**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

João Filipe da Costa Nunes

# Elaboração Semiautomática de uma Ontologia para Remédios a Partir de Textos Antigos de Medicina e de Culinária

December 2021

**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

João Filipe da Costa Nunes

# Elaboração Semiautomática de uma Ontologia para Remédios a Partir de Textos Antigos de Medicina e de Culinária

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
**Orlando Manuel de Oliveira Belo**
**Anabela Leal de Barros**

December 2021

## DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico.

Eu confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

## RESUMO

As ontologias podem ser descritas como uma estrutura de dados com uma semântica concisa, explícita e central. As ontologias têm sido utilizadas com o intuito de representar um domínio de conhecimento, ou pelo menos parte desse domínio, de forma a que, posteriormente, possam ser efetuadas interrogações sobre si, visando a exploração, de forma equilibrada e sustentada, do conhecimento representado nessa estrutura. Desta forma, facilmente poderá ser descoberto e explorado o conhecimento presente num dado domínio que se pretende representar e descobrir através da utilização deste tipo de estrutura.

O processo manual de construção de uma ontologia envolve diversos custos, nos quais se pode destacar, em particular, o custo temporal associado à concepção e implementação de uma ontologia para um grande domínio de conhecimento. É neste contexto que se insere a área da *Ontology Learning*, na qual são integrados vários métodos de diferentes áreas de machine learning no processo de construção de ontologias, com o intuito de o (semi)automatizar. Dentro das áreas de *machine learning* poderá ser destacada a área de *Natural Language Processing*, na qual podemos encontrar um conjunto muito diverso de técnicas que nos permitem identificar e extrair termos/conceitos pertinentes, propriedades e relações a partir de um domínio de conhecimento, representado, por exemplo, na forma de um texto. Dentro das diversas técnicas passíveis de serem aplicadas, pode-se destacar a aplicação de padrões léxico-sintáticos, que visam explorar formalidades linguísticas com o objetivo de retirar pares hiperónimo e hipónimo de maneira a identificar conceitos e relações da ontologia. Esta abordagem foi aplicada num conjunto de textos antigos de culinária, agricultura e medicina dos séculos XVI e XVII, escritos em Português antigo. Após termos criado a ontologia, desenvolvemos um sistema capaz de explorar o conhecimento que ela contém, dando particular atenção à exploração dos diferentes remédios, dos ingredientes que compõem estes remédios, e dos processos de preparação desses mesmos ingredientes. Com este sistema, facultámos uma maneira simples e intuitiva de explorar o conhecimento presente nos diversos medicamentos representados e caraterizados na ontologia desenvolvida.


**hiperónimo- palavra cujo significado, por ser mais geral, inclui o de outra(s) palavra(s) (exemplo: legume é hiperónimo de cenoura)**


PALAVRAS-CHAVE Ontologias, Ontology Learning, Extração semiautomática, Machine Learning, Natural Language Processing, Padrões léxico-sintáticos.


**hipónimo- palavra cujo significado está incluído no de outra, cujo sentido é mais geral (exemplo: cenoura é hipónimo de legume)**

## ABSTRACT

Ontologies can be described as a data structure with concise, explicit, and central semantics. Ontologies have been used in order to represent a knowledge domain, or at least part of that domain, so that later queries can be made over itself, that aim to explore, in a balanced and sustained way, the knowledge represented in this structure. By using this structure, the knowledge that is represented in a certain domain can easily be discovered and explored through methods that can be used to explore this type of structure.

The process of manually creating an ontology is filled with costs, in which we can highlight, in particular, the time cost associated with the conception and implementation of an ontology which covers a large domain of knowledge. In light of this issue, the area of Ontology Learning aims to cover these issues, where methods from different areas of machine learning are integrated in the ontology construction process, with the aim of (semi)automating. Within the machine learning areas, the Natural Language Processing area can be highlighted, where a large set of techniques can be applied in order to identify and extract relevant terms/concepts, properties and relationships from a domain of knowledge, represented for example in the form of a text. Among the various techniques that can be applied, one such technique can be through the application of lexical-syntactic patterns, which aim to explore linguistic formalities with the goal of extracting hypernym and hyponym pairs so that the relationships, concepts can be identified. This approach has been applied to a set of classical culinary, agricultural, and medical texts from the XVI and XVII centuries, written in classical Portuguese. After having created the ontology, a system was developed so that the knowledge contained within it could be explored, giving particular attention to the different remedies, the ingredients contained within the remedies, and the preparation process of these ingredients. With this system, an easy and intuitive way to explore the knowledge present in the different medicines represented within the ontology was developed.



KEYWORDS Ontologies, Ontology Learning, Semi-automatic extraction, Machine Learning, Natural Language Processing, Lexical-syntactic patterns.

# CONTEÚDO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# 1

# INTRODUÇÃO

## 1.1 ENQUADRAMENTO E MOTIVAÇÃO

Uma ontologia representa uma estrutura de dados com uma semântica concisa, explícita e central. A utilização de uma ontologia tem o intuito de representar um certo domínio de conhecimento, podendo esta representação ser total ou parcial, dependendo das necessidades em causa. Depois, tendo o domínio representado, será então possível elaborar interrogações de maneira a explorar o conhecimento representado na ontologia.

Em traços gerais, uma ontologia é composta por classes (ou conceitos), que representam termos concretos ligados ao domínio de conhecimento que se pretende representar. As propriedades descrevem atributos relacionados com, por exemplo, as classes representadas. De maneira a estabelecer uma conexão entre as diferentes classes faz-se uso das relações. O relacionamento entre classes é preponderante no que toca a definir a estrutura base da própria ontologia, por exemplo, através da definição de uma hierarquia de conceitos. Os relacionamentos são também fulcrais no que concerne à posterior tarefa de exploração e na criação e exploração de mecanismos, de forma a inferir sobre a informação representada na estrutura da ontologia. Por fim, numa ontologia também podem também ser definidos axiomas, que representam regras lógicas que no contexto da estrutura são sempre verdadeiras.

As ontologias revelam-se como sendo bastante vantajosas no que concerne ao seu principal foco, e aplicação, da sua estrutura para realizar suposições e métodos de inferência sobre os dados. Estes processos, poderão possibilitar, por exemplo, a criação de métodos de visualização simples sobre os dados representados na estrutura. Relativamente à inserção de novos elementos, uma ontologia poderá facilmente ser alargada, devido à flexibilidade da estrutura, sendo muito escalável neste tipo de operações El Kadiri et al. (2015). Por fim, uma ontologia facilmente poderá representar num modelo uma vista geral sobre um dado domínio de conhecimento, podendo depois ser facilmente partilhada com outros elementos Noy and Mcguinness (2001).

Não obstante, o processo de criação de uma ontologia poderá ser um processo custoso, em particular, naquilo que diz respeito ao tempo gasto na conceptualização da ontologia, fruto dos diversos elementos que compõem a estrutura da ontologia. Ora, este tempo gasto na criação da ontologia, em certos casos, até poderá ser impeditivo, dada a complexidade do processo de conceptualização da ontologia imposto pela grande dimensão do domínio de conhecimento que se pretende representar. Portanto, para domínios de conhecimento complexos (ou grandes) existe um elevado custo no que toca à aquisição de conhecimento e, inclusive, poderá resultar num *bottleneck* Maedche and Staab (2001). Para além disso, em certos casos, a tarefa de idealização e criação da ontologia,

poderá também envolver custos manuais adicionais, uma vez que certos domínios necessitarão da supervisão de um ou mais especialista, para fazer a composição correta da ontologia.

Assim, surgiu a necessidade de automatizar ao máximo o processo de geração de forma semiautomática de ontologias. É neste contexto, então, que se insere a área de *Ontology Learning*, onde são aplicadas diversas técnicas de diferentes campos da área de *machine learning*, mais propriamente do campo de *Natural Language Processing*. Estas técnicas são utilizadas e combinadas de forma a permitir uma aprendizagem semiautomática dos diversos elementos que compõem uma ontologia a partir de um certo domínio de conhecimento. Com o uso destas técnicas elimina-se, ou pelo menos atenua-se, grande parte das desvantagens associadas com a criação de forma manual de uma ontologia. Contudo, o uso destas técnicas não implica que não existam tarefas manuais, como, por exemplo, a verificação e validação dos resultados de forma a ajustar parâmetros associados ao processo de extração com o intuito de melhorar a qualidade da ontologia final.

No âmbito deste trabalho, será exposta uma metodologia utilizada para permitir uma aprendizagem de forma semiautomática de uma ontologia de medicamentos. De forma a compor a ontologia, são extraídos os principais conceitos presentes nos textos bem como os seus relacionamentos. O domínio de conhecimento representado na ontologia provém de um conjunto antigo de textos de culinária, agricultura e medicina dos séculos XVI-XVII Barros (2013, 2014, 2016). Nestes textos é possível encontrar um conjunto de remédios destinados a suavizar condições do corpo humano, por exemplo, onde será possível encontrar um conjunto de ingredientes e métodos de confeção para aliviar certas doenças, ferimentos ou sintomas.

Depois, com a obtenção desta ontologia, pretende-se explorar, através da criação de um sistema especificamente desenvolvido para o efeito, os dados representados na ontologia de forma sustentada e efetiva. Nesse processo é pretendido que se explore os diferentes remédios, os ingredientes que compõem estes remédios, e os processos de preparação desses mesmos ingredientes.

Desta forma, através da extração da ontologia será possível evidenciar, de uma forma simples e conveniente, o processo de confeção de medicamentos utilizado na época, o que poderá servir depois como uma fonte de conhecimento para os dias de hoje. O sistema de exploração dos dados, de uma forma efetiva irá suportar os mecanismos de inferência e de exploração da rede semântica criada. Inclusivamente, em certos casos será também possível observar, por exemplo, relacionamentos entre certos remédios, através da partilha de ingredientes, que não seria tão facilmente observável sem o apoio da ontologia extraída.

## 1.2 TRABALHO REALIZADO

Após o estudo de diversos sistemas e ferramentas criadas nesta área, o processo de extração que implementámos nesta dissertação é composto por três passos principais.

Numa primeira fase do trabalho implementado foi necessário primeiramente estudar, de forma manual, os textos disponíveis, de maneira a melhor entender que tipo de processo é que poderia ser aplicado, para simplificar o texto, mas sem distorcer o seu significado, uma vez que este significado é crucial de forma a validar a ontologia final. Como tal, desenvolveu-se um módulo de pré processamento, no qual foram aplicados, por exemplo, padrões de substituição, de acordo com potenciais interferências negativas que estes padrões poderiam ter no

processo de extração. Com a utilização deste módulo, foi possível obter um texto corrido, simplificado, mas sem distorcer em demasia o significado original. O texto neste ponto encontra-se também dividido entre os vários medicamentos presentes neste.

Na segunda fase da implementação do sistema de extração, que corresponde à extração de conceitos bem como dos seus relacionamentos, optou-se pela utilização de padrões léxico-sintáticos. Os padrões léxico-sintáticos identificam pares hiperónimo/hipónimo no texto, que duplamente identificam os principais conceitos, estes pares foram também utilizados para estabelecer uma hierarquia de conceitos.

A última fase da criação do sistema correspondeu à concretização da própria ontologia. Para tal, foi necessário identificar potenciais soluções que poderiam ser utilizadas de forma a armazenar e representar os elementos extraídos. Optou-se por representar os resultados através do *Neo4J* Santos López and Santos De La Cruz (2015), uma base de dados *NoSQL*, orientada por grafos.

No que concerne à exploração dos dados, e de forma a permitir que os resultados do processo anterior possam ser estudados e analisados de uma forma simples e conveniente, optou-se pela utilização de tecnologias web. A tecnologia escolhida foi o *Vue.js*, de forma a criar um *frontend web* com o intuito de disponibilizar um conjunto de mecanismos de pesquisa sob os dados armazenados na base de dados. De forma a concretizar com o sistema de exploração de dados, optou-se por utilizar uma arquitetura baseada em micro serviços, em que o *backend* foi implementado através da utilização do *Node.js*.

## 1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Para além do presente esta dissertação incorpora mais cinco capítulos. No capítulo 2 apresenta-se um breve "estado de arte" sobre ontologias. Na secção 2.1 é introduzida a problemática das ontologias, na qual são fornecidas algumas definições utilizadas para descrever o conceito das ontologias, bem como a sua utilização como uma estrutura de dados. De seguida, na secção 2.2, são apresentados os diversos elementos que compõem uma ontologia, sendo também fornecido um exemplo concreto de uma ontologia. Na secção 2.3 são apresentados diversos ciclos de desenvolvimento, associados à criação de ontologias. Na secção 2.4 são apresentados alguns conceitos relativamente à tarefa de avaliação de uma ontologia. O capítulo é encerrado com as secções 2.5 e 2.6, nas quais são apresentados, respetivamente, sistemas que utilizam ontologias bem como ferramentas de apoio à criação desta estrutura de dados.

No capítulo 3, aborda-se o tópico da aprendizagem semiautomática de ontologias. O capítulo é iniciado com a secção 3.1 na qual é introduzida a problemática associada com a extração de forma semiautomática de ontologias, *Ontology Learning*. Na secção 3.2 são apresentados dois processos na aprendizagem de ontologias, dando-se um maior destaque ao último processo apresentado. Nas secções que se seguem, são apresentadas algumas técnicas que podem ser utilizadas para concretizar as fases associadas ao processo de aprendizagem. Na secção 3.3 são apresentados alguns sistemas criados tendo por base uma extração semiautomática de uma ontologia. Na última secção do capítulo são discutidas algumas ferramentas criadas com o âmbito de facilitar o processo de extração.

No capítulo 4 é discutida a metodologia implementada para extrair uma ontologia a partir dos dados fornecidos. Na secção 4.1 é discutida a necessidade de implementar um esquema de extração para o domínio de conhecimento em mão, é também apresentado o processo implementado. Na secção 4.2 são discutidas as tecnologias utilizadas para concretizar a ferramenta de extração, e na secção 4.3, são discutidos os vários métodos utilizados em cada uma das fases do processo de extração apresentado anteriormente. De seguida, na secção 4.4 é evidenciada a tecnologia utilizada, de forma a representar os elementos extraídos a partir do processo anterior. O capítulo termina com a secção 4.5, na qual são retratados os dados gerais acerca do processo de extração, onde é também efetuada uma avaliação da ferramenta com base em seis textos.

O capítulo 5 é relativo ao sistema de exploração de dados. Este capítulo começa com uma secção na qual é fundamentada a necessidade de criação deste sistema. Depois, na secção 5.2, é apresentada a estrutura do sistema, as tecnologias escolhidas e a arquitetura do sistema. Na secção 5.3 são apresentados, com maior detalhe, os diversos componentes utilizados no backend do sistema de exploração. Na secção 5.4, por sua vez, é evidenciado o funcionamento geral do frontend. Por fim, na secção 5.4, são expostas algumas imagens relativas ao frontend desenvolvido para o sistema.

Para terminar a dissertação, no capítulo 6 são apresentadas algumas conclusões e comentários acerca do trabalho desenvolvido, bem como alguns aspetos a melhorar e a implementar no futuro.

# 2

# ONTOLOGIAS

## 2.1 DEFINIÇÃO E UTILIZAÇÃO

Nos últimos anos, e com os avanços tecnológicos que se têm verificado, o número de dados disponíveis na Internet aumentou exponencialmente. Como tal, uma quantidade vasta de informação, estruturada ou não estruturada, encontra-se ao dispor para se aplicarem mecanismos de mineração e de exploração. Isto significa que, a problemática da representação do conhecimento presente nessa informação, independentemente da sua estrutura, motiva a utilização de uma estrutura de dados que a permita representar.

As ontologias inserem-se neste contexto. Estas podem ser vistas como a estrutura de um dado conhecimento Chandrasekaran et al. (1999), sendo primordial para a construção do sistema de conhecimento subjacente aos dados envolvidos, permitindo, e por exemplo, a exploração da informação considerada pertinente nesses dados. Sem a representação da informação numa estrutura adequada não é possível fazer a exploração do conhecimento presente nos dados. A representação explícita do conhecimento presente sob a forma de uma ontologia pode, depois, ser facilmente partilhada com outras pessoas, ou certas partes da estrutura da ontologia podem ser reutilizadas para outros propósitos.

Historicamente o termo ontologia já existe há milhares de anos Roche (2003). O trabalho que Aristóteles realizou neste domínio foi dos primeiros a serem desenvolvidos. O termo Ontologia tem origem no Grego, sendo formado pelas palavras *ontos* (ser) juntamente com *logia* (estudos). Assim, pode ser entendido como o "estudo do ser". Filosoficamente, o estudo de ontologias pertence a um dos ramos da metafísica, que trata e procura responder a questões relacionadas com a existência. Mais recentemente, o filósofo Emmanuel Kant também explorou este ramo, definindo Ontologia como um relato sistemático da existência.

No que toca ao ramo computacional, a utilização deste termo começou no início da década de 90, sobretudo através da investigação (e da posterior utilização) de cientistas associados ao ramo da Engenharia do Conhecimento e da Inteligência Artificial. Desde então foram propostas várias abordagens relativamente à definição do termo neste ramo.

De acordo com Uschold Uschold and Gruninger (1996), uma ontologia é conhecimento partilhado acerca de algum domínio de interesse. Uma ontologia permite encapsular uma perspetiva global acerca desse domínio, perspetiva essa que pode envolver um conjunto de conceitos, na forma de entidades e atributos, e de relações num processo que se denomina de conceptualização.

Por sua vez Borst Borst (1997) definiu uma ontologia como sendo uma especificação formal de uma conceptualização partilhada, enfatizando o facto de que tem de haver um acordo relativamente à conceptualização da ontologia.

Por fim, a definição que foi dada por Gruber Gruber (1995), que em termos gerais, é a mais aceite e utilizada na área das Ciências da Computação. Gruber definiu uma ontologia como sendo a especificação explícita de uma conceptualização.

As ontologias revelam-se como sendo vantajosas no que concerne ao foco e uso da sua estrutura para realizar suposições e métodos de inferência sobre os dados, o que poderá possibilitar, por exemplo, a criação de métodos de visualização simples sobre os dados representados na estrutura. No que concerne à inserção de novos elementos numa ontologia, devido à flexibilidade da sua estrutura, uma ontologia poderá facilmente ser alargada, sendo escalável no que toca a este tipo de operações El Kadiri et al. (2015). Por fim, uma ontologia facilmente poderá representar num modelo uma vista geral sobre algum domínio de conhecimento. Depois, esta estrutura facilmente poderá ser partilhada com outros elementos Noy and Mcguinness (2001).

## 2.2 ESTRUTURA DE UMA ONTOLOGIA

A estrutura de uma ontologia varia de acordo com o domínio de conhecimento que se pretende representar e com o uso que terá no futuro. Como tal, os elementos que compõem uma ontologia variam de caso para caso. Contudo, em geral, pode-se destacar alguns dos elementos que poderão compor uma ontologia, nomeadamente Noy and Mcguinness (2001):

1. Classes, ou conceitos, que são os elementos primordiais na construção de uma ontologia. Servem para descrever conceitos num dado domínio. Tal como a classe "Destino" dentro de um domínio de conhecimento associado a viagens.

2. Subclasses, que representam classes mas de uma maneira mais específica. Por exemplo, dentro dos destinos de viagens existem subclasses como a subclasse do "Retiro", pelo que um retiro é um tipo de destino, logo é uma subclasse. As subclasses podem facilmente ser identificadas através de relações "is-a".

