outros nós, também chamado de *hub* (BARABÁSI, 2016). Embora não seja claro que a estrutura dos componentes conexos sejam livres de escala, a definição de ponto central em um grafo aleatório é mais complexa.

O OntExt gera uma nota para cada par de (*Sujeito, Objeto*), ranqueando as melhores instâncias de relações, baseando-se nos vetores de cada contexto e seus respectivos centroides. Como o HCSw não possui o conceito de centroide essa nota foi adaptada, utilizando a frequência relativa de cada par (*Sujeito, Objeto*) em relação aos grupos (e não contextos). Isto é, todos os pares são pontuados de acordo com o número de ocorrências com os contextos do grupo, dividido pelo total de ocorrências do par. Assim, pares que são frequentes em outros grupos recebem notas menores. Cada grupo promove 50 pares, um número arbitrário escolhido para ser consistente com o OntExt, que executa o mesmo corte. No entanto os pares são mantidos somente no grupo em que mais ocorreram, fazendo com que 50 seja o limite superior de instâncias em cada relação. Isso faz com que alguns grupos não tenham nenhum par de instâncias; esses grupos podem ser removidos como *outliers*.

A seção seguinte apresenta um experimento comparando os métodos HCSw e OntExt em um conjunto de dados da NELL, gerado pelo Prophet, como mencionado no Capítulo 3. Algumas comparações podem ser feitas a nível estrutural, antes do experimento. O OntExt utiliza o algoritmo de agrupamento K-Means, que é um particionador completo, colocando todos os elementos em um e somente um conjunto. O HCSw por outro lado não coloca todos elementos em um conjunto, optando por remover alguns elementos, mas qualquer elemento promovido é colocado em somente um conjunto. Dessa forma, o HCSw pode ser considerado mais flexível com *outliers*, não atribuindo contextos muito distintos como sinônimos da mesma relação.

Outra diferença significativa é a quantidade de relações promovidas por par de categoria. O OntExt usa o algoritmo K-Means com $K = 5$, promovendo até cinco relações por par de categoria¹. É possível alterar esse valor, até mesmo calculado heurísticamente a partir dos dados, mas não há um método claro de definir K sem testes empíricos, como feito para encontrar o valor cinco, que dependem dos próprios dados, e não necessariamente são generalizáveis para outros conjuntos de dados. Por outro lado, o HCSw não assume nenhuma propriedade sobre a quantidade total de grupos existentes, em vez disso usando uma definição mais localizada do que é um grupo altamente conexo.

4.3 Experimento

Foi realizado um experimento com cem pares de categorias promovidos pelo componente Prophet como potenciais pares com relações². Para todos os pares de categoria, foram executados tanto o K-Means sobre a matriz de co-ocorrência, como o HCSw sobre o grafo de co-ocorrência. Os dois métodos foram configurados para produzir como saída uma lista de relações, cada relação indicando o número de contextos sinônimos e uma lista de pares promovidos. Dois humanos independentemente classificaram cada relação como correta ou incorreta. Esses dois juízes tiveram uma taxa de concordância de 65,6%, com coeficiente κ 0,33, devido à natureza subjetiva de se classificar relações em língua natural. Segundo (LANDIS; KOCH, 1977), este valor do coeficiente κ é suficiente para dizer que a concordância é “justa”, embora o próprio autor diga que classificações da estatística são arbitrárias.

Na Tabela 2 são mostrados os resultados do experimento. A versão original do OntExt possui uma etapa de pós-processamento utilizando um classificador, mas como um artifício similar não foi desenvolvido para o HCSw (apesar de que nenhuma propriedade parece impedir tal método), os algoritmos foram comparados sem pós-processamento. O número de instâncias promovidas difere entre os dois juízes pois estes não eram obrigados a

¹ O único caso onde não são levantadas cinco relações é quando há menos que cinco sintagmas verbais encontrados.