3. *Slots* ou atributos, descrevem aspetos ou propriedades das classes. A título de exemplo, e ainda no domínio do conhecimento das viagens, pode ser definida a classe "Contacto", esta classe, depois, poderá conter atributos relacionados, como o código postal, o endereço email, o contacto telefónico, entre outros. Posteriormente, a estes atributos é atribuído um *range*, enquanto que para os atributos anteriores poderá ser um valor do tipo *string*.

4. Relações, representam a ligação entre os diversos componentes da ontologia. Classes como o "Retiro" e a classe "Destino" encontram-se ligadas através de uma relação "is-a".

A base de conhecimento composta através dos elementos referidos anteriormente apenas se poderá considerar completa através da inserção de instâncias de classe. Isto é, para uma dada classe, por exemplo “Retiro”, poderão se inserir várias instâncias como ‘Quinta das Flores’ ou ‘Parque Terra Nostra’.

A estrutura apresentada anteriormente poderá ser abstraída formalmente para um tuplo que contém cinco elementos Guedes et al. (2014):

$$O = <C,R,P,I,A>$$

Neste tuplo, C e R representam, respetivamente, os conceitos/classes e as relações da ontologia. Por sua vez, as relações poderão ser relações taxonómicas, ou seja, relações do tipo "is-a" que contribuem para o estabelecimento da hierarquia das classes, e não taxonómicas. Quanto ao símbolo P, este representa as propriedades da ontologia, pelo que se pode definir propriedades do objeto ou propriedades dos dados Ameen et al. (2012). As propriedades do objeto representam um certo predicado que relaciona um sujeito com um objeto. Enquanto que com as propriedades dos dados é possível associar com o predicado um único sujeito a algum tipo de valor, onde se poderão associar propriedades como *string*, *integer*, entre outras. O símbolo I representa as instâncias da ontologia. Isto é, os objetos concretos do domínio de conhecimento. Por fim, o A representa os axiomas da ontologia, os axiomas definem-se como regras lógicas que no contexto da estrutura da ontologia representam algo que é sempre verdade.

Apresenta-se de seguida uma ontologia desenvolvida no domínio de conhecimento das viagens realizadas por investigadores de *Stanford*.


Figura 1: Ontologia no domínio de conhecimento das Viagens - figura extraída de Knublauch (2016).

De maneira a podermos trabalhar com a ontologia utilizámos o Protégé Musen (2015). Basicamente, esta ferramenta propicia um ambiente munido de um conjunto de ferramentas bastante diverso que podem ser utilizados para fazer a construção de ontologias. Adicionalmente, utilizámos o *plugin OwlViz* para que fosse possível fazer a visualização completa da ontologia.

Como é possível verificar pela Figura 2, a classe "Destino" e a subclasse "Retiro" encontram-se representadas na ontologia, estando ambas ligadas através de um relacionamento “is-a”. Para além disso, pode-se verificar a existência de instâncias associadas, por exemplo, à classe representativa dos parques nacionais, na qual existem dois objetos concretos desta classe. No caso da representação anterior, apenas foram definidas relações do tipo “is-a”. Contudo, também poderiam ter sido definidas outras relações, mais concretas, entre as classes, como uma relação entre atividade e destino, por exemplo. **Qual relação?**

Ainda relativamente à hierarquia de classes/conceitos que foi abordada anteriormente, aquando da explicação formal dos elementos de uma ontologia, o mesmo fenómeno pode ser consultado na Figura 2, na qual o ponto de partida da hierarquia de conceitos é o termo/conceito mais geral até ao ponto de chegada que é o mais específico. Neste caso concreto, os pontos mais específicos são as classes “NationalPark” e “Farmland”.

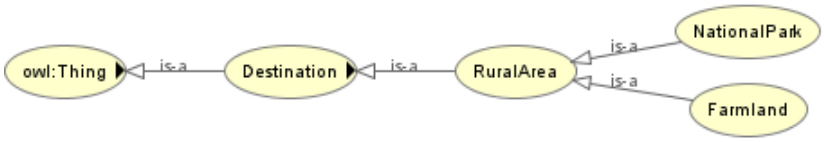


Figura 2: Hierarquia das classes.

Depois de definido este esquema, e com as instâncias corretamente associadas às suas classes, é possível efetuar interrogações sobre a ontologia, de maneira a permitir uma exploração dos dados presentes na sua estrutura. As interrogações poderão variar de interrogações simples, como por exemplo encontrar um dado hotel com certo nome, até interrogações mais complexas, onde, por exemplo, poderão ser feitas interrogações onde se acedem a múltiplas propriedades das diversas classes de forma a responder com o destino mais indicado com base nas propriedades desejadas por parte de um utilizador.

## 2.3 PROCESSO DE DESENVOLVIMENTO

No que concerne ao típico processo de desenvolvimento de ontologias existem diversas propostas por diferentes autores no que toca ao pipeline típico de construção de uma Ontologia. Embora hajam diferentes abordagens no que toca à construção, existem algumas regras comuns na idealização e posterior concretização Noy and Mcguinness (2001):

1. Considerar outras alternativas na idealização do domínio de conhecimento, visto que existem sempre diversas alternativas, mais ou menos complexas, no que concerne ao processo de conceptualização da ontologia;

2. O processo de desenvolvimento é um processo iterativo, conforme a necessidade a ontologia poderá ser alargada ou diminuída o que influencia a complexidade final da ontologia criada;

3. Classes e relações na ontologia deverão pertencer sempre ao domínio de conhecimento em questão. Caso contrário a ontologia criada não será fiel ao domínio que se pretende representar, o que resulta num pior aproveitamento da ontologia para os seus fins.

Relativamente às metodologias de desenvolvimento, quatro dos principais ciclos de desenvolvimento mais mencionados e utilizados na indústria são a metodologia *Tove*, as metodologias propostas por *Uschold* e *Noy* e por fim a *Methontology*.

### 2.3.1 *Metodologia Tove*

Na metodologia Tove, *Toronto Virtual Enterprise*, são propostos os seguintes passos de maneira a formalizar uma ontologia Jones et al. (1998) Kim et al. (1995) Cristani and Cuel (2005):

1. Cenário motivacional, conjunto de problemas ou questões que motivam a criação da ontologia. Ou seja, é o conjunto de requisitos necessários para a implementação da ontologia. Desta maneira, ao formalizar o conjunto de questões necessárias a responder, é então possível de uma maneira intuitiva encontrar soluções para resolver o problema;

2. Questões informais de competência, baseado no cenário motivacional são definidos os conjuntos de requisitos através de questões informais que a ontologia terá de responder;

3. Especificação da terminologia, os vários componentes da ontologia, mais concretamente as classes, as relações e os atributos são especificados formalmente e tipicamente em lógica de primeira ordem;

4. Questões formais de competência, os requisitos da ontologia levantados anteriormente são formalmente concretizados de acordo com a terminologia formalizada anteriormente.

5. Especificação dos axiomas, o conjunto de axiomas que restringe a ontologia são especificados em lógica de primeira ordem tal como foi feito para os restantes elementos da ontologia;

6. Teoremas de completude, corresponde à fase de avaliação da ontologia criada onde é verificado se a ontologia cumpre com os requisitos. Para tal, a avaliação é feita através da criação de condições onde as soluções para as questões de competência são completas.

A imagem apresentada de seguida, Figura 3, esquematiza e resume os princípios por detrás desta metodologia de desenvolvimento:


Figura 3: Esquema da metodologia Tove.

Como é possível verificar, esta metodologia é muito assente em questões formais e de lógica de primeira ordem pelo que será possível então definir uma ontologia final mais robusta e assente em lógica.

### 2.3.2 *Metodologia Uschold*

De seguida serão apresentadas três metodologias semelhantes entre si, não obstante possuem algumas diferenças.

Primeiramente, e tal como proposto por Uschold Uschold and Gruninger (1996), este ciclo de desenvolvimento foi das primeiras metodologias propostas nesta área. Trata-se portanto de um marco no que toca às boas práticas no desenvolvimento de ontologias. De seguida apresentam-se então o conjunto de passos necessários para o desenvolvimento de uma ontologia:

1. Identificação do objetivo e o domínio da ontologia, de maneira a guiar os posteriores passos;
2. Própria construção da ontologia que envolve três fases;
   2.1. Fase de captura, onde se identificam os termos chave e as relações entre os termos dentro do domínio de conhecimento;
   2.2. Fase de *coding*, representação explícita partindo da conceptualização idealizada anteriormente, os termos chave e relações entre os termos são traduzidos em termos dos elementos da ontologia. São identificadas portanto as classes, relações, atributos e indivíduos da ontologia;
   2.3. Fase de integração de ontologias, onde é relevante considerar a integração de ontologias ou parte de ontologias que já existem no mesmo âmbito do problema em mão, de maneira a completar/facilitar a construção da ontologia;
3. Avaliação e verificação, através de alguns métodos, que serão apresentados posteriormente. A ontologia é verificada e avaliada de maneira a averiguar se esta se encontra correta face o objetivo e o domínio proposto;
4. Documentação, onde são criados alguns comentários acerca da ontologia de maneira a depois facilitar o uso e a partilha da ontologia com outras pessoas.

O esquema apresentado na Figura 4 retrata o funcionamento geral deste ciclo de desenvolvimento, que como se poderá verificar, é um ciclo bastante iterativo e que coloca bastante ênfase na partilha/integração de ontologias já existentes no processo de desenvolvimento

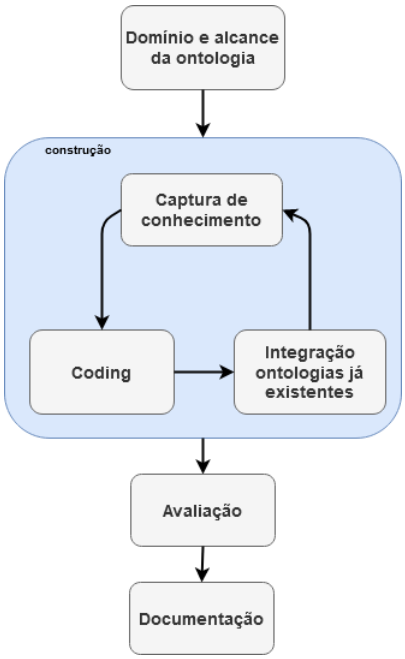


Figura 4: Esquema da metodologia proposta por Uschold.

### 2.3.3 *Metodologia Noy*

Por sua vez Noy Noy and Mcguinness (2001), ao esquematizar o seu próprio ciclo de desenvolvimento, baseou-se no ciclo apresentado anteriormente. Em geral as fases apresentadas previamente são respeitadas. Antes do desenvolvimento, existem alguns problemas e decisões que têm de ser tratados previamente.

Primeiramente, é necessário decidir em antemão o papel que a ontologia irá representar, qual o seu fundamento e a utilização que esta terá no futuro. Para além disso, o detalhe ou a superficialidade que a ontologia terá no que toca à captura do domínio de conhecimento também terá que ser decidido em antemão, uma vez que quanto mais abrangente a ontologia mais complexo será o processo de desenvolvimento. Esta decisão é fundamental, e poderá guiar e potencialmente resolver problemas que poderão acontecer no processo de desenvolvimento. No que toca às várias alternativas na conceptualização da ontologia, também deverá se procurar sempre optar pela solução que seja mais intuitiva.

Após a resolução de todas estas questões, o desenvolvimento da ontologia é expectável que seja mais simples, devido a toda a fundamentação que foi feita até este ponto. Como tal, o ciclo de desenvolvimento irá envolver as seguintes etapas:

1. Determinar o domínio e o alcance da ontologia, onde são colocadas as questões abordadas anteriormente, questões como o papel e a utilização que esta terá no futuro;

2. Procurar soluções já existentes, onde se deve considerar integrar ontologias já existentes neste âmbito, e depois refinar e adaptar consoante o necessário;

3. Criar um dicionário de termos dentro do domínio de conhecimento, onde devem ser enumerados termos e caraterísticas relevantes de acordo com o domínio que se pretende captar;

4. Definir as classes e a hierarquia de classes da ontologia, a hierarquia poderá seguir uma abordagem *top-down*, *bottom-up* ou uma mistura das duas. Uma hierarquia *top-down* segue uma abordagem onde o ponto de partida é a classe mais geral até à classe mais específica. A abordagem *bottom-up* por sua vez é o inverso da abordagem anterior;

5. Definir as propriedades para cada uma das classes delineadas anteriormente. As propriedades definidas são por sua vez guiadas pelo dicionário de termos;

6. Definir o tipo de dados de cada uma das propriedades;

7. Concretização da conceptualização anterior, onde a ontologia é criada de acordo com o resultado das fases anteriores.

Mais uma vez, foi elaborado um esquema, que poderá consultado na Figura 5, que sumariza e apresenta as etapas principais que compõem o ciclo de desenvolvimento proposto por Noy.

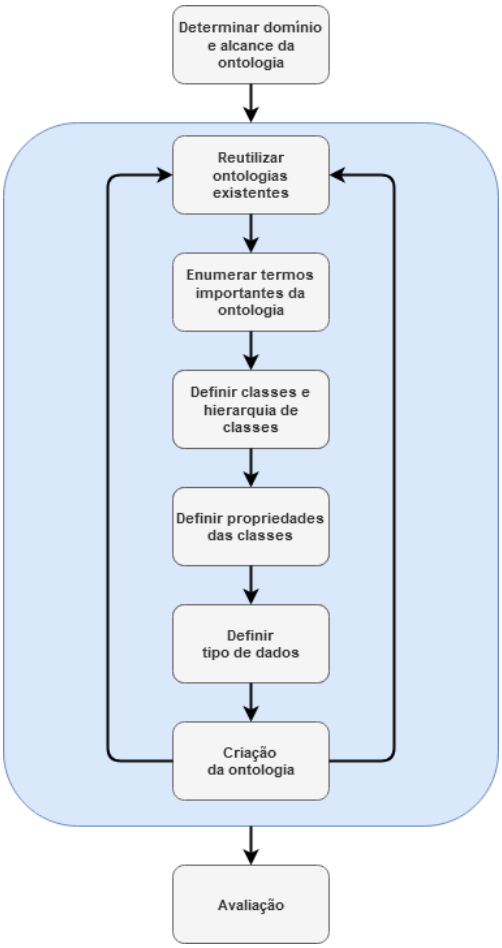


Figura 5: Esquema da metodologia proposta por Noy.

### 2.3.4 *Metodologia Methontology*

Por fim, uma das metodologias mais relevantes e mais aplicadas na indústria no processo de desenvolvimento é a *Methontology*. O ciclo de desenvolvimento desta metodologia pode ser abstraído e resumido em cinco fases cruciais.

Na primeira fase Janowicz et al. (2008), a fase de especificação, é necessário uma vez mais definir o uso e o domínio que a ontologia criada deverá albergar, pelo que deverá ser determinada o quão abrangente ou não a ontologia final deverá ser perante o conhecimento que se pretende representar. Na segunda fase, a etapa de conceptualização, o conhecimento que se pretende captar e representar em função da fase anterior terá que ser formalizado sob a forma de um modelo conceptual. No modelo conceptual são identificados os elementos da ontologia a representar. O uso de um glossário ou dicionário de termos Fernandez et al. (1997) poderá ajudar nesta fase. De seguida, na fase de formalização, o modelo conceptual criado anteriormente terá que ser concretizado na forma dos elementos bases da ontologia, ou seja são definidas as classes, relações e

atributos da ontologia. As duas últimas fases correspondem à implementação e manutenção. Na implementação a ontologia é criada através do uso, por exemplo, de alguma ferramenta de criação de ontologias. Na fase da manutenção, periodicamente será necessário atualizar e corrigir alguns erros que possam existir na ontologia.

Para além disso existem também algumas atividades complementares a este ciclo de desenvolvimento como a documentação e avaliação da ontologia bem como a tarefa de aquisição de conhecimento e a tarefa de integração de ontologias já elaboradas Pérez et al. (2004). Estas atividades complementares são executadas em paralelo com todas as fases de construção apresentadas anteriormente, contudo é feita a distinção que, por exemplo, enquanto que a documentação é um processo constante e linear no decorrer da construção da ontologia, isto é terá de ser efetuado em todas as fases, por sua vez a aquisição de conhecimento é mais comum e ocorre com maior volume nas fases mais iniciais do processo de desenvolvimento, ou seja nas fases de especificação e conceptualização.

Na Figura 6 é possível observar as etapas principais no desenvolvimento de uma ontologia segundo esta metodologia, bem como as atividades complementares esquematizadas na forma de um esquema. Nas etapas complementares é possível verificar através do relevo da curva as etapas principais em que ocorre uma maior incidência dessa atividade complementar.

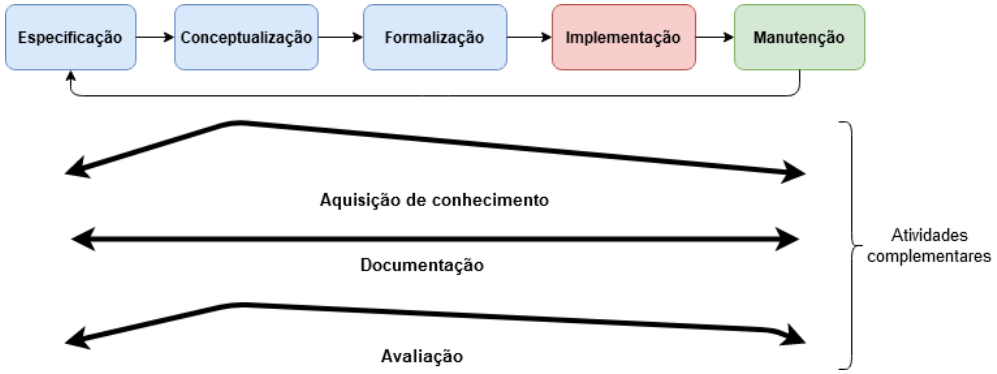


Figura 6: Esquema da metodologia Methontology.

Em suma, quando comparados todos os ciclos de desenvolvimento apresentados, e como se poderá constatar, os ciclos propostos por Uschold, Noy e o *Methontology* são bastante semelhantes, pelo que a metodologia proposta por Noy destaca-se mais pela baixa granularidade. A metodologia *Tove* por sua vez, devido ao seu foco no uso de lógica de primeira ordem, é a mais distinta quando comparada com as outras. Em todas as metodologias apresentadas existe também um foco no desenvolvimento iterativo da ontologia.

## 2.4 AVALIAÇÃO DE ONTOLOGIAS

Com o esquema da ontologia definido, é necessário agora avaliar a ontologia criada até este momento, de maneira a melhor entender se esta cumpre com os requisitos e se representa de uma maneira adequada o

domínio de conhecimento que pretende representar. A avaliação de uma ontologia é composta tanto pela verificação como pela validação Reyes-Peña and Tovar-Vidal (2019).

Na verificação é necessário determinar se a ontologia encontra-se corretamente modelada. Para tal, na verificação são analisadas questões lexicais onde se poderá consultar o dicionário de termos criado, caso este tenha sido criado, e verificar se o conhecimento representado na ontologia corresponde aos elementos presentes no dicionário. Poderão também ser analisadas questões semânticas referentes à consistência no que toca à captura de termos do domínio, e questões estruturais relativas à própria estrutura da ontologia, onde se verifica que não existem, por exemplo, *loops* na constituição da ontologia.

No que toca à validação, uma das técnicas mais utilizadas para validar a ontologia é a técnica *Data Driven*. Nesta técnica são utilizados dados textuais do mesmo domínio de conhecimento da ontologia criada, de maneira a que se compare esses dados textuais com os elementos da ontologia. Para tal, podem ser aplicados mecanismos de extração de termos Brewster et al. (2004) dos dados textuais, sendo que de seguida esses elementos retirados automaticamente são comparados com os termos presentes na ontologia criada. A ontologia depois pode ser penalizada por cada termo que tem em falta relativamente aos dados textuais ou por termos que estão presentes na ontologia mas que não estão presentes nos dados textuais. A pontuação no final deste processo irá ditar o quão adequada a ontologia se encontra face ao domínio de conhecimento que pretende representar.

Outra técnica de validação também utilizada é a *Application Driven* Brank et al. (2005). Tipicamente as ontologias são utilizadas em conjunção com alguma aplicação, de tal forma que ao avaliar o desempenho da aplicação, isto é a qualidade das respostas dadas por parte da aplicação, é então possível inferir e determinar o quão adequada a ontologia se encontra. Contudo, uma das principais desvantagens em utilizar esta técnica poderá ser o próprio papel da ontologia visto que esta poderá desempenhar um papel pequeno na aplicação, logo os resultados da aplicação poderão não ser representativos da qualidade da ontologia criada.

## 2.5 APLICAÇÕES E SISTEMAS ONTOLÓGICOS

São muitas as possíveis áreas e problemáticas que as ontologias poderão resolver. Para além destas conseguirem representar o conhecimento embebido em, por exemplo, dados não estruturados. Segundo Uschold Uschold and Gruninger (1996) o uso e aplicações de ontologias podem-se dividir em três categorias distintas.

Na primeira categoria, e tendo em conta no que concerne às vantagens inerentes no uso de ontologias, no que toca à criação de uma estrutura de dados unificadas e com uma semântica comum, ao utilizar uma ontologia é possível resolver problemas de comunicação. Ou seja, com o uso de uma ontologia é então possível utilizar esta como uma plataforma que permita a troca e partilha de conhecimento entre diversas pessoas ou máquinas dentro de uma organização. Assim, é possível que uma equipa de trabalho tenha acesso a uma estrutura com elementos claros e concisos através da definição à partida de uma terminologia base para todos os objetos e relações.

Na segunda categoria, a de interoperabilidade, poderão existir situações em que diferentes utilizadores necessitem de partilhar dados ou que utilizam ferramentas diferentes. Ora, devido às caraterísticas mencionadas anteriormente, uma ontologia poderá servir como uma solução de integração de diferentes ferramentas no mesmo

ambiente de utilização. Ou seja, através do uso de uma ontologia é possível integrar e descrever diferentes tipos de informação provenientes de diferentes fontes numa estrutura única.

A última categoria das aplicações das ontologias é relativa ao uso destas na criação de aplicações/sistemas de conhecimento. Em título exemplificativo, uma ontologia poderá ser usada e exportada para um sistema de base de dados, a base de dados consequentemente irá partilhar a estrutura da ontologia, pelo que depois podem ser feitas interrogações à base de dados através de operações *CRUD* (*create*, *read*, *update*, *delete*) de maneira a suportar operações efetuadas sobre a aplicação/sistema.

Fruto das diversas possíveis aplicações de ontologias são vários os sistemas idealizados e criados com o uso desta estrutura de dados.

Um dos vários sistemas que utilizam soluções ontológicas é o *LinkedDesign* El Kadiri et al. (2015). O objetivo da criação deste sistema passou pela conceção de uma plataforma centralizada onde diversos fabricantes poderiam partilhar conhecimento e informação acerca de todas as áreas do ciclo de vida do seu produto. Ora, este conhecimento que é partilhado poderá ser acerca de vários aspetos relativamente ao seu produto, pelo que existem portanto problemas relacionados com a interoperabilidade. As ontologias por sua vez permitem representar o conhecimento através de um modelo comum, centralizado e com uma semântica bem definida em relação a todos os participantes. O que irá servir como base para a troca, acesso e partilha de informação entre as diversas pessoas e sistemas independente do *background*.

Outro sistema que utiliza também uma ontologia para permitir a modelação e a consequente navegação do conhecimento presente na estrutura é o projeto *NOPIK* (*Personal Information and Knowledge Organizer Network*) Öhgren (2004). O objetivo neste sistema é a criação de uma plataforma distribuída que auxilie na tarefa de gestão e armazenamento de conhecimento acerca da informação pessoal de diferentes fontes. Tal como no caso anterior, a ontologia foi utilizada de maneira a que seja possível a criação de uma semântica e estrutura comum a todos os participantes, desta maneira são resolvidos problemas relacionados com a interoperabilidade para que depois seja possível a partilha e consequente exploração do conhecimento.

## 2.6 FERRAMENTAS DE DESENVOLVIMENTO

Existem várias ferramentas e ambientes de criação de ontologias, das principais ferramentas utilizadas poderá se destacar o *Protégé* Musen (2015). Esta ferramenta disponibiliza uma interface gráfica e configurável, para além de um elevado foco na *API* e em questões de modularização. O *Protégé* apresenta também uma arquitetura bastante flexível no que toca à adição de *plugins* feitos por terceiros, *plugins* esses que podem oferecer novos e inovadores mecanismos à ferramenta.

Outra ferramenta para auxiliar na tarefa de construção de ontologias é o *NeOn Toolkit* NeOn Technologies Foundation (2014). Esta ferramenta também disponibiliza uma interface gráfica, de maneira semelhante ao *Protégé*. Para além disso, também são disponibilizados mecanismos que abrangem todo o ciclo de desenvolvimento de uma ontologia. Contudo, ao contrário do *Protégé*, o *NeOn Toolkit*, embora também ofereça a possibilidade de adição de *plugins*, fruto da menor popularidade desta ferramenta não existem tantos *plugins* disponíveis.

O *OWLGrEd* também é um editor gráfico de ontologias Liepins et al. (2014), pelo que este utiliza uma linguagem de modelação baseada em *UML* (*Unified Modeling Language*). O foco desta ferramenta não é abranger o ciclo de vida de desenvolvimento de uma ontologia, embora isto também seja possível neste ambiente, mas sim a visualização e *debug* de ontologias ao utilizar um ambiente *UML* comum a todos os desenvolvedores de *software*. Para tal, são traduzidos para *UML* os elementos da ontologia, pelo que o resultado final é uma ontologia com um aspeto semelhante a um diagrama de classes. Desta maneira, é possível mais facilmente introduzir esta área das ontologias a uma público mais abrangente.

# 3

# EXTRAÇÃO SEMIAUTOMÁTICA DE ONTOLOGIAS

## 3.1 A PROBLEMÁTICA DA APRENDIZAGEM DE ONTOLOGIAS

O processo de construção manual de ontologias, tal como foi abordado, é um processo que incorre em vários custos. Esses custos variam desde temporais, devido ao tempo que é gasto na modelação e captação de novo conhecimento para representar na ontologia, até aos custos associados à necessidade de haver um especialista que consiga mapear e esboçar corretamente um certo domínio de conhecimento para uma ontologia. Ora, se para certos casos os custos não impedem a construção manual de uma ontologia, ou seja, o domínio de conhecimento que se pretende representar é pequeno, para outros a representação de forma manual poderá até mesmo ser impossível. Isto significa que, para casos nos quais o domínio de conhecimento é largo e vasto poderá se incorrer num *bottleneck* Maedche and Staab (2001) no que toca à representação desse conhecimento numa ontologia de forma manual.

Como tal, surgiu então a necessidade de automatizar o máximo possível o processo de geração de ontologias, fruto da situação retratada anteriormente. É neste contexto que se insere a área de *Ontology Learning* Maedche and Staab (2001).

O objetivo da *Ontology Learning* passa então pela integração de diferentes áreas de *Machine Learning*, mais propriamente da área de *Natural Language Processing*, de maneira a facultar o uso de técnicas para identificar, por exemplo, termos/conceitos, propriedades, e relações que se encontram presentes nos dados Wong et al. (2012). Depois, todos estes elementos serão utilizados para compor uma ontologia para albergar o domínio de conhecimento pretendido. Porém, o processo de extração não se poderá considerar totalmente automático, uma vez que requer a atenção do utilizador para validar/avaliar e ajustar os parâmetros de acordo com os resultados.

No que concerne ao grau de dificuldade da extração, esta considera-se tão complicada quanto a granularidade final que se pretende na ontologia Wong et al. (2012). Ontologias *lightweight* representam ontologias que pouco ou nada integram axiomas na sua estrutura. Exemplos destas ontologias são os glossários ou os dicionários de dados. No outro lado do espectro encontram-se as ontologias *heavyweight*, que por sua vez utilizam axiomas na sua estrutura. Portanto, a extração de ontologias *lightweight* é feita com uma menor complexidade quando comparada com uma extração de ontologias *heavyweight* devido à dificuldade em extrair regras lógicas, os axiomas, a partir do texto.

No entanto, se por um lado a extração semiautomática de ontologias permite que se elimine a maior parte dos custos associados à construção manual de ontologias, por outro lado também existem certas desvantagens associadas a este processo.

Primeiramente, existem questões e problemas relacionados com a escalabilidade deste processo, uma vez que com uma incorporação em larga escala de dados num processo de extração, poderão surgir problemas relacionados com a fraca escalabilidade dos algoritmos de *machine learning* utilizados para a extração de elementos da ontologia. Uma vez que os algoritmos que são aplicados para permitir a extração dos elementos por norma são computacionalmente exigentes. Em situações nas quais existe uma quantidade grande de dados poderá ser até impeditivo a utilização de certos tipos de algoritmos, uma vez que existiria um custo elevado em termos computacionais. Como tal, têm de ser desenvolvidas estratégias específicas de maneira a poder distribuir melhor os recursos computacionais, ou até mesmo, permitir a reutilização de conhecimento passado que já foi processado, para que não seja necessário processar esse conhecimento novamente caso apareça no futuro.

Os dados que são utilizados no processo de extração da ontologia também têm que ser verificados, uma vez que, tipicamente, existe bastante incerteza em relação aos dados que são utilizados, especialmente em casos nos quais estes são retirados da Internet a partir de fontes heterogéneas Somodevilla et al. (2018). Ou seja, em certos casos os dados que estão a ser utilizados para extrair uma ontologia são de fraca qualidade, não apresentando grandes interligações entre conceitos, por exemplo. Isto, originará uma ontologia de fraca qualidade por consequência. Portanto, antes de qualquer tipo de algoritmo é necessário proceder a uma análise manual do texto, de maneira a anotar e resolver possíveis inconsistências que existam no texto.

Por fim, o grau de intervenção por parte do utilizador também é um problema que terá que ser decidido de antemão. Em teoria, o processo é tão mais desejável quanto menor for a intervenção humana. Contudo, na prática a intervenção humana é desejável uma vez que, e tal como foi abordado anteriormente, para que seja possível garantir a qualidade de uma ontologia é necessário que o utilizador proceda a uma avaliação dos resultados de forma a poder a ajustar os parâmetros necessários para garantir melhores resultados numa próxima iteração.

## 3.2 PROCESSO TÍPICO DE EXTRAÇÃO SEMIAUTOMÁTICO

Relativamente ao processo típico de extração semiautomática no que toca à metodologia a seguir para fazer a extração da ontologia, mais uma vez, e como aconteceu no caso da construção manual de ontologias, não existe nenhuma convenção que suporte a sua realização.

Uma das primeiras metodologias propostas nesta área foi feita por Buitelaar Brewster (2006), que organizou a metodologia através de um *layer cake* organizado por ordem crescente da dificuldade das subtarefas de extração semiautomática dos elementos da ontologia do texto. As subtarefas presentes neste esquema, que se poderá verificar na Figura 7, foram inspiradas também no processo de construção manual de ontologias.

Para melhor exemplificar o *output* pretendido para cada fase, apresentado na Figura 7, utiliza-se mais uma vez o domínio de conhecimento das viagens. Para tal, analisando o processo da tarefa mais simples para a mais complicada, primeiramente foram enumerados alguns termos relacionados com o domínio das viagens e, de

seguida, dentro dos termos foram identificados os termos sinónimos, o que poderá indicar uma relação entre os termos. Na fase seguinte, são construídos os conceitos, fruto da identificação dos termos e sinónimos feita anteriormente. No esquema apresentado também é possível observar um exemplo relacionado com a hierarquia de classes "is-a", juntamente com um exemplo de uma relação entre classes. Por fim, e o que se pode considerar como sendo a tarefa mais complexa num processo de extração semiautomático, apresenta-se um exemplo de um axioma, representando uma regra lógica no domínio de conhecimento albergado na ontologia.

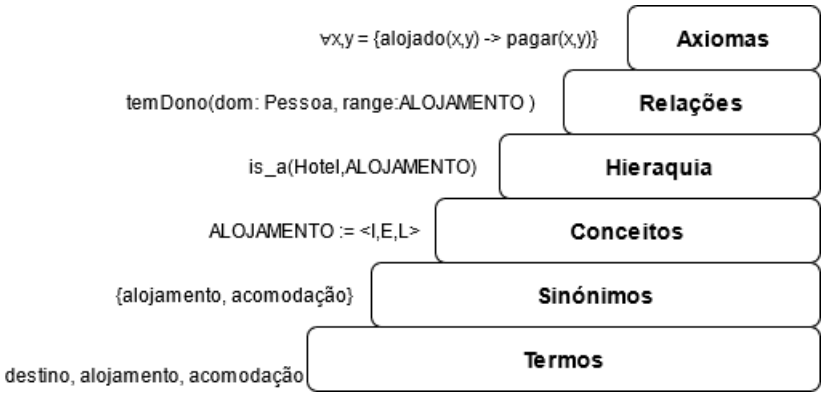


Figura 7: *Ontology Learning Layer Cake*.

Contudo o esquema anteriormente poderá ser simplificado Tiwari and Jain (2014), pelo que existem sub tarefas que poderão ser executadas na mesma fase. A sub tarefa dos termos e sinónimos poderá ser abstraída para a mesma tarefa, uma vez que ambas são semelhantes em termos linguísticos e, também, em termos da aplicação de métodos para permitir esta extração. As sub tarefas das relações e da hierarquia também foram transformadas para apenas uma tarefa, que implica a extração de relações do tipo "is-a", que está relacionada com a hierarquia, bem como outras relações mais complexas, como, por exemplo, a "temDono" que está presente no exemplo anterior. Além disso, foi possível converter ambas as sub tarefas anteriores para uma só, porque ambas representam relações.

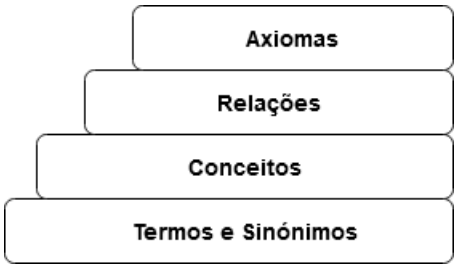


Figura 8: *Ontology Learning Layer Cake* proposto por Tiwari.

Em termos mais concretos, Asim Asim et al. (2018) propôs a sua própria metodologia, como se poderá verificar na Figura 9. Esta metodologia é composta por seis fases distintas. Onde, nesta metodologia estão incorporadas

diversas fases, desde a análise e pré processamento do texto até à última fase onde é elaborada a avaliação da ontologia extraída de forma concretizar a sua extração.

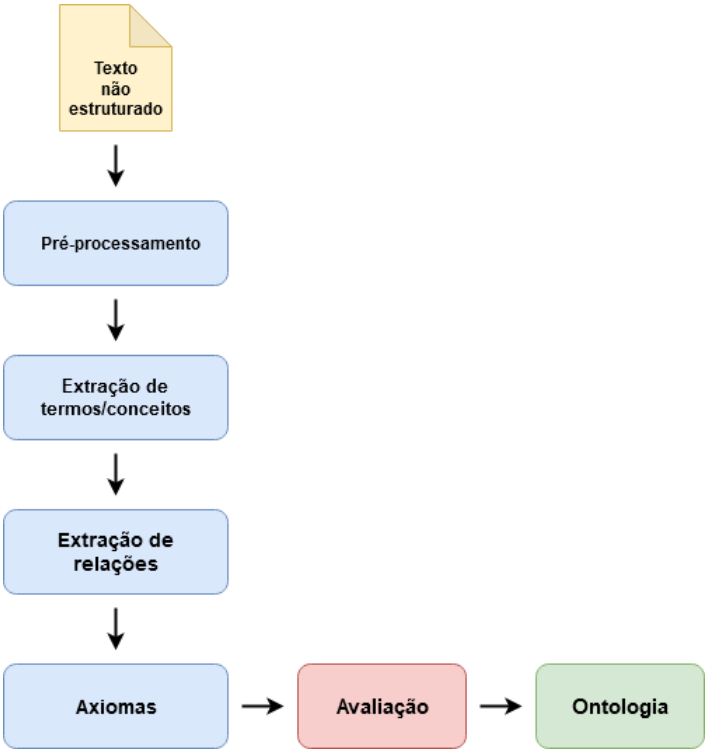


Figura 9: Metodologia proposta por Asim.

Na primeira fase El Ghosh et al. (2017), o pré processamento do texto dado como *input* para a construção da ontologia, é realizado um conjunto de ações com o intuito de limpar e preparar o corpo do texto para a posterior aplicação de algoritmos. Como tal, nesta fase são removidas possíveis ambiguidades bem como potenciais inconsistências estruturais ou semânticas que, depois, podem comprometer a ontologia final, o que, por consequência, irá resultar num texto mais simples e coeso.

Na fase seguinte, através da aplicação de mecanismos linguísticos ou estatísticos é possível extrair termos/-conceitos para a utilização posterior como elementos de uma ontologia. O *output* desta fase é um *cluster* de termos/conceitos, sem qualquer relação e ligação entre si. Posteriormente, será necessário relacionar todos estes elementos, sendo necessário, então, realizar a fase da extração de relações. Relativamente às relações, é importante distinguir o tipo de relações que vão ser extraídas Wong et al. (2012). Existem as relações taxonómicas, que são relacionadas com a hierarquia de classes, ou seja as relações "is-a", e relações não taxonómicas que correspondem a relações entre conceitos da ontologia, como, por exemplo, a relação "temDono" da Figura 7. Mais uma vez, nesta fase, podem ser empregues algoritmos baseados, principalmente, em questões linguísticas ou estatísticas, que são utilizados para analisar as dependências entre termos/conceitos no texto.

Na última fase, que é a fase mais complexa neste processo todo, podem ser empregues técnicas de programação lógica para permitir a extração de axiomas.

Por fim, e tal como acontece em qualquer metodologia de construção manual de ontologias, é necessário verificar e validar a ontologia construída, com o objetivo de averiguar se esta se encontra correta face ao domínio de conhecimento que se pretende representar. Para tal podem ser aplicadas algumas das técnicas e métodos de verificação/validação já discutidas anteriormente na secção 2.4.

### 3.2.1 *Pré Processamento*

A fase de pré processamento possui a finalidade de gerar um texto mais coeso, simples e pronto, para se aplicarem de seguida os algoritmos de extração. Como tal, nesta fase, os tipos de métodos utilizados variam consoante os dados dados como *input*.

O pré processamento poderá ser iniciado com a eliminação de inconsistências estruturais. Isto é, eliminar espaços e mudanças de linhas múltiplas. De seguida poderão ser eliminadas as *stop words* contidas no texto Vijayarani et al. (2015). As *stop words* indicam um conjunto de palavras comuns na língua, que aparecem em larga quantidade, que não oferecem qualquer tipo de significado extra ao texto, ou seja, são palavras irrelevantes no contexto geral do texto. O conjunto de *stop words* tipicamente é composto por conjunções, determinantes e preposições. Ao eliminar estas palavras mais irrelevantes pretende-se obter um texto final que possua, na sua maioria, apenas os termos mais relevantes para o domínio que o texto representa. De maneira a filtrar ainda mais o texto, pode-se aplicar também um filtro de tamanho mínimo de palavras, onde, por exemplo, poderão ser eliminadas palavras do texto que contêm um tamanho mínimo inferior a dois caracteres.