² O código-fonte do experimento, em Python, é disponibilizado no repositório <https://github.com/diorge/cooccurrence-relationship-extraction>

classificar todas as relações, dado que algumas relações poderiam requerer conhecimento de domínios específicos³.

Um caso especial das relações promovidas são os *singletons*, sintagmas verbais sem sinônimos encontrados. Ao filtrar tais relações, há uma diferença significativa na precisão somente nas classificações do juiz 1, nas relações geradas pelo HCSw. Na execução do OntExt, 43,2% das relações encontradas eram *singletons*, enquanto no HCSw esse número aumenta para 91,9%. A quantidade de *singletons* por si não é um problema, desde que estas relações sem sinônimos estejam corretas. Embora seja intuitivo que dentre tais relações ocorra uma taxa maior de incorretas, por possuírem menos evidências de que sejam corretas, isso não parece ocorrer com o OntExt. No HCSw também não é claro que isso ocorre; embora a precisão aumente significativamente para o juiz 1, o mesmo não acontece com o juiz 2, o que indica que há outro fator não identificado.

	OntExt	OntExt sem <i>singletons</i>	HCSw	HCSw sem <i>singletons</i>
Corretas (E1)	136	77	323	37
Promovidas (E1)	285	162	827	67
Precisão (E1)	47,7%	47,5%	39%	55,2%
Corretas (E2)	161	91	476	40
Promovidas (E2)	275	156	808	64
Precisão (E2)	58,5%	58,3%	58,9%	62,5%

Tabela 2 – Performance dos métodos OntExt e HCSw, para os mesmos 100 pares de categoria, de acordo com dois humanos juízes (E1 e E2)

Os resultados do OntExt são condizentes com o que foi apresentado em (MOHAMED; Hruschka Jr; MITCHELL, 2011), onde são relatadas 115 relações corretas dentre 252 relações, uma precisão de 45,6%. Em geral, não é possível dizer que um dos métodos, OntExt e HCSw, seja melhor que o outro em termos de precisão, pois a própria variação entre os métodos, com diferença máxima de 39% para 47,5% (11,5 pontos percentuais, ou 82,1% do valor), é equivalente a variação entre os humanos, com diferença máxima de 47,5% para 58,3% (10,8 pontos percentuais, ou 81,5% do valor).

É possível, no entanto, dizer que o HCSw possui uma revocação maior que o OntExt, desde que não sejam removidos os *singletons*. Tal afirmação é decorrência da precisão similar, e do fato que o OntExt gera no máximo cinco relações por par de categoria, sendo assim no máximo 500 relações nesse experimento, enquanto o HCSw produziu 827 relações. É possível fazer o OntExt gerar mais relações por par de categoria alterando o hiper-parâmetro K do K-Means, mas como notado em (MOHAMED; Hruschka Jr; MITCHELL, 2011), a configuração $K = 5$ tem a maior precisão empírica.

Se dois métodos possuem a mesma precisão, aquele que gera o máximo possível de relações é preferível. Isso não só vem do fato de que a base de conhecimento cresce

³ O juiz 2 ignorou todas as relações com a categoria *mldata*, “conjuntos de dados de aprendizado de máquina”; as demais relações foram todas preenchidas por ambos os juízes.

mais rapidamente, e com a mesma qualidade pois a precisão se mantém, mas também pela possibilidade de mais facilmente controlar o balanço entre precisão e revocação. Assumindo que seja possível criar uma etapa de classificação como pós-processamento, se esse classificador funciona através de atribuir notas a cada instância, como uma regressão logística ou *Naïve-Bayes*, ou seja, um ranqueador, então é possível escolher diferentes limiares para a classificação de uma relação como correta ou incorreta, diretamente afetando o balanço entre precisão e revocação. É esperado que mais relações preditas como incorretas pelo classificador cause um aumento da precisão em troca de uma redução na revocação, e o oposto é esperado quando menos relações são preditas como incorretas.

5 Geração da estrutura de co-ocorrência

Atualmente, a NELL não executa o OntExt como componente devido ao custo computacional do método. O maior custo existente é na construção da matriz de co-ocorrência. Neste capítulo, é apresentada uma análise do custo computacional do OntExt, e técnicas utilizadas para tornar sua execução viável, em especial aquelas de processamento distribuído. Nota-se que tanto a matriz de co-ocorrência como o grafo de co-ocorrência são criados a partir dos mesmos dados, não há uma diferença significativa no custo de criação entre as duas representações.

O primeiro passo a se considerar são as computações que não dependem das duas categorias a serem buscadas. Ou seja, etapas de pré-processamento, executadas antes de qualquer chamada ao OntExt. Esses custos são amortizados ao longo do tempo, sendo executados uma única vez, independente de quantos pares de categorias forem ser processados. Naturalmente, alterar o conjunto de dados SVO de entrada implica em realizar essas etapas de pré-processamento novamente. É importante notar que o conjunto de dados utilizado nesse trabalho possui somente sentenças nos padrões “Sujeito-Verbo-Objeto” ou “Sujeito-Verbo-Preposição-Objeto”, que tende a possuir mais relações binárias (BANKO; ETZIONI, 2008).

Foram implementadas três etapas de pré-processamento. Inicialmente é feito um filtro sobre o número de ocorrências de uma sentença no *corpus*. Sentenças que ocorreram poucas vezes podem indicar erros de digitação ou expressões que não são comuns na língua. Utilizando este filtro para manter somente sentenças que ocorreram cinco ou mais vezes em um *corpus* inicial com 220 milhões de sentenças distintas resultou em 6,5 milhões de sentenças distintas. Similarmente, a segunda etapa mantém somente sentenças cujo par (*Sujeito*, *Objeto*) ocorreu pelo menos cinco vezes, com a mesma motivação do filtro anterior. O número de ocorrências cinco é o mesmo utilizado em (MOHAMED; Hruschka Jr; MITCHELL, 2011). Isso resulta em um conjunto de dados com 689 mil sentenças distintas.

Por fim, uma terceira e última etapa é ordenar essas 689 mil sentenças. Essa etapa não é um filtro de pré-processamento propriamente dito, mas facilita a computação do particionamento, uma etapa posterior. Uma característica é que a ordenação das sentenças deve ser feita primeiro pelo sujeito, depois pelo objeto, e por último no campo do verbo. Isso permite um particionamento mais eficiente, pois um determinado par (*S*, *O*) sempre está nas sentenças adjacentes.

Esse novo conjunto de dados, com 689 mil sentenças já ordenadas, pode ser diretamente utilizado pelo OntExt. Considerando que cada execução do método utiliza somente um par de categorias, é possível paralelizar a execução de múltiplos pares de categorias trivialmente. No entanto, o consumo de memória RAM nesse caso pode ser excessivo, já que nenhuma

informação é compartilhada entre as execuções. O foco dessa seção é sobre o processamento de um único par de categorias.

Uma vez que o conjunto de dados é filtrado somente para sentenças que ambos sujeito e objeto pertençam às categorias desejadas, um último filtro é executado. Essa etapa remove sentenças cujo sintagma verbal não acontece com um número suficiente de pares (*Sujeito, Objeto*). O objetivo desse filtro é remover casos em que ou o objeto ou o sujeito estão incorretamente inseridos em suas respectivas categorias, ou possuem homônimos em outra categoria. Por exemplo, caso esteja sendo processo o par de categorias (*Pessoa, Roupa*), a sentença “João come manga” será mantida utilizando os filtros anteriores, mas poucas outras sentenças utilizarão o verbo “come”. Dessa forma, todas as sentenças desse verbo podem ser removidas. É utilizado o limiar de pelo menos três ocorrências distintas do sintagma verbal para que ele seja mantido, assim como na descrição original do algoritmo (MOHAMED; Hruschka Jr; MITCHELL, 2011).