Após simplificar o texto, geralmente opta-se por de seguida separar as frases e as palavras com algum tipo de separador. De forma a separar as frases utiliza-se o *sentence splitting*, e de maneira a separar as palavras utiliza-se o *tokenization* Cimiano et al. (2009). Ambos os separadores podem variar, de acordo com o texto a processar, mas geralmente utilizam-se sinais de pontuação e espaços para obter no final, para cada frase, uma lista com as palavras em cada índice. Desta maneira é possível obter uma estrutura que poderá ser mais facilmente interpretada e percorrida pelos sucessivos algoritmos de extração dos elementos para compor a ontologia.

De maneira a preparar o resultado anterior para os algoritmos de extração, podem ser aplicadas algumas técnicas mais avançadas tal como a *lemmatization*. O objetivo desta técnica é reduzir a complexidade das palavras, ou seja, reduzir sempre que possível as palavras à sua forma mais básica Tiwari and Jain (2014). Com isso, pretende-se, homogeneizar a complexidade textual. A título exemplificativo, ao aplicar esta técnica, palavras como “conduzia” e “conduzo” seriam substituídos por “conduzir”.

Nesta fase também será possível realizar o processo de anotação do texto, podendo para isso serem utilizados algoritmos como, por exemplo, o *Part of Speech Tagging*. O objetivo deste processo será associar a cada *token*, proveniente da *tokenization*, a sua respetiva classe gramatical Kumawat and Jain (2015). De tal maneira que, para um exemplo como a “Conduzo carro grande”, ao utilizar o *Part of Speech Tagging* será possível associar a cada palavra os seguintes *tokens*: Conduzo(*verb*) / carro(*noun*) / grande(*adjective*).

### 3.2.2 *Extração de Conceitos e Classes*

No que diz respeito à segunda etapa do processo, tal como aconteceu anteriormente, existe uma variedade de técnicas muito diversificadas, tanto linguísticas como estatísticas, que podem ser aplicadas de maneira a permitir a extração de conceitos e classes do texto. Neste processo podem-se destacar as técnicas de *seed words*, *C/NC-Value*, algoritmos de *clustering* e, por fim, através do uso de padrões léxico-sintáticos.

#### *Seed Words*

O primeiro algoritmo de extração de conceitos e classes que vamos analisar denomina-se de *seed words* Fraga et al. (2017). A utilização deste algoritmo requer a atenção e interação de algum especialista, de maneira a construir um dicionário de termos relevantes e importantes em relação ao domínio de conhecimento. De seguida, com base no dicionário que é utilizado como *seed*, são filtrados e retirados todos os termos presentes no dicionário construído. Esta extração poderá até mesmo ser melhorada com o cálculo de medidas de similaridade entre termos do texto e os termos contidos no dicionário, para permitir uma maior diversificação no processo de extração. Contudo, esta técnica é custosa, uma vez que requer a análise e levantamento de termos dentro do domínio de conhecimento, por parte de um especialista, de maneira a construir o dicionário.

#### *C/NC-Value*

A segunda técnica, que combina uma abordagem linguística e estatística, é a *C/NC-Value* Frantzi et al. (1998). Esta técnica, ao contrário do caso anterior, não requer intervenção humana. O principal objetivo passa pela identificação de termos complexos, ou seja termos constituídos por mais do que uma palavra que podem compor uma classe da ontologia.

Nesta técnica utiliza-se um conjunto de termos candidatos como input, sendo que no final dada uma pontuação, que combina tanto o *C-Value* como o *NC-Value*. O conjunto de termos candidatos poderá ter sido obtido através da aplicação, por exemplo, de uma técnica como o *Named Entity Recognition*, que identifica o tipo do termo que foi processado até aquele momento, como por exemplo, pessoa ou data, entre outros. E, por fim, um *Chunker* para marcar pedaços de texto de acordo com a informação semântica desse pedaço. Por exemplo, “NP – Noun Phrase”, que poderá ser associado a uma frase como “Agentes de polícia subcontratados”. Os termos candidatos são depois obtidos ao processar a lista gerada pelo *chunker* de acordo com alguma medida.

Após a obtenção da lista de termos candidatos, com a aplicação do *C-Value* é calculado o *termhood* dos termos multi palavra e por fim é produzida uma lista ordenada por *termhood*. A título exemplificativo, para o termo anterior “agentes de polícia subcontratados” este termo poderá representar um conceito na ontologia. Porém, este termo também poderá ser composto por termos mais pequenos como “agentes de polícia” e “polícia subcontratados”, pelo que o *C-Value* também terá de ser calculado para cada um dos sub termos. O *C-Value* é calculado através da combinação de vários métodos, como o cálculo da frequência do termo multi palavra nos dados textuais e também o cálculo do número de palavras do termo, uma vez que é expectável termos com um menor número de palavras apareçam mais frequentemente que termos com maior tamanho. Assim, é importante distinguir a frequência do termo consoante o seu tamanho. Até este ponto do cálculo do *C-Value* ainda não foi

incorporada informação contextual, informação essa que é utilizada para o cálculo do *NC-Value*. No *NC-Value* é utilizada uma *sliding window* Asim et al. (2018) em relação ao termo candidato, de uma palavra para a esquerda e para a direita, sendo depois calculado um peso que é relacionado com a frequência de ocorrência daquelas palavras de contexto juntamente com o termo candidato. Por fim o *NC-Value* é calculado tendo em conta este peso juntamente com o *C-Value* para esse termo.

### *Clustering*

No processo de extração de conceitos e classes também poderão ser utilizados algoritmos de *clustering* Arora et al. (2017), como, por exemplo, o *k-means clustering* para extrair *clusters* de termos semelhantes entre si, estes termos poderão representar depois classes de uma ontologia. Para tal, primeiro é necessário anotar o texto, procedendo à aplicação de técnicas como as já discutidas *Tokenization* e o *Part of Speech Tagging* juntamente com o *Chunker*. De seguida, a partir deste texto já anotado, é possível filtrar todos os *chunks* exceto aqueles aos quais estão associados alguma *tag*, como, por exemplo, filtrar todos os *chunks* exceto os que contêm a identificação de *noun phrase*. Após a obtenção destes termos é possível de seguida aplicar, caso ainda não tenham sido aplicadas, técnicas adicionais de pré processamento, como a *Lemmatization* de maneira a reduzir os termos à sua forma mais básica.

De seguida, através do cálculo de medidas de similaridade, tanto semânticas ou sintáticas ou até mesmo uma combinação das duas, é possível obter uma matriz de similaridade. Esta matriz é de seguida usada com um algoritmo de *clustering*, como o *k-means*, de maneira a produzir *clusters* de termos aglomerados, de acordo com a similaridade entre si. Estes termos aglomerados e separados por *clusters* de seguida podem ser usados para identificar classes e relações de uma ontologia. Na Figura 10 é possível observar um esquema relativo a uma sequência de ações que pode ser realizada após o texto já ter sido pré processado e anotado.


Figura 10: Exemplo de uma sequência de instruções quando se utiliza algoritmos de *clustering*.

### *Padrões Léxico-Sintáticos*

Outro técnica que pode ser utilizada na extração de conceitos, e até mesmo no posterior estabelecimento de relações entre estes, é a aplicação de padrões linguísticos. Esta técnica é particularmente interessante no que toca à extração de pares hiperónimo/hipónimo, que depois poderão ser utilizados para estabelecer uma hierarquia de conceitos através de relações "is-a". Hearst Hearst (1992) propôs os padrões léxico-sintáticos que apresentamos na Tabela 1 para fazer a extração de relações de hiperónimo e hipónimo a partir de um texto.

Tabela 1: Padrões léxico-sintáticos propostos por Hearst Hearst (1992).

| Regra | Padrão |
|---|---|
| i | NP such as {NP,}* {(or\|and)} {NP} |
| ii | such NP as {NP,}* {(or\|and)} {NP} |
| iii | NP {,NP}*{,} or other NP |
| iv | NP {,NP}* {,} and other NP |
| v | NP {,} including {NP,}* {or\|and} NP |
| vi | NP {,} especially {NP,}* {or\|and} NP |

Desta maneira será possível no texto identificar conceitos, representados na tabela por NP, e as suas relações, satisfazendo as duas fases, de extração e de relação dos conceitos. A título exemplificativo, em frases como "(...) authors such as Herrick, Goldsmith and Shakespeare (...)" será possível identificar pares hiperónimo/hipónimo como os seguintes: (author,Herrick), (author,Goldsmith) e (author,Shakespeare), que nos permite saber também que, por exemplo, Goldsmith é um autor, permitindo facilmente extrair uma hierarquia de conceitos através destes padrões. De uma maneira muito simples, será possível representar o conceito "Autor" com algumas instâncias de autores, de tal maneira que se poderá abstrair a situação representada no diagrama da Figura 11.

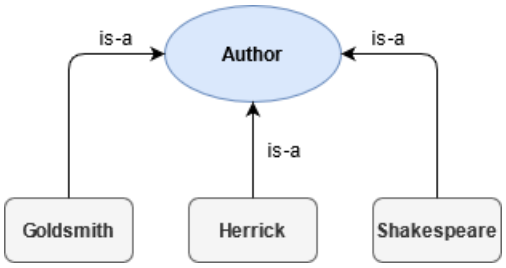


Figura 11: Esquema de uma ontologia criada a partir da aplicação de padrões propostos por Hearst.

No que concerne à aplicação de padrões léxico-sintáticos na língua portuguesa Machado and Strube de Lima (2015) propuseram um conjunto de padrões, que se poderão consultar na Tabela 2, de forma a melhor generalizar uma extração de pares hiperónimo e hipónimo aplicado ao português.

Tabela 2: Padrões léxico-sintáticos propostos por Machado Machado and Strube de Lima (2015).

| Regra | Padrão |
|---|---|
| i | SN (,)* como (SN,)* (SN(elou))* SN |
| ii | SN (,)* (tais\|tal) como (SN,)* (SN(elou))* SN |
| iii | SN (,)* incluindo (SN,)* (SN(elou))* SN |
| iv | SN (,)* especialmente (SN,)* (SN(elou))* SN |
| v | (SN (oulel,))* (outroloutra)(s)? SN |
| vi | tipo(s)? de SN : (SN,)* (SN(elou))* SN |
| vii | SN (,lélsãolforam)? chamad(olaloslas) (de)? (SN,)* (SN (elou))* SN |
| viii | SN ((,)? também)? (,lélsãolforam)? conhecid(olaloslas) como (SN,)* (SN (elou))* SN |
| ix | (SN (oulel,))* (qualquerlquaisquer) outr(ola)(s)? SN |
| x | SN é (ola) SN |
| xi | SN é (umluma) SN |
| xii | SN são SN |

Os padrões apresentados são fruto da inovação e adaptação de outros trabalhos realizados no mesmo âmbito Machado and Strube de Lima (2015). Através da utilização destes padrões anteriores é possível generalizar uma extração de pares para grande parte do discurso na língua portuguesa.

A título exemplificativo, para a frase “A América do Sul é composta por vários países como o Brasil, Equador e Bolívia.”, ao se analisar a Tabela 2 será possível verificar que, devido ao uso do “como” na frase, ao aplicar o padrão 1 serão retirados os seguintes pares: (país, Brasil), (país, Equador) e (país, Bolívia). Ou seja, em termos da estrutura de uma ontologia, mais uma vez poder-se-á considerar "País" como um conceito, enquanto que os diversos países mencionados serão instâncias desta classe, de tal maneira que o seguinte esquema da ontologia, presente na Figura 12, representa esta situação retratada.

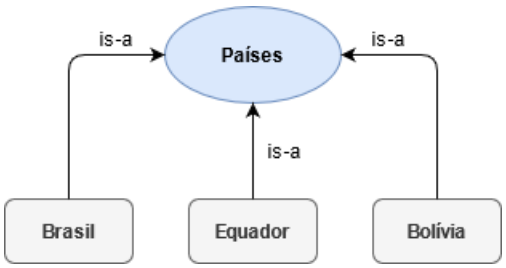


Figura 12: Esquema de uma ontologia criada a partir da aplicação de padrões propostos por Machado.

Portanto, como é possível verificar a aplicação de padrões léxico-sintáticos é adequada e útil, principalmente para a extração da hierarquia de conceitos a partir do texto, ou seja, a identificação de conceitos e a posterior

identificação, por consequência, das relações entre os conceitos através do estabelecimento das relações do tipo "is-a".

### 3.2.3 *Extração de Relações*

Geralmente, após a extração dos conceitos, estes encontram-se organizados por *clusters* sem qualquer tipo de ligações entre si, pelo que é primordial, no contexto da estrutura da ontologia, estabelecer relacionamentos entre os termos que foram extraídos a partir do processo anterior.

A exceção à regra verifica-se através da utilização dos padrões léxico-sintáticos. Uma vez que ao extrair os conceitos, através do uso de padrões, ao mesmo tempo que são extraídos os conceitos são também extraídas as relações que ligam os termos entre si.

#### *Recency Vector*

A extração de relações poderá ser suportada pela frequência de distribuição Kaushik and Chatterjee (2018). O propósito desta técnica assenta no cálculo dos vetores de posição no texto de cada um dos termos, que representam os conceitos, extraídos anteriormente. Ou seja, para um termo w, identificado anteriormente, são definidos dois vetores. O primeiro vetor definido é relativo à posição do termo w no corpo de texto:

$$P = (p1, p2, p3, ..., pn) \tag{1}$$

Em que n representa a n-ésima ocorrência do termo w no texto e, pn a n-ésima posição do termo w no texto. O segundo vetor definido corresponde ao *recency vector*, no qual se pretende determinar o quão recente é que o termo w ocorre no texto. Para o cálculo deste vetor é necessário utilizar o vetor definido anteriormente e, de seguida, são subtraídas as posições consecutivas do vetor de posição:

$$RC = (p2 - p1, p3 - p2, ..., pn - pn - 1) \tag{2}$$

Após a obtenção do *recency vector* para um termo w, as relações entre os termos são identificadas ao comparar os *recency vectors* entre os diferentes termos, uma vez que é assumido que termos relacionados possuem estes vetores semelhantes.

#### *Term Subsumption*

Outro método, que faz uso de modelos probabilísticos, é o *term subsumption* Fotzo and Gallinari (2004). O objetivo deste método é permitir uma generalização e aprendizagem de uma hierarquia de conceitos de forma automática. É assumido que termos que ocorrem com mais frequência numa coleção de dados conferem um maior grau de importância no texto. Portanto, estes termos mais frequentes podem até mesmo definir o tema do texto, enquanto que outros termos menos frequentes podem apenas existir para descrever alguns aspetos acerca do domínio de conhecimento.

É com esta base que se definiu a técnica de *term subsumption*, na qual a relação entre dois termos, relações taxonómicas "is-a", é determinada tendo por base a análise da coocorrência destes dois termos nos dados textuais (documentos). Por exemplo, para dois termos t1 e t2, o termo t1 subsume o termo t2 "is-a(t2,t1)", se o termo t1 ocorre em todos os documentos que o termo t2 aparece também. A seguinte fórmula matemática retrata o cálculo necessário para calcular a medida de subsunção de dois termos.

$$\begin{aligned} n(t1,t2) &= \text{número de documentos que contêm o termo t1 e o t2} \\ n(t2) &= \text{número de documentos que contêm o termo t2} \\ P(t1|t2) &= n(t1,t2)/n(t2) \end{aligned} \tag{3}$$

Poder-se-á também considerar que o *threshold* não seja tão alto no que toca à subsunção do termo t1 em t2. Para tal adiciona-se um *threshold*, representado pelo símbolo t na equação 4, pré determinado de tal maneira que:

$$P(x|y) > t\% \tag{4}$$

Para além disso, também é necessário verificar que o termo t1 possui uma maior coocorrência que o termo t2, de maneira a garantir a subsunção, ou seja:

$$P(t1|t2) > P(t2|t1) \tag{5}$$

Ambos os métodos apresentados são adequados para a extração de relações taxonómicas "is-a". Para o caso da extração de relações não taxonómicas, e como seria de esperar, a extração deste tipo de relações é mais complexa, podendo ser obtida através do uso de mecanismos supervisionados de programação em lógica indutiva Lima et al. (2019).

### 3.2.4 *Extração de Axiomas*

Tal como na extração de relações não taxonómicas, a extração de axiomas também se adivinha como uma tarefa complicada. Tal como para o caso anterior da extração de relações não taxonómicas, também podem ser aplicados mecanismos de programação em lógica indutiva de maneira a permitir a extração de axiomas. Outras técnicas também já foram usadas com sucesso para permitir esta extração como, por exemplo, os *Hidden Markov Models* (*HMM*).

Os *Hidden Markov Models* tratam-se de um tipo de modelo das cadeias de *Markov* Jiomekong et al. (2019), que representam um autómato de estados composto por uma sequência de estados ligados entre si através de arestas. Cada aresta contém informação acerca da probabilidade de transição entre estados. O modelo descreve um processo probabilístico, no qual o comportamento do processo a cada estado t é apenas dependente do estado anterior, verificando-se a seguinte equivalência:

$$P(qt|q1,q2,...,qt-1) = P(qt|qt-1) \tag{6}$$

Para cada estado, num momento t, é também gerado um *output*, ou observação, ot. A distribuição de probabilidades associada é apenas dependente do estado atual, e não dos estados ou observações anteriores, ou seja:

$$P(ot|o1,..,ot-1,q1,...,qt) = P(ot|qt) \quad (7)$$

????

Esta técnica foi aplicada com sucesso num domínio de conhecimento associado à linguagem de programação *JAVA* Jiomekong et al. (2019). Nesta aplicação, após uma análise da linguagem, definiram-se 4 estados, que cobrem a generalidade da sintaxe da linguagem, de maneira a definir o autómato. O primeiro estado, PRE, corresponde ao conhecimento antecessor, o segundo estado TARGET corresponde ao conhecimento que se pretende captar, o terceiro estado POST representa o conhecimento posterior e, por fim, um estado OTHER para tratar e caraterizar situações anómalas nas quais o conhecimento não antecede nem precede o TARGET.

De seguida, para cada um destes estados foram definidas *keywords* correspondentes às palavras chaves do domínio, como, por exemplo, para o PRE podem ser definidas *keywords* correspondentes à restrição de acesso como *public* e *private*. O objetivo final do *Hidden Markov Model* passa pela descoberta da melhor sequência de estados que explica a sequência de observações. Esta sequência de estados escondidos, mais prováveis, poderá ser determinada através do algoritmo de *Viterbi* Forney (1973). De notar que esta técnica, para além da capacidade para extrair axiomas, também poderá ser aplicada de maneira a extrair classes e relações.

A última fase deste processo de desenvolvimento é relativa à avaliação da ontologia criada. Tal como foi referido anteriormente, existem diversas técnicas que podem ser utilizadas para validar a ontologia, como as técnicas *data* e *application driven*, tal como foi descrito na secção 2.4.

## 3.3 APLICAÇÕES E SISTEMAS

Tal como é o caso das potenciais aplicações das ontologias criadas de forma manual, na aprendizagem de forma semiautomática de ontologias a maior parte das possíveis aplicações mantêm-se com a vantagem de possuir menos custos.

Não obstante, e num caso concreto, a extração semiautomática de ontologias é relevante no que toca à caraterização e etiquetação de documentos. Para tal, os mecanismos de construção de ontologias de forma semiautomática podem ser aplicados a cada um dos documentos que se pretende caraterizar e etiquetar. O resultado final desta fase será um conjunto de ontologias, que representam o domínio de conhecimento embebido em cada um dos documentos dados como *input*. Com a obtenção das várias ontologias, através de medidas de semelhança do conhecimento representado na ontologia, ou seja, através da análise da estrutura das ontologias, será possível posteriormente agrupar os documentos, consoante o cálculo da similaridade das ontologias.

A extração de ontologias também é relevante na criação de mecanismos de exploração de documentos. Através da exploração e inferência na ontologia é possível criar mecanismos de visualização do conhecimento presente na ontologia, o que poderá resultar numa descoberta de nova informação que não seria tão observável em antemão.

Devido às vastas aplicações, e áreas, que estas técnicas de extração semiautomática possuem, são vários os sistemas já criados tendo por base estes princípios.

No caso de Kaushik Kaushik and Chatterjee (2018) o propósito por detrás da criação de um sistema que permite a extração semiautomática de uma ontologia encontra-se na necessidade de estruturar dados relacionados com a agricultura, com o propósito de mais tarde serem efetuadas interrogações acerca da ontologia. O esquema proposto por estes autores encontra-se dividido em duas fases.

Na primeira fase o objetivo final passa pela extração do vocabulário, ou seja, os termos/conceitos mais pertinentes nos dados textuais. Para tal, fez-se uso do *RENT algorithm* que se trata de um conjunto de expressões regulares específicas ao domínio em questão. Neste domínio os termos são retirados conforme as expressões definidas, ou seja, ao se definir um conjunto de palavras-chave é possível retirar termos pertinentes no texto. Como exemplo, a seguir à palavra *cultivation* é expetável que se sigam um conjunto de termos relevantes em relação ao cultivo. Estes termos extraídos são depois pesados de acordo com algumas métricas, como através do cálculo da frequência de aparecimento no texto, de maneira a diminuir ao máximo o ruído representado na forma de termos irrelevantes, que poderão ter sido extraídos através de alguma expressão regular.

Com o conjunto de termos/conceitos segue-se a fase seguinte, a de extração de relações. Aqui, os autores optaram por utilizar um esquema de *Open Information Extraction*, que denominaram de mOIE, visto que não necessitavam nem usavam conhecimento prévio acerca das relações existentes entre os termos. Uma das abordagens utilizadas para a extração de relações é baseada principalmente em estatística e assenta no cálculo de vetores de posição para cada termo extraído anteriormente, esta técnica que foi abordada anteriormente na secção 3.2.3.

Ayadi Ayadi et al. (2019), optaram por utilizar técnicas de *Deep Learning*, juntamente com técnicas de *Natural Language Processing* de maneira a desenvolver um sistema capaz de extrair e classificar instâncias que estivessem presentes num dado texto para povoar uma ontologia já existente. As instâncias extraídas são depois analisadas por algum especialista de maneira a depois poderem ser utilizadas para integrar a ontologia.

Nesta abordagem, inicialmente o texto é pré-processado, utilizando-se técnicas de *Natural Language Processing* para remover algumas inconsistências e simplificar o texto. Por exemplo, através da remoção de *stop words*. Além disso, e de modo a preparar o texto para o algoritmo de *Deep Learning*, é aplicado nele uma *tokenization* bem como uma normalização.

De seguida com os dados já pré-processados, os dados são fornecidos como *input* a uma rede neuronal, utilizando-se o algoritmo *Word2vec* de maneira a produzir *word embeddings*. *Word embedding* trata-se de uma representação do texto num formato vetorial num espaço de representação. Quanto mais próximos se encontram os vetores mais semanticamente próximas são as palavras. Além disso, também são calculados os *embeddings* correspondentes aos conceitos representados na ontologia, com o intuito de os comparar com os outros já extraídos. Quanto mais semelhantes forem os vetores mais relacionados se encontram os elementos, desta maneira é possível associar as instâncias extraídas com os respetivos conceitos representados na ontologia.

## 3.4 FERRAMENTAS

Relativamente a ferramentas desenvolvidas neste âmbito uma das mais relevantes é a *Text2Onto* Cimiano and Völker (2005). Esta ferramenta disponibiliza uma plataforma de aprendizagem semiautomática de uma ontologia a partir de texto. A ferramenta combina técnicas de *Natural Language Processing* com técnicas de *Data Mining*. Para cada fase do processo de extração é possível escolher diversos algoritmos que melhor se adequam ao problema, por exemplo, a extração de conceitos poderá ser efetuada através do uso do *C/NC-Value* ou *Tf-idf*. Ambas as abordagens são estatísticas. As relações taxonómicas, que contribuem para o estabelecimento de uma hierarquia de conceitos, são extraídas através do uso de padrões léxico-sintáticos, enquanto que as não taxonómicas são extraídas através do uso de regras de associação Al-Arfaj and Al-Salman (2015).

O *OntoGen* Fortuna et al. (2007) é outra ferramenta, que se diferencia devido ao facto de ser um editor gráfico semiautomático de ontologias. O seu propósito será ajudar no processo de criação de uma ontologia, através da sugestão de possíveis classes e relações, com base no texto dado como *input*. As sugestões são depois aceites, recusadas ou até mesmo ajustadas manualmente pelo utilizador. O processo de sugestão da ferramenta é feito através de processos supervisionados ou não supervisionados. Os métodos não supervisionados, por exemplo, geram uma lista de termos/conceitos através do uso de algoritmos como o *k-means clustering*. Os métodos supervisionados tipicamente requerem que o utilizador tenha conhecimento acerca do domínio em causa, sendo usados métodos como o *Support Vector Machine Active Learning*. Neste método são colocadas questões ao utilizador e consoante as respostas o sistema refina a sua procura por conceitos.

Existem também várias bibliotecas que podem ser usadas em conjunção com alguma linguagem de programação. De referir o *spaCy* Honnibal et al. (2020) e o *NLTK* Bird et al. (2009). Ambas as bibliotecas foram desenvolvidas para a linguagem *Python*, podendo ser facilmente acedidas através do *Python Package Index* (PyPi).

Tanto o *spaCy* como *NTLK* oferecem mecanismos de processamento natural de linguagens, de tal maneira que facilmente se poderá efetuar o *tokenization*, o *part of speech tagging*, a *lemmatization*, bem como um reconhecimento das entidades (*named entity recognition*) presentes no texto, de uma maneira intuitiva. Relativamente ao *spaCy*, são oferecidos e disponibilizados modelos já pré treinados, prontos para serem aplicados a qualquer tipo de dados textuais. Existem alguns modelos treinados para várias linguagens. No caso do Português, existem três modelos treinados com um número crescente de dados, dados esses que foram retirados a partir de um conjunto de notícias. O *NLTK* também disponibiliza um modelo, também já pré treinado, para a língua portuguesa, com um *treebank* denominado floresta, que é utilizado para a análise morfossintática dos dados. O *treebank* poderá ser utilizado para, por exemplo, proceder a um *part of speech tagging* de uma certa frase. Ambos também podem ser treinados com *custom data*. Nestes casos, utilizando o *spaCy*, a título exemplificativo, facilmente poderá ser estabelecido um *pipeline* do modelo com apenas os componentes relevantes, no qual não será necessário, por exemplo, carregar e treinar o componente de *named entity recognition* caso o utilizador não o necessite.

# 4

# EXTRAÇÃO DE UMA ONTOLOGIA DE MEDICAMENTOS

## 4.1 CONTEXTUALIZAÇÃO E APRESENTAÇÃO DO PROCESSO

No que concerne à ferramenta concebida, com o intuito de permitir uma aprendizagem e extração de forma semiautomática de uma ontologia a partir de dados textuais, nas seguintes secções será exposto o processo de criação desta solução.

Os dados textuais utilizados remontam aos séculos XVI-XVII e foram extraídos a partir de um conjunto de textos de culinária, agricultura e medicina Barros (2013, 2014, 2016). Nestes textos estão incluídos medicamentos e mezinhas da medicina tradicional e acreditada à época, bem como as dietas e alimentos salutíferos, utilizados em termos preventivos e adequados a cada doente.

Por exemplo, no pequeno trecho que apresentamos de seguida, podemos identificar um conjunto de vários ingredientes que compõem uma receita específica, mais concretamente a receita de "Prioris":

*"Macela, Coroa de Rei, Maluaisco, ou maluas, cosido tudo com agoa e metido este cosimento em hua bexiga de boi posta sobre a pontada."*

Os ingredientes referidos são depois associados à receita de onde foram extraídos, que depois permitirão fazer a composição de uma ontologia simplificada, o que permitirá representar o domínio, com o aspeto apresentado na Figura 13.

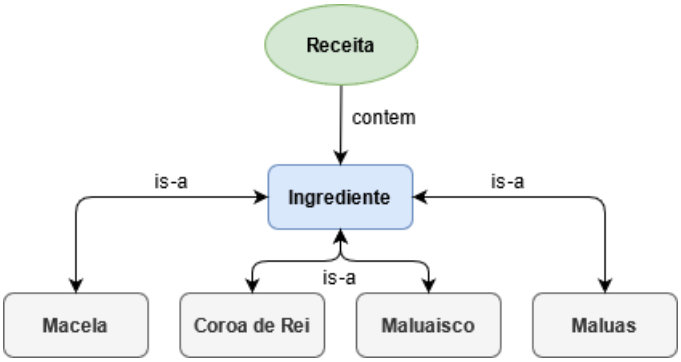


Figura 13: Esquema de uma ontologia de medicamentos.

A principal função da ontologia extraída a partir destes textos será servir como base para a criação de um sistema de exploração. Assim, será possível efetuar interrogações sobre o conhecimento da ontologia, com o intuito de permitir uma visualização, de forma efetiva e sustentada, dos dados relativos aos diversos medicamentos, ingredientes e as suas respetivas propriedades e processos de preparação. Para além disso, será também possível identificar as relações existentes entre remédios referidos no texto, os quais não seriam tão facilmente identificadas sem o uso de uma ontologia sobre o seu domínio de conhecimento.

No que concerne ao processo aplicado, neste caso particular, poder-se-á considerar que o processo se desenrola ao longo de quatro fases, na prática reduziram-se em apenas três, uma vez que juntamos a fase de extração de relações juntamente com a de extração de termos/conceitos do texto, fruto do método aplicado. As quatro fases foram motivadas a partir dos requisitos abordados anteriormente, os quais foram necessários para que a ontologia pudesse ser utilizada para responder a interrogações sobre os dados.

Assim, no diagrama de atividade apresentado na Figura 14 podemos ver uma esquematização das fases do processo de geração semiautomático de ontologias. Posteriormente, cada uma destas fases será discutida com maior detalhe.

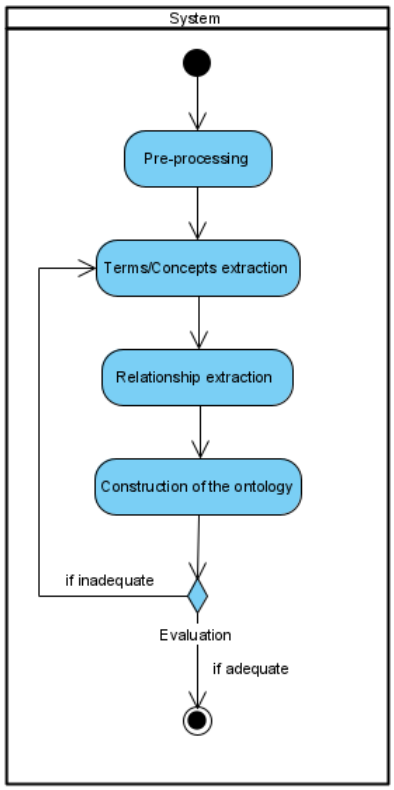


Figura 14: Diagrama das principais fases aplicadas no processo de geração da ontologia de medicamentos.

## 4.2 TECNOLOGIAS E BIBLIOTECAS UTILIZADAS

Antes de começar a conceção do sistema foi primeiro necessário pesquisar e identificar tecnologias que pudessem ajudar nas diversas ações de processamento do texto que precisávamos de realizar. Dentro das várias linguagens de programação, e as principais bibliotecas de *Natural Language Processing*, optou-se pela a escolha da linguagem *Python* como base para o desenvolvimento dos nossos programas e o *spaCy* como a biblioteca de apoio às tarefas de processamento e extração do texto. A escolha da linguagem de programação deveu-se, essencialmente, à sua popularidade, como a linguagem mais usada para a criação de soluções de *machine learning*, sucesso esse que em parte é devido à quantidade de bibliotecas que estão disponíveis e que facilmente poderão ser acedidas através do *Python Package Index* (PyPi).

Em relação à biblioteca de *Natural Language Processing* o *spaCy* foi escolhido, primeiro devido ao bom desempenho que tem demonstrado nas suas aplicações, quando comparado com outras soluções semelhantes, como o *NLTK*. Por exemplo, o *spaCy* processa mais palavras por segundos, mantendo ao mesmo tempo um bom desempenho Honnibal et al. (2020). Além disso, o *spaCy* oferece também *pipelines* (modelos) pré-treinados, de tamanho variáveis, prontos para ser aplicados em qualquer tipo de dados textuais. Através destes *pipelines* será possível aceder a vários componentes úteis para a tarefa de *Natural Language Processing*, como componentes de *tokenization*, *part of speech tagging*, *named entity recognition*, entre outros.

O *spaCy* oferece três modelos para o processamento de textos em língua portuguesa, pré-treinados, com dados provenientes de fontes de notícias. Para o contexto do nosso trabalho optou-se por utilizar o modelo médio, *pt_core_news_md*. Embora o modelo escolhido tenha sido treinado com dados do português atual, este revelou-se na mesma adequado para os dados em português antigo que foram usados neste caso de estudo. Para além disso, foram também efetuados testes com o modelo *pt_core_news_sm*. Contudo, este, devido ao menor número de dados com que foi treinado, não se revelou tão adequado no que diz respeito à precisão nas das suas previsões, quando comparado com o *pt_core_news_md*.

## 4.3 TÉCNICAS UTILIZADAS NA EXTRAÇÃO DE CONCEITOS E RELAÇÕES

O processo de extração de termos/conceitos e relações para compor a ontologia apenas poderá ser realizado, com aproveitamento, se os dados textuais se encontrarem coesos, simples e preparados de tal maneira que seja simples a aplicação dos sucessivos algoritmos.

Para que tal aconteça, primeiramente, serão discutidas as técnicas que foram utilizadas para permitir um pré processamento adequado do texto. De seguida será apresentada a técnica utilizada para permitir uma extração simultânea dos conceitos bem como a respetiva hierarquia de conceitos.

### 4.3.1 *Pré Processamento*

No que respeita ao pré-processamento do texto, antes de se aplicar alguma técnica foi preciso estudar com atenção os textos disponíveis e entender quais as técnicas é que poderiam ser aplicadas, de forma a distorcer o

mínimo possível o significado do texto. Isto é fundamental para que a ontologia final seja fiável e o mais próxima possível do domínio do conhecimento que se pretende representar.

Uma vez que os dados disponíveis estão escritos em português antigo, optou-se por restringir o pré processamento ao mínimo possível, de maneira a manter o significado do texto intacto. Como tal, o pré-processamento iniciou-se com a eliminação de espaços e de mudanças de linha. De seguida, foram removidos os caracteres especiais do texto, dado que estes interferiam com a *tokenization* e a análise de *Part of Speech* feita pelo *spaCy*. Foram removidos caracteres especiais como apóstrofos, que são identificados como um *token* individual no modelo pré treinado do *spaCy*, juntamente com outros casos indesejáveis para o processamento, como a inclusão de certos padrões no texto como, por exemplo, "[3]", que são padrões que não oferecem significado à frase. Além disso, foram também efetuadas algumas substituições no texto como, por exemplo, em certos casos existiam palavras como "b[e]ba", o que não representa uma palavra válida. Como tal, na palavra anterior, os parênteses retos foram substituídos por caracteres vazios, o que produziu a palavra "beba", uma palavra que já é válida.

Como exemplo, tome-se em consideração a seguinte receita, dada como *input* deste módulo:

*"Outro.*
*[10]*
*Pª resoluer nascidas, farinha de semeas, oleo Rosado, vinho. Outro. Cosimento de barbasco, oleo rosado com hu' pouco de asafraõ, farinha de trigo galego feito huas papas. Outro pera faser arebentar em 24 horas. hua gemma de ouo anasada com asucar mto be' ate q' fique grosa; estendase em hu' paninho, e ponhase, renovandose como estiuer seca, e ella faz buraco, e chama a mala, e continuena depois de aberto, o tempo q' quisere', ꝑq' atte a rais chupa."*

O módulo de pré processamento, através dos métodos criados para o efeito, processou o texto anterior, o que resultou no seguinte *output*:

*"Outro.P resoluer nascidas, farinha de semeas, oleo Rosado, vinho. Outro. Cosimento de barbasco, oleo rosado com hu pouco de asafraõ, farinha de trigo galego feito huas papas. Outro pera faser arebentar em 24 horas. hua gemma de ouo anasada com asucar mto be ate q fique grosa; estendase em hu paninho, e ponhase, renovandose como estiuer seca, e ella faz buraco, e chama a mala, e continuena depois de aberto, o tempo q quisere, pq atte a rais chupa."*

Como é possível observar, o texto anterior será interpretado com mais facilidade pelo modelo escolhido no *spaCy*, sem que houvesse um impacto significativo no significado textual. O texto foi, portanto, simplificado em termos de estrutura, pelo que o texto convertido passou a ser um texto corrido. Para além disso também é possível observar algumas substituições, como a substituição de caracteres especiais, tais como os apóstrofos. Em cada uma das receitas existe também um identificador da receita, um código, que é utilizado para efetuar a separação dos textos por receitas. No final do processamento, esta separação irá gerar uma lista na qual, em cada índice, é guardado o texto já processado. Após a obtenção da lista dos textos já processados, a lista é percorrida e cada texto individual é separado em frases. Embora o *spaCy* ofereça componentes para detetar e

segmentar as frases, a segmentação produzida pelo *spaCy* em certos casos produziu frases incorretamente segmentadas, pelo que foi implementada uma segmentação básica através do caractere ponto final.

Cada um dos textos processados foram guardados numa base de dados de documentos, em *MongoDB*. A adoção do *MongoDB* deveu-se, essencialmente, à sua escalabilidade e rapidez Gyorodi et al. (2015), tanto na inserção de documentos, como na sua posterior pesquisa efetuada num sistema de exploração de dados.

A Figura 15 sintetiza na forma de um diagrama de atividade o processo completo envolvido no pré processamento do texto desde a remoção de espaços até à inserção de cada um dos remédios separados na base de dados referida anteriormente.

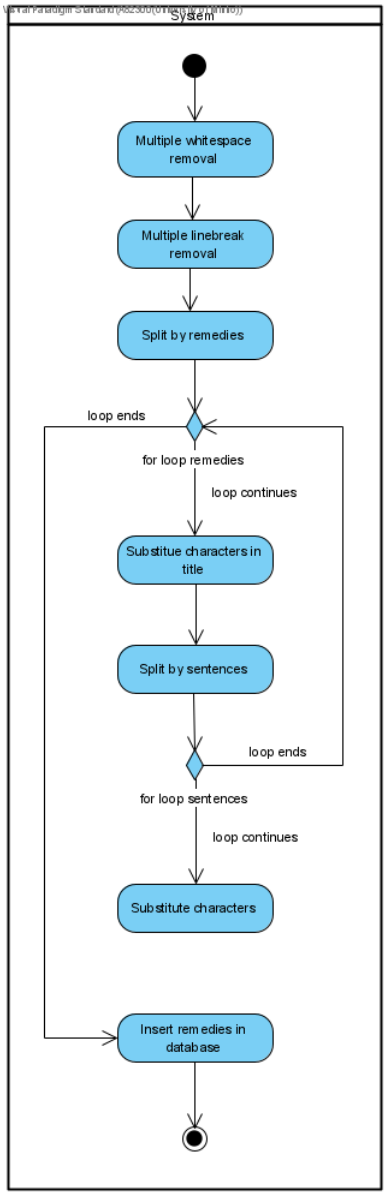


Figura 15: Processo completo do pré processamento do texto.