As sentenças restantes são utilizadas para montar a matriz ou grafo de co-ocorrência para o par de categorias; o processo para as duas representações é o mesmo, e daqui em diante será discutido somente em termos da matriz. A forma de execução aqui descrita assume um processamento distribuído dos dados. Tal técnica permite que sejam utilizados vários computadores para o cálculo da matriz, facilitando o acesso a grandes quantidades de memória RAM, e aumentando a escalabilidade do método.

Para realizar a distribuição dos dados, é necessário criar um esquema de particionamento. Como todas as computações partem do subconjunto de sentenças definidos por um par (*Sujeito, Objeto*), é possível fazer dois tipos de particionamento, por um dos dois elementos (sujeito ou objeto), criando um particionamento menos granular, ou utilizando os dois elementos do par, sendo assim um particionamento mais granular. Em geral, particionamentos mais granulares oferecem melhor escalabilidade, mas podem acarretar em maiores custos de manutenção e sincronização (*overhead*). A escolha entre as duas granularidades depende principalmente do tamanho do conjunto de dados, das especificações das máquinas que farão o processamento, e a quantidade de máquinas disponíveis, mas podem haver outros fatores práticos que afetem a eficiência (sistema operacional, implementações de bibliotecas, entre outros).

Assumindo o particionamento mais granular¹, podemos estruturar o problema da seguinte maneira: cada partição gera uma sequência de pares de sintagmas verbais (v_1, v_2), e o valor da matriz de co-ocorrência na posição M_{v_1, v_2} é igual a contagem de ocorrências de (v_1, v_2) na sequência gerada ao se concatenar a sequência de pares de todas as partições.

Esse problema pode ser visualizado através do *framework MapReduce* (DEAN; GHEMAWAT, 2008). Nesse caso, transformar a partição em uma sequência de pares de sintagmas verbais é a função de mapeamento, enquanto a contagem das ocorrências

¹ Particionamentos mais grossos podem ser quebrados em repetidamente executar o mais granular; como não há dependência de dados, a descrição deste processo é idêntica para ambos os casos.

de pares de sintagmas é a função de redução (ou função agregadora). Essa função de mapeamento pode ainda ser descrita como uma auto-junção (“*self-join*”) da partição, se visualizada como uma tabela de dados, filtrando as ocorrências de modo combinatório. Uma forma mais fácil de entender essa solução é utilizando a linguagem SQL para descrevê-la; a Figura 2 mostra a consulta SQL que monta a matriz de co-ocorrência no formato de matriz esparsa por mapa de coordenadas, formato comumente utilizado (BELL; GARLAND, 2009).

```

1 -- assumindo que existe uma tabela 'dataset'
2 -- com as colunas (s, v, o)
3 SELECT
4     d1.v as left_verb,
5     d2.v as right_verb,
6     COUNT(*)
7 FROM
8     dataset d1,
9     dataset d2
10 WHERE
11     d1.s = d2.s
12     AND d1.o = d2.o
13     AND d1.v < d2.v -- operador '<=' caso queira
14                     -- preencher a diagonal principal
15 GROUP BY
16     left_verb, right_verb

```

Figura 2 – Consulta SQL gerando os valores das coordenadas da matriz de co-ocorrência

A consulta SQL presente na Figura 2 gera o mapa de coordenadas da matriz de co-ocorrência, mas sem conhecimento dos dados pode não conseguir otimizar e paralelizar a execução. Foi desenvolvido um código em PySpark² que executa essa mesma consulta, mas permitindo que cada partição seja executada em um *worker* independente, de acordo com as configurações do *Spark Cluster* que for executar o código. Essa nova implementação permite que a geração da matriz de co-ocorrência seja escalável em relação ao número de sentenças do conjunto de dados.