### 4.3.2 *Extração de conceitos e relações*

Após termos obtidos os textos processados, foi possível nesta instância aplicar o modelo *spaCy* escolhido para o efeito. O modelo foi aplicado a cada uma das frases, individualmente, em cada um dos textos que foram escolhidos para a obtenção da ontologia. Para tal, foi definido um *custom pipeline* de um modelo, onde um componente particular do pipeline, o *Named Entity Recogniton*, foi treinado com dados particulares e relativos à problemática.

Embora ainda não tenha sido utilizado, o componente de *Named Entity Recogniton*, este foi treinado com os dados presentes nos textos fornecidos. Para isso, foi necessário anotar os dados de maneira a que depois fosse possível realizar a aprendizagem com estes dados de maneira a treinar este componente. Desta maneira, é possível esquematizar o *pipeline* do modelo personalizado, representado na Figura 16, utilizado na extração de conceitos e relações da seguinte forma.


Figura 16: *Pipeline* de componentes do modelo utilizado.

Tal como mencionado, embora a componente de *ner* do modelo não tenha sido utilizada para efeitos da extração, devido ao foco em outras soluções de extração, esta encontra-se pronta e treinada de tal maneira que permitirá a classificação de *custom entities*, como entidades de ingrediente e de medicamento. Contudo, será necessário fornecer mais dados de treino, de maneira a tornar mais fiável o reconhecimento por parte deste componente.

Em termos do funcionamento geral do modelo, após cada frase ser dada como *input* ao modelo, o primeiro componente do modelo o *tok2vec* (*token-to-vector*) é responsável por separar o texto em *tokens* individuais. De seguida, cada um dos *tokens* é representado em termos de um vetor num espaço de representação, ou seja é representado na forma de um *word embedding*. A representação vetorial é atingida através da utilização de uma rede neuronal, mais propriamente uma *CNN* (*Convulutional Neural Network*). Depois a rede neuronal e os *embeddings* são partilhados nos sucessivos componentes o que traz algumas vantagens, como o tamanho reduzido dos modelos, uma vez que os *embeddings* são apenas calculados e copiados uma vez, razão pela qual também existem ganhos temporais ao utilizar uma solução deste género.

De seguida o *morphologizer* é responsável por classificar e identificar as *Part of Speech tags* de cada *token*. O modelo standard do *morphologizer* é feito através da construção de um modelo *tagger*, que utiliza o componente *tok2vec* de forma a adicionar uma nova camada à rede, com uma função de ativação *softmax*, de maneira a produzir as previsões das *Part of Speech tags*. Depois de identificadas as *tags* segue-se a aprendizagem da dependência entre os diversos vetores, através do componente *parser* ou *DependancyParser*. Mais uma vez é

reutilizado o *tok2vec*, de maneira a criar um novo modelo denominado de *Transition Based Parser*. Este modelo tanto pode ser utilizado para o *Named Entity Recognition* como para o *Dependancy Parsing*.

Um dos componentes mais importantes no contexto do problema é o *attribute_ruler*. Este componente funciona como um motor de pesquisa no texto. Por fim, o *lemmatizer* simplifica o texto, reduzindo as palavras à sua forma básica. No final é produzido um objeto *doc*, que contém uma sequência de objetos *token* relativos a cada palavra. Para cada objeto *token* será possível consultar informação referente ao output de cada componente da pipeline. Pelo que será possível consultar a *Part of Speech tag*, a entidade caso tenha sido detetada, a forma *lemmatized* da palavra, entre outros.

De maneira a realizar a extração de conceitos e relações, foram utilizados padrões léxico-sintáticos, que foram construídos através do uso de *Part Of Speech tags*, devido ao maior grau de controlo que estes oferecem, ao contrário do que acontece com as expressões regulares normais. O *spaCy* disponibiliza um objeto *matcher* que é utilizado para encontrar *strings* nos textos que cumpram com algum tipo de padrão. Os padrões desenvolvidos são depois utilizados em conjunção com o objeto *matcher*.

O processo de extração de conceitos e as suas respetivas relações foi iniciado com uma análise manual dos textos, de maneira a interpretar e identificar possíveis padrões léxico-sintáticos para extrair os pares hiperónimo/hipónimo. Tendo em conta os requisitos e o papel da ontologia, a extração foi focada no levantamento dos ingredientes que compõem uma dada receita, bem como a posterior identificação das quantidades e métodos de confeção associados, caso existissem.

Para a identificação dos ingredientes, foi possível, primeiramente, assumir a existência do hiperónimo comum a todos, com o valor de ingrediente, faltando apenas agora extrair os hipónimos relacionados com este hiperónimo a partir do texto. De maneira a que consiga explicar melhor a extração de hipónimos e o trabalho associado a esta tarefa, na Figura 17 poderá ser consultado na forma de um diagrama de atividade uma esquematização do processo geral.

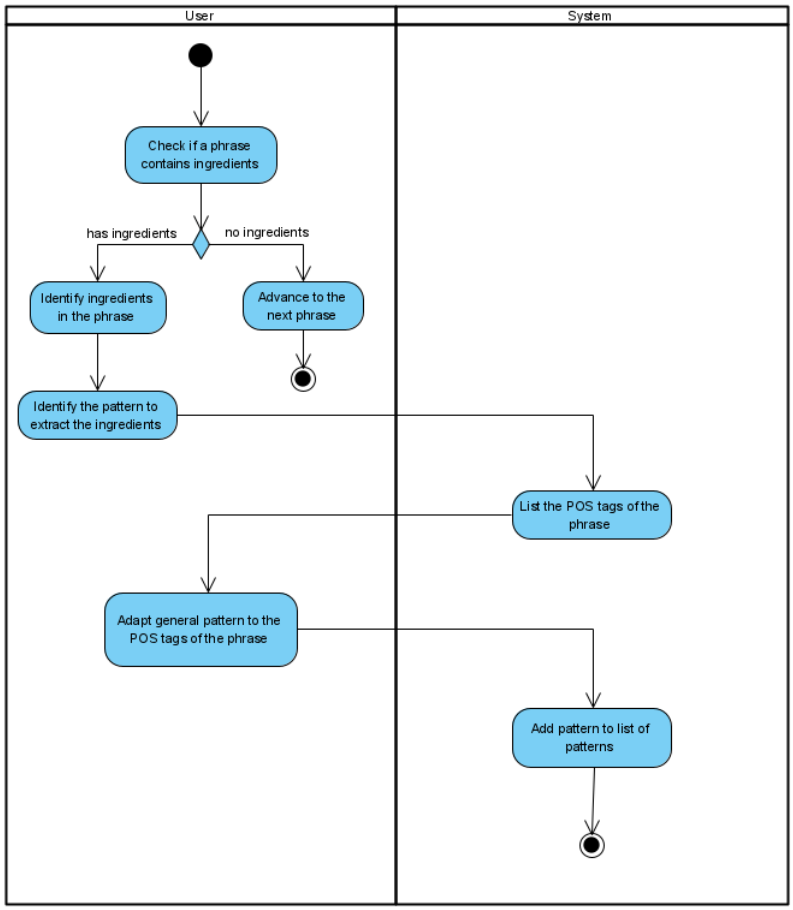


Figura 17: Diagrama do processo geral associado à criação de padrões léxico-sintáticos.

Para complementar a explicação do processo envolvido na criação de padrões léxico-sintáticos apresentamos uma frase, que inclusivamente já foi apresentada na secção 4.1, presente num das receitas que foi utilizada no decorrer da extração de uma ontologia:

*"Macela, Coroa de Rei, Maluaisco, ou maluas, cosido tudo com agoa e metido este cosimento em hua bexiga de boi posta sobre a pontada."*

Assim, o primeiro passo que realizámos foi a identificação dos ingredientes de forma a que fosse possível fazer a composição de um padrão. Na frase anteriormente apresentada é possível identificar quatro ingredientes para integrar a composição de um padrão léxico-sintático, nomeadamente:

*"**Macela**, **Coroa de Rei**, **Maluaisco**, ou **maluas**, cosido tudo com agoa e metido este cosimento em hua bexiga de boi posta sobre a pontada."*

De seguida, foi necessário identificar os elementos para compor um padrão que permitisse extrair os ingredientes identificados anteriormente. Como podemos verificar, no caso da frase anterior, os ingredientes

identificados encontram-se todos antes das palavras chaves "cosido tudo", pelo que podemos verificar e generalizar, com alguma certeza, que se poderão encontrar ingredientes antes das palavras "cosido tudo". Depois de identificado o padrão, seguiu-se uma consulta às restantes receitas pelo padrão definido, de maneira a ajustá-lo face aos problemas que poderão acontecer posteriormente. Uma vez que este padrão para diferentes contextos poderá conter algumas alterações mínimas, portanto é necessário contemplar os diferentes contextos. Agora com o padrão identificado segue-se a construção do padrão, veja-se:

*"**Macela**, **Coroa de Rei**, **Maluaisco**, ou **maluas**, **cosido tudo** com agoa e metido este cosimento em hua bexiga de boi posta sobre a pontada."*

Depois de termos identificado as palavras chaves do padrão, passámos à consulta da informação do objeto *doc*, proveniente da análise do *spaCy* para a frase que foi utilizada, mais propriamente é elaborada a consulta das *Part Of Speech tags*. De forma a simplificar a criação dos padrões foram definidos três estruturas típicas de padrões, estas são ajustadas conforme o caso que se pretende contemplar.

De maneira a generalizar a criação de padrões, definiram-se padrões com blocos contendo as *tags* mais comuns (e mais desejáveis) para suportar o processo de extração. Os blocos na sua estrutura possuem, tipicamente, quatro *tags*. Nestes blocos de quatro *tags*, que depois serão repetidos consoante o formato do padrão que se pretende implementar, encontram-se destacadas quatro *tags* relativas a nomes (*NOUN*), a nomes próprios (*PROPN*), a adjetivos (*ADJ*) e verbos (*VERB*). O próprio formato dos padrões, a sua estrutura, foi inspirado pelos padrões criados por Machado Machado and Strube de Lima (2015) que poderão ser consultados na Tabela 3.

Tabela 3: Exemplos de alguns padrões léxico-sintáticos criados.

| Regra | Padrão |
|---|---|
| i | com(er\|laõla) (B1,)* (B2(elou))* B3 |
| ii | such NP as {NP,}* {(or\|and)} {NP} |
| iii | (B1,)* tomado |
| iv | beb(er\|er\|alaõla\|ida\|ido) (B1,)* (B2(elou))* B3 |
| v | NUM (onsas\|onças) de (B1,)* (B2(elou))* B3 |
| vi | (NUM)? (onsa\|onça) de (B1,)* (B2(elou))* B3 |
| vii | (B1,)* ((elou)B2)* NUM (onsas\|onças) PUNCT |
| viii | (B1,)* ((elou)B2)* (NUM)? (onsa\|onça) PUNCT |
| ix | mesturarlheaõ (B1,)* (B2(elou))* B3 |
| x | (B1,)* ((elou)B2)* (mesturada\|misturese) con (B3,)* (B4(elou))* B5 |
| xi | (B1,)* ((elou)B2)* tudo mesturado |
| xii | B1 misturandoo (B2,)* (B3(elou))* B4 |

Na Tabela 3 é possível, através dos padrões criados, identificar alguns dos ingredientes referidos no texto. Quase todos os padrões seguem a mesma estrutura, existindo algumas diferenças relativas ao local, antes ou depois das palavra chave, no qual aparecem os ingredientes. Neste caso, B(1..N) é utilizado para identificar os blocos de *tags* utilizados para compor a estrutura do padrão. Os blocos variam de caso para caso, mas em muitos desses casos, tal como foi dito anteriormente, um bloco envolve tipicamente a repetição de dois grupos *NOUN/PROPN/ADJ/VERB*, com uma *tag* de conjunção (*ADP*) a dividir ambos os grupos.

Em certos casos as estruturas dos padrões gerais foram também simplificadas, uma vez que não existiu a necessidade de introduzir complexidade adicional à construção de um padrão utilizado para casos simples. Para esses casos, o padrão foi simplificado, adotando o formato: palavra-chave (B1,)* ((e|ou)B2)*. Tendo em conta esta generalidade, foram identificados outro conjunto de padrões. Alguns exemplos desses padrões léxico-sintáticos mais simplificados estão apresentados na Tabela 4.

Tabela 4: Exemplos de alguns padrões com padrões léxico-sintáticos gerais simplificados.

| Regra | Padrão |
|---|---|
| i | (lauar com\|lauall\|allaueno) (B1,)* ((e\|ou)B2)* |
| ii | traga (B1,)* ((e\|ou)B2)* |
| iii | esfregalas com (B1,)* ((e\|ou)B2)* |
| iv | emsima (B1,)* ((e\|ou)B2)* |

Existem também alguns outros padrões que não obedecem a nenhum caso geral identificado. São padrões que foram criados para situações particulares e, por vezes, únicas, que não se repetem em mais nenhum lugar nos dados utilizados.

No que diz respeito ao processo próprio da extração de termos/conceitos, no qual se estabelecem por consequência, as relações, existe um conjunto de tarefas associadas a esta extração, que podem ser esquematizadas no diagrama de atividade apresentado na Figura 18.

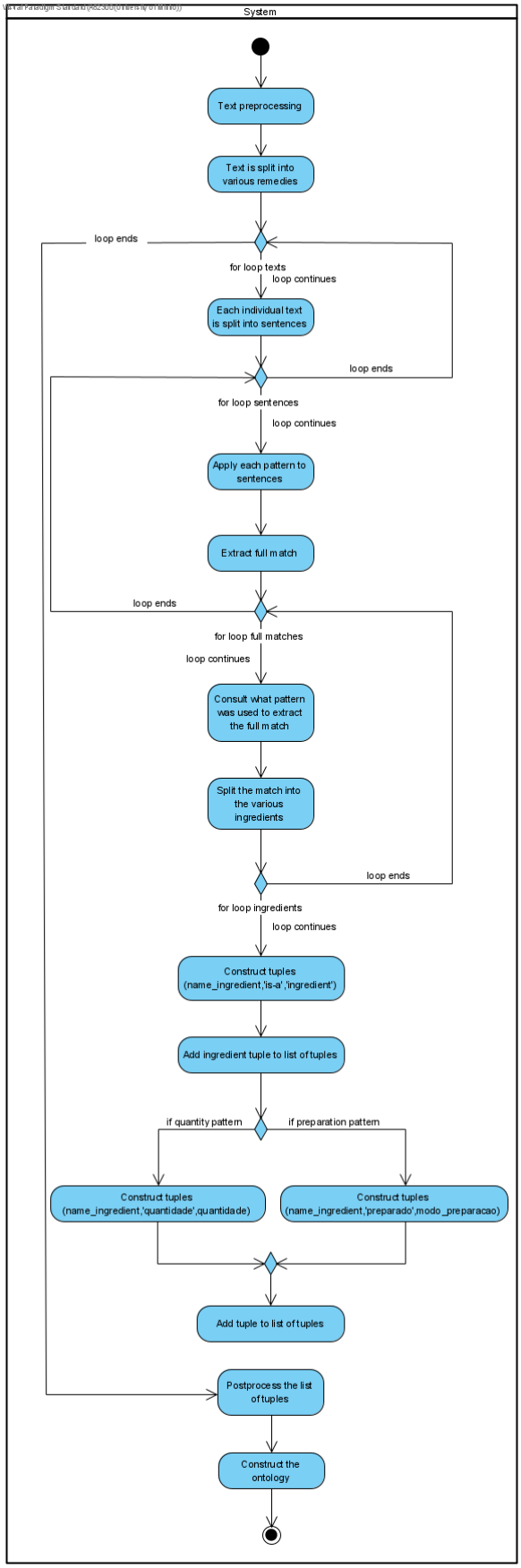


Figura 18: Processo completo da extração de termos/conceitos e relações.

Portanto, tal como era expectável, após a obtenção das frases, partindo dos textos processados, seguiu-se a aplicação de cada um dos padrões desenvolvidos a cada uma das frases dos textos usados no processo de extração. De seguida, caso existisse um *match* de algum padrão para numa dada frase, seria extraída uma lista contendo cada um dos *matches* desse padrão na frase utilizada. A esta lista de *matches* seria aplicado um filtro para retirar ocorrências únicas, uma vez que, até chegar ao *full match*, existem vários *matches* intermédios. Como exemplo, com a frase anterior, foi retirada uma lista com os seguintes *matches*:

*['Macela, Coroa de Rei, Maluaisco, ou maluas, cosido tudo', ', Coroa de Rei, Maluaisco, ou maluas, cosido tudo', 'Coroa de Rei, Maluaisco, ou maluas, cosido tudo',... ]*

O filtro que foi aplicado inicialmente ordenou a lista obtida, através da aplicação do dado padrão à frase, de acordo com os tamanhos das strings, e fazendo-se de seguida a remoção de todas as *substrings* através de um método de compreensão de listas. O resultado da aplicação deste filtro foi armazenado numa lista contendo cada *full match* único.

Cada *full match*, através da aplicação do filtro anterior, foi depois percorrido e, conforme o padrão utilizado, fizeram-se sucessivos *splits* do *full match* de maneira a separar os ingredientes presentes neste. Para que fossem realizados os *splits*, foi necessário orientá-los, numa primeira instância, através de uma separação do *full match* pela palavra chave utilizada no padrão que extraiu esse mesmo *match*. Desta maneira, foi obtida uma sub *string* contendo apenas os ingredientes presentes nesse *match*. No caso anterior foi feito um *split* utilizando as palavras chave "cosido tudo", o que deu origem a uma lista de ingredientes, na qual no primeiro índice está contida a *sub string* correspondente aos ingredientes. Como evidência desse processo, veja-se o caso retratado anteriormente:

*['Macela, Coroa de Rei, Maluaisco, ou maluas, ', ' ']*

De seguida, e de acordo com a posição na qual se encontra a *string* que contém os ingredientes, neste caso no primeiro índice da lista, nesta *string* foi verificado o caso em que esta possui vírgulas. Depois, verificou-se a existência de conjunções como o "e" e o "ou". No caso mais simples, a extração dos ingredientes é direta sem serem efetuados *splits*, uma vez que para este caso não existem conjuções nem vírgulas. Depois, foram também contemplados os casos em que apenas existiam vírgulas, ou os casos mistos em que existiam vírgulas juntamente com alguma das conjunções (ou ambas). Para a *string* utilizada como exemplo, far-se-á um *split* tanto por vírgulas como pela conjunção ou, o que fará com que seja possível observar a seguinte lista resultante deste *split*:

*['Macela', ' Coroa de Rei', ' Maluaisco', 'maluas']*

Seguidamente foram construídos tuplos (nome do ingrediente, 'is-a', 'Ingrediente'), de acordo com o resultado dos *splits* feitos a partir do processo anterior. Além do mais, e de modo a complementar ainda mais a ontologia final, em certos casos e de acordo com a finalidade de alguns padrões, foi adicionada informação relacionada com a quantidade de um determinado ingrediente no contexto da receita e, também, o próprio modo de confeção do ingrediente, por exemplo frito, cosido, entre outros.

Esta extração de informação adicional foi apoiada devido ao uso de certos padrões. Para a quantidade, padrões como “onsas/onças” do padrão v da Tabela 3, contêm informação acerca da quantidade. A extração dos modos de confeção foi obtida por padrões em que é explícito o modo de preparação do ingrediente. Como no caso anterior, no padrão cosido tudo foi diretamente retirada a informação relativa ao modo de preparação dos ingredientes, que neste caso foram cozidos. Esta informação adicional foi depois adicionada à mesma lista dos ingredientes com os seus respetivos tuplos. A lista de tuplos gerada para a frase anterior foi a seguinte:

*[('Macela', 'is-a', 'Ingrediente'), ('Macela', 'preparado', 'cosido'), ('Coroa de Rei', 'is-a', 'Ingrediente'), ('Coroa de Rei', 'preparado', 'cosido'), ('Maluaisco', 'is-a', 'Ingrediente'), ('Maluaisco', 'preparado', 'cosido'), ('maluas', 'is-a', 'Ingrediente'), ('maluas', 'preparado', 'cosido')]*

Este processo foi depois repetido para as diversas frases. Tendo a lista de ingredientes e as suas propriedades, o passo seguinte passou pelo pós processamento da lista de ingredientes, em que foram aplicados alguns métodos para formatar a lista de ingredientes e as suas propriedades.

No pós processamento da lista de tuplos, aplicaram-se alguns métodos de forma a realizar tarefas como a separação de ingredientes compostos em múltiplos ingredientes individuais, ou até mesmo a substituição de caracteres ou *strings*. No que concerne à separação de ingredientes, veja-se o seguinte exemplo:

*agoa Rosada com hua drama de almesega em po* $\xrightarrow{\text{split por com}}$ *agoa Rosada && hua drama de almesega em po*

Os tuplos que são alvos de um pós processamento são depois substituídos pelo resultado do pós processamento. Para o caso anterior, visto que foi efetuado um *split*, é substituído o tuplo original com o resultado do *split* anterior ao "com", ou seja é substituído pelo valor agoa Rosado, de tal forma que ('agoa Rosada','is-a','Ingrediente'). Posteriormente é também adicionado mais um tuplo com o valor à frente do "com", um tuplo ('hua drama de almesega em po','is-a','Ingrediente').

## 4.4 CONSTRUÇÃO DA ONTOLOGIA

No que concerne à representação dos elementos, provenientes da lista de tuplos, na forma de uma ontologia optou-se por representar num grafo implementado em *Neo4J*. O *Neo4J* trata-se de um sistema de gestão de base de dados *NoSQL* que permite acolher base de dados orientadas ao grafo Santos López and Santos De La Cruz (2015). Basicamente, a estrutura desta base de dados é composta pela conjugação dos nodos do conhecimento integrado na ontologia e as suas relações, podendo existir atributos associados a cada um desses nodos ou às relações, descrevendo as propriedades de cada um desses elementos.

Assim, tendo em conta o tipo de tuplos presente na lista proveniente do processo de extração já pós processada, em primeiro lugar foi efetuada a criação dos nodos relativos aos remédios/medicamentos e os ingredientes. Aquando da criação do nodo relativo ao ingrediente também é estabelecida diretamente uma relação entre o nodo relativo ao remédio ao qual pertence o ingrediente (que foi criado anteriormente) e o nodo do ingrediente. Após a criação dos nodos e dos seus relacionamentos, fizemos o processamento das propriedades contidas

na lista de tuplos, de forma a que fosse possível armazenar a informação acerca da quantidade e do modo de confeção do ingrediente, respetivamente.

Na Figura 19 é possível observar quatro nodos relativos a quatro remédios/medicamentos distintos, tendo cada um deles os seus ingredientes associados. Ademais, também é possível ver que alguns dos ingredientes são partilhados pelos quatro remédios representados. De forma a obter uma representação tal como se poderá observar na Figura 19 foi necessária a criação de uma *constraint* relacionada com o nome do ingrediente, de tal que maneira que não se repita a inserção de ingredientes com o mesmo nome.

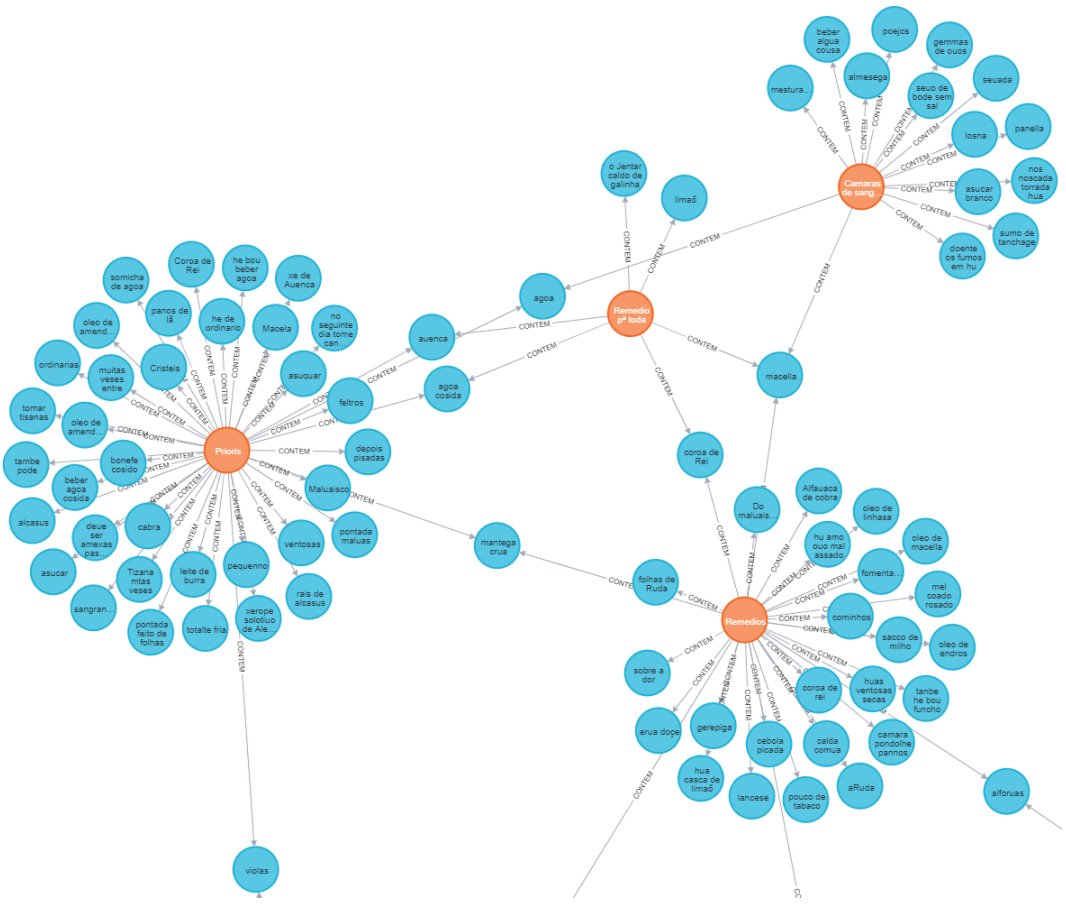


Figura 19: Extrato da ontologia criada contendo apenas a representação de quatro remédios, contendo a sua caracterização e relacionamentos.

O constraint referido anteriormente é útil no que toca a avaliação dos resultados através da ferramenta de visualização. Contudo, esta restrição posteriormente foi removida, uma vez que também existe uma propriedade nos nodos representativos dos ingredientes relativa ao modo de confeção. Assim, com a restrição do nome do ingrediente não se tem em conta as informação adicional do ingrediente nas receitas nas quais ele poderá aparecer.

## 4.5 AVALIAÇÃO DOS RESULTADOS

Através da aplicação dos métodos apresentados anteriormente foi possível gerar uma ontologia de medicamento. A ontologia foi extraída a partir de um conjunto de 68 receitas distintas. No total, o conjunto de receitas apresenta cerca de 473 frases, o que implicou o processamento e análise de 8466 palavras.

Em termos do número de palavras, a maior receita processada foi a receita "Tabardilho". Esta receita é composta por um conjunto de 20 frases, que incorporavam 732 palavras na estrutura da receita. Por sua vez, a receita que possui um maior número de frases é a receita "Remedios pª a Colica", que integra 26 frases com 295 palavras.

Quanto aos menores números verificados no processo de extração, o medicamento no qual se pode verificar um menor número de palavras processadas é o "Cameras de quentura", que implicou apenas o processamento de 9 palavras, juntamente com 2 linhas. Relativamente ao medicamento composto por um menor número de frases, existem vários medicamentos compostos por apenas duas frases, nos quais se contabilizaram neste cálculo também a frase correspondente ao título do medicamento. Aqui, pode-se verificar que o medicamento anterior também possui um número reduzido de frases, sendo composto por apenas duas frases, apenas uma é dedicada à receita, se for removida a frase relativamente ao nome do ingrediente.

Através da aplicação dos padrões léxico-sintáticos, foi possível identificar no conjunto de receitas processadas cerca de 813 ingredientes, nos quais 750 são ingredientes distintos. A receita na qual foram identificados o maior número de ingredientes foi a receita que também possuía o maior número de palavras processadas, a receita "Tabardilho". Nesta receita foram identificados 44 ingredientes. A receita com o menor número de ingredientes identificados corresponde também à receita com o menor número de palavras processadas (e linhas), ou seja, a receita "Cameras de quentura", que teve apenas um ingrediente identificado. O ingrediente que mais foi classificado, isto é, o ingrediente que aparece num maior número de receitas, foi o "macella", que foi utilizado em 5 receitas distintas. Por outro lado, existem também vários outros ingredientes que podemos considerar como incomuns. Estes ingredientes foram utilizados em apenas uma receita. Veja-se, por exemplo, o ingrediente "tigella de caldo temperado", o qual apenas poderá ser encontrado na receita "Regimento das Amexas de sene – Purga Suave".

Os dados apresentados ao longo desta análise estão esquematizados, e sumarizados, na Tabela 5.

Tabela 5: Dados retirados acerca do processo de extração.

| | | |
|---|---|---|
| Dados gerais extração | Num receitas | 68 |
| | Num frases | 473 |
| | Num palavras | 8466 |
| | Num ingredientes únicos | 750 |
| | Num total de ingredientes | 813 |
| Remédio com maior/menor num palavras | Maior | Tabardilho (732 palavras) |
| | Menor | Cameras de quentura (9 palavras) |
| Remédio com maior/menor num frases | Maior | Remedios pª a Colica (26 frases) |
| | Menor | Cameras de quentura (2 frases) |
| Remédio com maior/menor num ingredientes | Maior | Tabardilho (44 ingredientes) |
| | Menor | Cameras de quentura (1 ingrediente) |
| Ingrediente mais comum/incomum | Maior | Macella (5 receitas) |
| | Menor | tigella de caldo temperado (1 receita) |

No que diz respeito à avaliação concreta da ontologia, e como foi possível constatar na secção 2.4, existem uma plenitude de métodos de avaliação que podem ser aplicados. No entanto, optámos por focar a avaliação da ontologia apenas num conjunto de seis textos, correspondentes a seis medicamentos/receitas. Em termos gerais, na escolha dos textos, o principal critério foi a extensão dos textos, devido à maior complexidade inerente, bem como à sua maior completude no processo de avaliação. Fizemos isso em vez de escolher textos com um menor número de palavras, o que não permitiria proceder a uma avaliação tão exaustiva devido à falta de dados. Em relação à avaliação, esta foi focada na verificação e validação dos conceitos identificados como ingrediente, mais concretamente o seu nome.

Assim, de acordo com o critério anterior, para cada um dos textos escolhidos, foram verificados e validados os resultados de forma manual. Neste processo, anotou-se o número de conceitos corretamente classificados, bem como o número total de conceitos, com o objetivo de calcular a precisão do método de extração. Além disso, contabilizou-se também o número de ingredientes não identificados pelo sistema e analisou-se a sua precisão, tendo em conta estes ingredientes em falta.

Na equação 8 podemos ver a forma como é medida a precisão dos resultados verificados, isto é, a percentagem de conceitos identificados corretamente, para se analisar a correlação entre o número de ingredientes corretamente classificados com o número de ingredientes identificados no total.

$$\text{Precisão} = \frac{N_{ingredientes\_corretos}}{N_{ingredientes\_totais}} \tag{8}$$

Utilizando a Equação 9 podemos analisar a precisão dos resultados, tendo em conta o número de conceitos que não foram identificados pelo sistema. Para obter este número de conceitos foi necessário proceder a uma análise manual dos textos.

$$\text{Precisão(Falta)} = \frac{N_{ingredientes\_corretos}}{(N_{ingredientes\_totais} + N_{ingredientes\_falta})} \quad (9)$$

Utilizando-se as equações anteriores foi possível verificar os seguintes resultados para o dado conjunto de textos utilizados, apresentados na Tabela 7. Os seis textos utilizados para a avaliação da ontologia correspondem, respetivamente, às receitas dos seguintes medicamentos: "Remedios para Cameras", "Prioris", "Tabardilho", "Pª Sangue dos Narises", "Pª Dor de Olhos", "Pª Lombrigas".

Tabela 6: Resultados da Avaliação dos seis textos selecionados.

| | Texto 1 | Texto 2 | Texto 3 | Texto 4 | Texto 6 | Texto 7 |
|---|---|---|---|---|---|---|
| Número de palavras | 414 | 463 | 732 | 267 | 268 | 214 |
| Precisão | 85% | 82.5% | 81.82% | 91.3% | 78.79% | 81.81% |
| Precisao Falta | 69.39% | 68.75% | 70.59% | 75% | 72.22% | 75% |
| Precisao Média | 83.54% | | | | | |
| Precisao Falta Media | 71.83% | | | | | |

No processo de avaliação dos resultados não foi tido em conta a estrutura da palavra. Se o sistema identificasse o nome do ingrediente juntamente com outras palavras não referentes ao próprio ingrediente, este seria na mesma contabilizado como um conceito corretamente classificado. Por exemplo, no caso do texto número cinco, correspondente à receita "Pª Dor de Olhos", o conceito "xerope Rosado lançando" foi classificado como um conceito correto, apesar de o ingrediente conter uma palavra a mais. Ora, através do uso de padrões, ao fazer a sua generalização, é normal que em certos casos os ingredientes não sejam totalmente corretos quanto à forma, embora na prática correspondam essencialmente ao ingrediente que se pretende identificar.

Devido ao uso desta metodologia pode-se considerar que os resultados encontram-se dentro do esperado, pelo que a precisão média do processo de extração resultou num valor de 83.54%, o que indica que embora exista ruído (ingredientes incorretamente classificados) estes não distorcem em demasia os resultados. Portanto, os dados extraídos por parte dos padrões, na sua maioria, identificam corretamente os ingredientes. Por outro lado, quando se tem em conta os ingredientes que não foram identificados, a precisão para os 6 textos analisados nunca foi menor que 68%, tendo sido verificado o seu valor mínimo no texto 2. A média dos seis textos foi de 71.83%. Este valor indica que, na grande maioria, os principais ingredientes foram identificados nos textos. Contudo, como seria de esperar, em certos casos não foi possível criar qualquer tipo de padrão para identificar os ingredientes, o que impossibilitou a sua extração.

# 5

# EXPLORAÇÃO DOS DADOS

## 5.1 FUNDAMENTAÇÃO DO SISTEMA

De forma a permitir uma exploração mais sustentada e efetiva acerca da informação armazenada na base de dados, com o intuito de revelar informação específica acerca dos ingredientes, em particular, a sua quantidade e o seu modo de confeção, idealizámos e implementámos um sistema de exploração especialmente orientado para o manuseamento do conhecimento adquirido com a ontologia gerada. Este sistema foi concebido como um meio para auxiliar o processo de análise dos resultados obtidos. Tendo isso em consideração, nesse sistema integrámos dois métodos de pesquisa.

Com o primeiro método de pesquisa conseguimos disponibilizar o conjunto de ingredientes, métodos de confeção e quantidades (caso existam) para as receitas de medicamentos que foram processadas e cujo conhecimento adquirido foi colocado numa base de dados *Neo4J*, enquanto que com o segundo método de pesquisa, podemos escolher e analisar o conjunto de receitas em que certo ingrediente é incorporado na sua confeção.

Em suma, através da utilização desta ferramenta de exploração de dados, é possível facilmente analisar e verificar os dados provenientes do processo de extração e consultar receitas/medicamentos relacionados entre si de uma forma simples e intuitiva. Sem esta ferramenta os processos de consulta e análise da ontologia criada seriam uma tarefa bastante árdua de realizar e levar a bom termo.

## 5.2 ESTRUTURA FUNCIONAL DO SISTEMA

De forma a proceder com criação do sistema de auxílio à exploração de dados foi necessário identificar as tecnologias de apoio ao desenvolvimento, bem como idealizar uma estrutura aplicacional que permitisse uma maior persistência, escalabilidade e resistência a potenciais falhas que pudessem eventualmente ocorrer.

### 5.2.1 *Tecnologias Utilizadas*

A escolha de cada uma das tecnologias e das ferramentas a utilizar assentou, mais uma vez, num conjunto particular de requisitos funcionais e operacionais, nomeadamente o grau de escalabilidade, o grau de utilização na indústria, a eficiência, a rapidez e o desempenho das ferramentas.

Assim, para o ambiente de execução e de consumo dos dados utilizámos a tecnologia de *Javascript server-side*, o *Node.js* Shah and Soomro (2017). Esta tecnologia foi utilizada no decorrer da implementação dos serviços no *backend*. A escolha recaiu sobre o uso desta tecnologia uma vez que responde a muito dos requisitos mencionados anteriormente, para além de que utiliza o *Javascript* como a linguagem de programação. Desta maneira, é possível então utilizar a mesma linguagem de programação tanto no desenvolvimento *client side* bem como no *server side*. Para além disso, um bom desempenho está assegurado com o uso destas tecnologias, devido ao modelo *non blocking* de *I/O*, no qual as operações de *input* e *output* são também realizadas de forma assíncrona Shah and Soomro (2017). Também a escalabilidade do sistema fica garantida, uma vez que esta tecnologia é *event-based* ao invés de *thread-based* como outras tecnologias do mesmo âmbito.

Por sua vez, para a implementação do *frontend* utilizou-se o *Vue.js* que se caracteriza como uma *framework* de *Javascript* de código aberto focado no desenvolvimento de interfaces Pšenák and Tibenský (2020). A escolha desta tecnologia foi devido a três fatores essenciais, o primeiro é relativo ao tamanho associado a esta ferramenta, uma vez que esta solução é mais *lightweight* quando comparada com outras semelhantes como o *Angular.js* Jain et al. (2014) e *React.js* Maratkar and Adkar (2021). Para além disso, existe também um grande foco na reutilização de código, o que promove uma maior legibilidade, fruto da arquitetura baseada em componentes, onde é possível registar componentes, que podem representar certos aspetos das páginas como *buttons*, para depois serem reutilizados no *template* das páginas. O último fator associado à escolha desta tecnologia é devido à quantidade de ferramentas e bibliotecas já desenvolvidas de auxílio, como é o caso do *Vuetify* Leider and Leider (2016), que é uma *framework* de *UI* construída, que disponibiliza um conjunto de componentes reativos.