Para encontrar o esforço computacional dessa abordagem, partindo do conjunto de dados já filtrado e ordenado, incluindo o filtro executado após encontrar somente as sentenças relevantes para a relação, isto é, aquelas em que o objeto e o sujeito são do par de categorias buscado, utiliza-se a seguinte análise. O tempo de execução do particionamento dos dados cresce de forma linear em relação ao total de sentenças do conjunto de dados filtrado. O executor de cada partição precisa somente calcular as possíveis combinações de sintagmas verbais dentro de sua partição, que tomando exatamente $\frac{n(n+1)}{2}$ iterações, uma solução assintoticamente quadrática ($O(n^2)$), com n sendo o número de elementos na

² Disponibilizado no repositório <https://github.com/diorge/cooccurrence-relationship-extraction>

partição. A última execução do processo é a agregação dos resultados de cada partição. O agregador precisa passar uma vez por todas as coordenadas de cada partição, sendo uma solução linear em relação ao número de coordenadas, que é a somatória de todas as células da matriz de adjacência.

Uma estimativa do tempo computacional total, desconsiderando custos de *overhead*, pode ser feito utilizando as variáveis descritas na Tabela 3. O particionamento é feito em $O(r)$, p executores precisam de no máximo $\frac{t(t+1)}{2}$ iterações cada, sendo assim um custo somado de c , e o agregador faz um passe único em uma sequência de tamanho máximo c . O custo pode ser calculado como $O(r + c + c)$; utilizando os limites superiores, temos que $O(c) = O(p \cdot \frac{t(t+1)}{2}) = O(r \cdot r \cdot (r + 1)) = O(r^3)$. Logo, todo o processo quando colocado em escala é limitado pela assintótica de $O(r + r^3 + r^3) = O(r^3)$.

Variável	Descrição	Limite superior
r	Total de sentenças no conjunto de dados filtrado	
p	Total de partições (pares distintos (S, O))	$\frac{r}{3}$
t	Tamanho máximo de uma partição	$\frac{r}{3}$
c	Total de coordenadas (somatória de toda a matriz)	$p \cdot \frac{t(t+1)}{2}$

Tabela 3 – Variáveis relacionadas ao custo computacional

No entanto, esse esforço computacional é distribuído nos nós executores. A etapa de particionamento é executada em uma única máquina, a coordenadora que distribui a execução, e essa etapa é o menor custo. Para a etapa de mapeamento, o custo que cada nó executa no pior caso é $O(\frac{t(t+1)}{2}) = O(r^2)$. Finalmente, a etapa de redução é um pouco mais complexa; enquanto seu custo assintótico no pior caso permanece $O(r^3)$ e precisa ser executada em uma única máquina, na realidade o pior caso é raro, devido à esparsidade dos dados. Por exemplo, para o par de categorias *building* e *visualizablescene*, com $r = 1854$ (e portanto $r^3 \approx 6,3 \cdot 10^9$), a execução teve exatamente 1693 iterações, menor que o próprio valor de r .

Com essa nova forma de construir a representação de co-ocorrência, seja uma matriz esparsa ou grafo, o tempo necessário para a computação é por ordens de magnitude menor que a implementação anterior. A melhor implementação do antigo algoritmo era capaz de processar 15 pares de categorias em 25 minutos, utilizando um servidor com 256 GB de memória RAM (BROGNARA; Hruschka Jr, 2016). A versão atual, sendo executada em uma única máquina sem fazer uso do processamento distribuído, processou 100 pares de categorias em 15 segundos, ignorando o tempo de pré-processamento independente dos pares de categorias utilizados, que durou cerca de 10 minutos, utilizando um computador pessoal com 16 GB de memória RAM. A maior diferença entre os dois experimentos é que a nova versão consegue fazer todo o processamento em RAM, devido à utilização de estruturas esparsas, enquanto a antiga implementação ultrapassava o limite em memória com suas estruturas densas, e precisava fazer armazenamento temporário em disco rígido.

6 Considerações Finais

Neste trabalho, foi apresentado o problema de extração de relações, em especial sua aplicação dentro do sistema NELL. Foram estudados métodos de ER presentes na literatura, e para a aplicação desejada de encontrar novas relações sem a supervisão humana, o método OntExt, desenvolvido anteriormente para a NELL, mostrou-se mais adequado. No entanto, o método não é executado devido ao seu custo computacional.

Com o objetivo de encontrar um método similar ao OntExt que possa ser executado dentro da NELL, duas frentes foram atacadas. Uma foi a conversão da estrutura de matriz de co-ocorrência, como utilizado no OntExt, para uma representação de grafo. A outra pesquisa realizada foi a possibilidade de otimizar a criação da estrutura de co-ocorrência, seja matriz ou grafo, fazendo uso de processamento distribuído e da distribuição esparsa dessas estruturas.

É apresentada, na Seção 3.6, uma discussão sobre a forma como métodos de extração de relações semânticas podem ser avaliados. Não existe uma forma aceita na literatura de como os métodos podem ser avaliados e comparados, e cada trabalho apresenta seu próprio conjunto de métricas. É argumentado que a precisão e a revocação, esta sendo estimada pelo total de relações corretas promovidas, são as duas métricas mais relevantes para o problema. Ainda é possível, em alguns casos, utilizar a métrica “precisão @ N”, abstraindo a precisão e revocação para uma única métrica, mas esta requer que os métodos utilizados produzam valores de confiança para cada relação promovida.

Representar a estrutura de co-ocorrência como um grafo foi discutido no Capítulo 4. Considerando a estrutura do problema, foi necessário adaptar um algoritmo de agrupamento em grafos já existe, em um novo método chamado HCSw. Essa nova representação permite uma avaliação semântica e local de algumas propriedades, por exemplo realizando um agrupamento inicial por componentes conexas do grafo. Diferente do OntExt, esse novo método não tem um limite de relações encontradas por par de categoria. Foi realizado um experimento, comparando o OntExt e o HCSw no mesmo conjunto de dados. Constatou-se que os métodos possuem precisão similar, enquanto o HCSw possui uma revocação maior, assim como um maior tempo de processamento.

A nova forma de construir a matriz de co-ocorrência é apresentada no Capítulo 5. Foi desenvolvida uma forma distribuída de fazer o processamento da estrutura, devidamente particionando o conjunto de dados, e definindo processos independentes que não necessitam compartilhar informações intermediárias, garantindo assim uma maior escalabilidade do método ao permitir que diversas máquinas de pequeno porte dividam igualmente o processamento. A implementação anterior da OntExt processou 15 pares de categorias em 25 minutos, utilizando um servidor de médio porte, enquanto a versão atual processou 100 pares de categorias em 15 segundos, em um computador pessoal.

O experimento realizado no Capítulo 4 não aplica etapas de pós-processamento, como o OntExt originalmente faz. O desenvolvimento de uma etapa de pós-processamento para o HCSw é interessante, pois seria possível a escolha de limiares de balanço entre precisão e revocação do método, assim como o cálculo da métrica “precisão @ N”.

Não é possível concluir, na implementação utilizada, qual dos dois métodos é mais vantajoso ser aplicado na NELL. Segundo o experimento realizado, a precisão não difere significativamente, e existe uma escolha entre a maior revocação do HCSw ou o menor tempo de processamento do OntExt. É interessante, no entanto, a aplicação do novo método de criação da estrutura de co-ocorrência, que faz uso da esparsidade do problema para reduzir o custo computacional significativamente.

Referências

- AGGARWAL, C. C.; HINNEBURG, A.; KEIM, D. A. On the surprising behavior of distance metrics in high dimensional space. In: SPRINGER. *International conference on database theory*. [S.l.], 2001. p. 420–434.
- AGICHTEN, E.; GRAVANO, L. Snowball: Extracting relations from large plain-text collections. In: ACM. *Proceedings of the fifth ACM conference on Digital libraries*. [S.l.], 2000. p. 85–94.
- BANKO, M. et al. Open information extraction from the web. In: *IJCAI*. [S.l.: s.n.], 2007. v. 7, p. 2670–2676.
- BANKO, M.; ETZIONI, O. The tradeoffs between open and traditional relation extraction. *Proceedings of ACL-08: HLT*, p. 28–36, 2008.
- BARABÁSI, A.-L. The scale-free property. In: *Network science*. [S.l.]: Cambridge university press, 2016. cap. 4.
- BARCHI, P. H. et al. *Expansão de Ontologia através de leitura de máquina contínua*. Dissertação (Mestrado) — Universidade Federal de São Carlos, 2014.
- BASWANA, S.; SEN, S. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, Wiley Online Library, v. 30, n. 4, p. 532–563, 2007.
- BELL, N.; GARLAND, M. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: *Proceedings of the conference on high performance computing networking, storage and analysis*. [S.l.: s.n.], 2009. p. 1–11.
- BROGNARA, D.; Hruschka Jr, E. R. *Aprimoramento na geração de matrizes de co-ocorrência*. São Carlos: Departamento de Computação, Universidade Federal de São Carlos, 2016. Trabalho de conclusão de curso.
- CARLSON, A. et al. Coupling semi-supervised learning of categories and relations. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*. [S.l.], 2009. p. 1–9.
- CARLSON, A. et al. Toward an architecture for never-ending language learning. In: ATLANTA. *AAAI*. [S.l.], 2010. v. 5, p. 3.
- DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, ACM New York, NY, USA, v. 51, n. 1, p. 107–113, 2008.
- ETZIONI, O. et al. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, Elsevier, v. 165, n. 1, p. 91–134, 2005.
- HARTUV, E.; SHAMIR, R. A clustering algorithm based on graph connectivity. *Information Processing Letters*, v. 76, n. 4-6, p. 175–181, dez. 2000. ISSN 00200190. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S0020019000001423>>.

- HASEGAWA, T.; SEKINE, S.; GRISHMAN, R. Discovering relations among named entities from large corpora. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. [S.l.], 2004. p. 415.
- LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. *biometrics*, JSTOR, p. 159–174, 1977.
- MOHAMED, T. P.; Hruschka Jr, E. R.; MITCHELL, T. M. Discovering relations between noun categories. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. [S.l.], 2011. p. 1447–1455.
- NAKASHOLE, N.; WEIKUM, G.; SUCHANEK, F. PATTY: a taxonomy of relational patterns with semantic types. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. [S.l.]: Association for Computational Linguistics, 2012. p. 1135–1145.
- NAVARRO, L. F. et al. *Mining ontologies to extract implicit knowledge*. Dissertação (Mestrado) — Universidade Federal de São Carlos, 2016.
- PAWAR, S.; PALSHIKAR, G. K.; BHATTACHARYYA, P. Relation extraction: A survey. abs/1712.05191, 2017. Disponível em: <<http://arxiv.org/abs/1712.05191>>.
- SHARAN, R.; SHAMIR, R. CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis. In: . [S.l.]: AAAI Press, 2000. p. 307–316.
- SODERLAND, S. et al. Adapting Open Information Extraction to Domain-Specific Relations. *AI Magazine*, v. 31, n. 3, p. 93, jul. 2010. ISSN 0738-4602, 0738-4602.
- STOER, M.; WAGNER, F. A simple min-cut algorithm. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 44, n. 4, p. 585–591, 1997.
- WIJAYA, D. T. *VerbKB: A Knowledge Base of Verbs for Natural Language Understanding*. Tese (Doutorado) — Ph. D. Dissertation, Carnegie Mellon University, 2016.
- WILKS, Y. Information extraction as a core language. In: *International Summer School on Information Extraction*. [S.l.: s.n.], 1997.