### 5.2.2 *Arquitetura do Sistema*

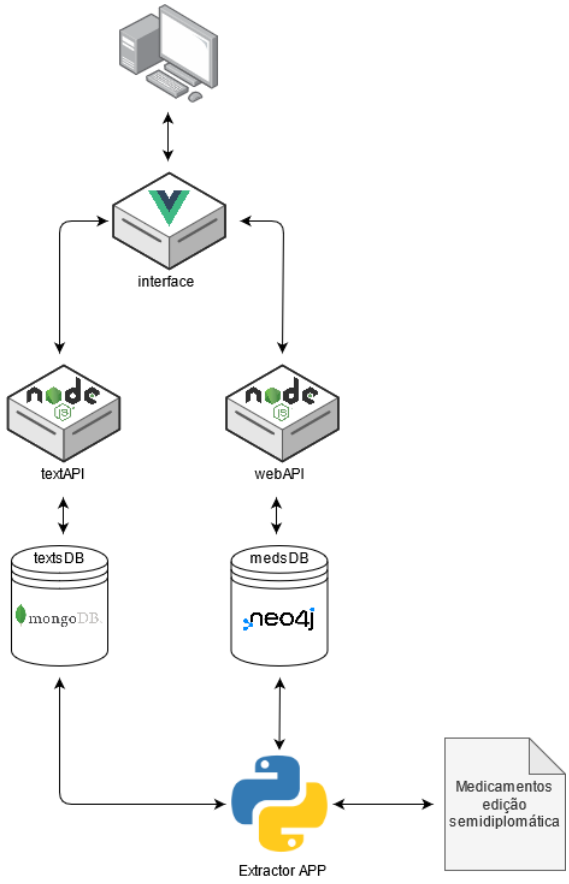


Figura 20: Arquitetura do sistema de exploração dos dados.

Na Figura 20, apresentada anteriormente, podemos observar a arquitetura geral que foi implementada para a criação do sistema de exploração dos dados. No diagrama encontra-se esquematizado os passos iniciais, desde o fornecimento de dados para a ferramenta de extração, um módulo que foi escrito em *Python*, até à página web do utilizador final, que foi desenvolvida através do uso do *Vue.js*.

O sistema assenta numa arquitetura baseada em micro serviços. Os micro serviços permitem suportar uma arquitetura na qual os serviços do sistema são tratados como unidades autónomas e isoladas Viggiato et al. (2018). O objetivo principal deste tipo de arquitetura é o de isolar as *business functionalities* do sistema, de tal forma que cada serviço será independente um do outro, o que permite fazer um desenvolvimento mais isolado e mais modular e posteriormente um *deployment* do sistema independente para cada serviço implementado. Para além das vantagens referidas, existem outras mais, como por exemplo o isolamento de falhas, ou seja, se um micro serviço falhar apenas esse micro serviço será afetado, ficando os restantes serviços do sistema em funcionamento, caso não exista dependência entre os micro serviços criados.

Na Figura 20 podemos observar dois micro serviços essenciais para o funcionamento do sistema. O primeiro serviço denomina-se de *webAPI*, que poderá ser considerado como o micro serviço mais central e fulcral para o funcionamento dos sistema. Este micro serviço é responsável por consumir e servir os dados da base de dados, na qual se encontram guardados os resultados provenientes da extração, ou seja, as receitas e os seus respetivos ingredientes e propriedades.

O segundo micro serviço, *textAPI*, é responsável também por consumir e servir os dados provenientes da base de dados na qual se encontram guardados os textos processados, bem como algumas das propriedades desses próprios textos. Ambos os micro serviços mencionados foram implementados utilizando-se o *Node.js*.

Por fim, e de maneira a possibilitar a visualização dos resultados através de páginas web, criámos um serviço denominado de interface, que foi desenvolvido em *Vue.js*.

## 5.3 BACKEND

No *backend* encontra-se a camada de acesso dos dados, presentes na base de dados, bem como a parte lógica da aplicação que não poderá ser colocada no lado do cliente devido a, por exemplo, questões de segurança.

### 5.3.1 *Base de dados Neo4j*

De forma a manter a persistência dos dados extraídos a partir do módulo de extração, e tal como foi abordado anteriormente, foram definidas bases de dados. A principal base de dados que suporta a exploração dos resultados foi elaborada tendo por base o *Neo4j*, denominando-se de *medsDB*.

O *Neo4j* trata-se de um sistema de gestão de base de dados *NoSQL* que permite acolher base de dados orientadas ao grafo Santos López and Santos De La Cruz (2015). Na estrutura desta base de dados a informação encontra-se organizada em nodos, relações e propriedades. Os nodos descrevem entidades ou conceitos no grafo. A cada um dos nodos podem ser associadas etiquetas de modo a descrever os diferentes papeis que estes têm no sistema. As relações permitem conectar os nodos do grafo, no contexto da ontologia servem para, por exemplo, estabelecer a hierarquia de conceitos. Por sua vez, as propriedades são utilizadas para descrever aspetos tanto dos nodos como das relações.

A linguagem utilizada para efetuar interrogações acerca da informação armazenada no *Neo4j* denomina-se de *Cypher* Francis et al. (2018), pelo que esta linguagem foi inspirada pelo SQL. Na sintaxe das interrogações em *Cypher* é utilizado *ASCII-art* de maneira a estruturar as *queries* da seguinte maneira: (a) -> [r] -> (b), em que “a” e “b” descrevem, respetivamente, nodos do grafo e “r” representa um tipo de relacionamento. De forma a efetuar as interrogações, existe um conjunto de cláusulas como o *Match* que é utilizado para procurar um certo padrão, tal como o anterior, com um funcionamento semelhante ao *SELECT* presente no *SQL*. De forma a obter os resultados de uma interrogação é necessário também inserir a cláusula *RETURN* no final, pelo que depois podem ser especificados que aspetos da interrogação é que se pretendem obter. De forma a modificar o grafo existem também as cláusulas *CREATE*, para criar novos nodos e relações no grafo, *SET* para modificar propriedades, e por fim o *DELETE* para remover nodos do grafo.

Neste caso de estudo, e no que concerne à estrutura do grafo armazenado em *Neo4j*, foram definidos dois tipos de nodo. O primeiro nodo é referente ao tipo "Remedio", que é alusivo às receitas dos medicamentos que posteriormente têm os ingredientes associados. O segundo tipo de nodo é o "Ingrediente", pelo que este tipo de nodo representa os ingredientes extraídos a partir do módulo de extração. Como só existem dois tipos de nodos, apenas foi definido um tipo de relação, a relação "CONTEM" que conecta o nodo "Remedio" com os nodos "Ingrediente". Relativamente às propriedades, estas foram definidas tanto para os nodos como para as relações. Para os nodos do tipo "Remedio" foram definidas duas propriedades, a primeira propriedade "nome" armazena a informação referente ao nome da receita do medicamento, a segunda propriedade "num_medicamento" guarda o código do medicamento, estabelecido aquando da extração. O nodo "Ingrediente" pode, ou não, conter uma propriedade referente ao modo de confeção, denominada de "modo_preparacao". Os relacionamentos também podem conter uma propriedade "quantidade", referente à quantidade do certo ingrediente na receita do medicamento.

Na Tabela 7 é possível verificar o conjunto de propriedades definidas, bem como o tipo de dados associado a cada uma das propriedades.

Tabela 7: Propriedades definidas na base de dados de grafos.

| Propriedade | Domínio | Valor |
|---|---|---|
| nome | Remedio | *string* |
| num_medicamento | Remedio | *string* |
| quantidade | CONTEM | *string* |
| nome | Ingrediente | *string* |
| modo_preparacao | Ingrediente | *string* |

### 5.3.2 *Base de dados MongoDB*

A segunda base de dados utilizada para suportar o sistema de exploração de dados foi concebida utilizando o *MongoDB*, denominando-se de *textsDB*.

O *MongoDB* caracteriza-se como uma base de dados *NoSQL* orientada ao documento, que providencia alto desempenho, alta disponibilidade bem como escalonamento automático Chauhan (2019). Os documentos são uma estrutura de dados composta por pares campo e valor, semelhante a objetos *JSON*. As coleções são utilizadas para armazenar os documentos, estas coleções, por sua vez, podem ser vistas como tabelas numa base de dados relacional. A base de dados é utilizada para armazenar o conjunto de coleções.

Na base de dados *texstDB*, responsável por armazenar os textos associados às receitas processadas bem como informação adicional, foi definida uma coleção com um formato específico de documento. O documento possui na sua estrutura 5 campos distintos. O primeiro campo é relativo ao "_id" que representa o identificador único da receita processada. O segundo campo "code" é relativo ao código do medicamento processado, de

seguida temos o campo “texto” que representa o texto completo da receita processada. Por fim, os campos “num_words” e “num_sentences” são utilizados para armazenar a informação relativa ao número de palavras e frases do texto processado.

Na Figura 21 é possível observar os campos, bem como os tipos de dados, presentes no formato de documento definido.

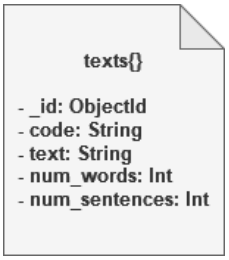


Figura 21: Estrutura do documento definida.

### 5.3.3 *Micro serviço webAPI*

O micro serviço mais fulcral, e essencial, para o funcionamento do sistema é o micro serviço *webAPI*. Tanto este micro serviço, como o *textAPI*, foram criados através do uso do *Node.js* juntamente com a *framework Express.js*, de forma a criar uma *API* de dados para ser utilizada no lado do cliente.

O *webAPI* é responsável por aceder e fornecer os dados provenientes da base de dados *medsDB*, possuindo para tal uma conexão à base de dados *Neo4j*. Assim, este micro serviço serve como uma camada de acesso sobre os dados armazenados na base de dados de grafos. Para tal, e de maneira providenciar os dados, foram definidos dois conjuntos de rotas, com o intuito de elaborar interrogações sobre os dados presentes na base de dados. Os dois conjuntos de rotas foram definidos tendo por base o tipo de nodos presentes na base de dados.

O primeiro conjunto de rotas é relativo aos nodos do tipo “Ingrediente”. Para este conjunto de rotas foi necessário, primeiramente, analisar funcionalmente os requisitos de modo a conseguir ajustar o número de rotas necessárias para o funcionamento da aplicação de exploração. Como tal, podem-se destacar o seguinte conjunto de rotas:

#### *GET ingredientes/*

É extraído o conjunto de informação total relativo aos nodos do tipo “Ingrediente”.

#### *GET ingredientes/mostCommonIngredient*

É disponibilizada a informação relativa ao nodo do tipo “Ingrediente” mais comum na base de dados. De forma descobrir o ingrediente mais comum, apenas é tido em conta a propriedade do ingrediente relativa ao nome. Os resultados do *Match* no final são ordenados de forma descendente.

***GET ingredientes/leastCommonIngredient***

É obtida informação acerca do nodo do tipo “Ingrediente” menos comum na base de dados. De forma análoga ao anterior, também apenas é tido em conta a propriedade do nome do ingrediente. Os resultados do *Match*, por sua vez, são ordenados de forma ascendente.

***GET ingredientes/totalNumberOfIngredients/***

Rota com o objetivo de disponibilizar o número total de ingredientes presentes na base de dados. Para tal, é elaborado um *Match* para cada nodo do tipo “Ingrediente”, depois é contabilizado o resultado desta interrogação através do *COUNT*.

***GET ingredientes/numberOfIngredient/:nome***

Definida com o intuito de contabilizar o número de ocorrências de certo ingrediente nas receitas. Para tal, é necessário disponibilizar um parâmetro para a rota com o nome do ingrediente que se pretende contabilizar. O nome do ingrediente é depois fornecido como parâmetro da *query*, sendo facultado o *COUNT* das ocorrências.

***GET ingredientes/ingredientesRemedio/:code***

Utilizada para obter o conjunto de ingredientes e relações relativos a uma receita de um medicamento de acordo com o parâmetro relativo ao código da receita do medicamento. Assim, é necessário passar como parâmetro da rota o código da receita que se pretende obter os ingredientes, relações e as respetivas propriedades. O código da receita é depois utilizado no *Match* da interrogação, mais propriamente como parâmetro no campo relativo ao nodo do tipo “Remedio”.

***GET ingredientes/ingredientesNoRemedio/:nome***

É facultado o conjunto de ingredientes e relações relativos a uma receita de um medicamento de acordo com o parâmetro do nome da receita do medicamento. Rota análoga no seu funcionamento à anterior, com a diferença de ser utilizado o campo do nome da receita no *Match* dos nodos do tipo “Remedio”, ao invés do código.

***GET ingredientes/multiplosRemedios/:code***

Rota definida com o intuito de permitir uma pesquisa de múltiplas receitas de medicamentos, ao contrário de uma única. Para tal, como parâmetro da rota são dados os códigos das receitas separados por “_”, para cada código é de seguida efetuada uma interrogação de forma a extrair o conjunto de informação relativa aos ingredientes, relações e propriedades dessa receita individual. Os resultados de cada iteração são depois armazenados numa lista.

***GET ingredientes/multiplosIngredientesNosRemedios/:nome***

É obtida a informação relativa à receita, nodos do tipo “Remedio”, onde os ingredientes são incorporados na sua confeção. Rota em que é necessário passar como parâmetro os nomes, separados por “_”, dos ingredientes para os quais se pretende extrair a informação relativa às receitas onde são incorporados. Para cada nome é efetuada uma interrogação de forma a retirar a informação relativa ao nodo “Remedio” ao qual se encontra conectado. Da mesma maneira que a rota anterior, os resultados de cada iteração são armazenados numa lista.

O segundo conjunto de rotas definido é relativo, por sua vez, aos nodos do tipo “Remedio”. Como tal, foram definidas as seguintes rotas:

***GET remedios/***

É obtido o conjunto de informação total relativo aos nodos do tipo “Remedio”.

***GET remedios/:code***

Rota em que é necessário o parâmetro “code” de forma a extrair a informação relativa a uma dada receita de acordo com a propriedade relativa ao código da receita.

***GET remedios/mostLeastIngredients/:code***

Utilizada para extrair, para fins estatísticos, a receita que possui um maior ou menor número de relações. Para tal, como parâmetro da rota é passado um “code” que é relativo ao tipo de operação que se pretende efetuar. Caso se pretenda obter a informação do nodo do tipo “Remedio” com um maior número de relações (ingredientes), é fornecido como parâmetro o número zero. Por outro lado, caso se pretenda obter a informação da receita com o menor número de ingredientes associada é fornecido o parâmetro com o número 1. Na *query* associada a esta rota, é apenas contabilizado o número de relacionamentos associado ao dados nodos do tipo “Remedio”, sendo dado como resultado o maior ou o menor número, de acordo com a operação pretendida.

Todas estas rotas que foram definidas são depois processadas no respetivo controlador, onde são efetuadas todas as interrogações, associadas à rota, na base de dados, de forma a extrair a informação. Os dados obtidos da base de dados são depois formatados, e fornecidos, na forma de um objeto *JSON*.

### 5.3.4 *Micro serviço textAPI*

O segundo micro serviço concebido trata-se do *textAPI*, que é utilizado como camada de acesso aos dados armazenados na base de dados orientada ao documento, a base de dados *textsDB*, possuindo por sua vez uma conexão ao *MongoDB*. Tal como aconteceu no micro serviço anterior, foi definido um conjunto de rotas, e os seus respetivos controladores, de forma a providenciar o conjunto de informação armazenado na base de dados *MongoDB*.

Ao contrário do que aconteceu anteriormente, e dado que apenas existe uma coleção com um tipo de formato da base de dados, foi apenas definido um conjunto de rotas, conjunto esse que se poderá ver de seguida:

***GET textos/***

É dado como resultado o conjunto de documentos relativos a todos os textos presentes na base de dados.

***GET textos/:code***

Rota em que é necessário o parâmetro "code", de forma a poder aceder e retornar a informação relativa ao documento que possui o dado código da receita.

***GET textos/mostLeastWordsLines/:code***

Utilizada para fins estatísticos, como parâmetro da rota é utilizado o "code" que indica o tipo de interrogação que será feito à base de dados. Esta rota é utilizada para extrair os documentos com o maior ou menor número de palavras/frases.

***GET textos/findMultipleTexts/:code***

Rota definida com o intuito de permitir a extração de um conjunto de documentos de acordo com os códigos das receitas. Para tal, como parâmetro da rota é necessário fornecer os códigos das receitas intercaladas com o separador "_". De seguida, para cada código é elaborada uma interrogação à base de dados de forma a extrair o documento associado a esse código. A cada iteração os documentos são armazenados numa lista.

Para todas as rotas foram também definidos um conjunto de controladores, responsáveis por efetuar as interrogações respetivas à base de dados. A informação devolvida pela *API*, tal como no caso anterior, também é devolvida em formato *JSON*.

## 5.4 FRONTEND

De forma a representar a parte gráfica e visual do aplicação de exploração concebeu-se um serviço denominado de interface que é responsável por fornecer as capacidades gráficas da aplicação aos utilizadores.

Para tal, e tal como foi mencionado anteriormente, para a criação deste serviço foi utilizado o *Vue.js* juntamente com uma libraria que disponibiliza componentes gráficos adicionais denominada de *Vuetify*.

O serviço criado tem a responsabilidade de servir como o ponto de contacto do utilizador com o sistema. Assim sendo, foi criado um conjunto de funcionalidades gráficas de forma a permitir que o utilizador interaja com o sistema. Estas interações, por consequência, originam pedidos *HTTP* aos micro serviços do *backend*, de maneira a permitir o acesso aos diversos dados disponibilizados através das chamadas às *APIs*.

Em termos da estrutura deste serviço, a interface é constituída por duas páginas, denominadas de *views*. A primeira página, *home*, é o ponto de contacto inicial de um utilizador com o sistema, e possui o conjunto de

interações que o utilizador poderá efetuar de modo a explorar os dados. A segunda página criada, com o nome de *results*, apenas poderá ser acedida se o utilizador efetuar uma pesquisa. Nesta página *results* será possível consultar os resultados da pesquisa efetuada, onde é possível observar um conjunto de formas de visualização dos resultados.

Assim, dentro das possíveis interações no *frontend* pode-se destacar, essencialmente, dois tipos de operações requisitadas pelo utilizador. A primeira interação é relativa à pesquisa tanto por nome das receitas de medicamentos como por nome dos ingredientes. A segunda interação é relativa à visualização de estatísticas associadas ao processo de extração.

No primeiro tipo de pesquisa, a pesquisa por nome das receitas, poderão ser escolhidas as várias receitas dos medicamentos para as quais se pretende consultar a informação que foi extraída e que está presente na base de dados.

No que concerne à lógica associada a este tipo de pesquisa, após terem sido escolhidos os nomes das receitas, são extraídas informações adicionais a partir destes nomes como os códigos associados às receitas, de forma a permitir mais facilmente a extração da informação associada a estas receitas. Após serem obtidos os códigos associados às receitas escolhidas, o utilizador é reencaminhado para a página *results*. Nesta página, quando esta é criada, é efetuado um primeiro pedido *GET* ao micro serviço da *API* no *backend*, ao *textAPI*, onde através dos códigos é invocada a rota *findMultipleTexts* com os códigos das receitas associados ao parâmetro da rota, rota essa apresentada anteriormente, de forma a retirar os textos associados a cada uma das receitas escolhidas. Após obter a lista de textos, é feito mais um pedido *GET*, desta vez ao micro serviço *webAPI*, através da rota *multiplosRemedios* de maneira a obter a lista de ingredientes, e as suas propriedades, associadas a cada um dos códigos das receitas dos medicamentos. Após ter sido obtido o conjunto de ingredientes, é elaborado um conjunto de operações de modo a preparar os componentes gráficos que são utilizados para a visualização dos resultados. Com base no nome dos ingredientes é possível destacar os nomes dos ingredientes aos correspondentes textos. São também criadas duas listas adicionais, com os nodos e ligações, de modo a formatar os dados para o formato de grafo, componente esse disponibilizado através do *vue-echarts* (Yiling et al., 2021).

Na Figura 22 é possível observar um esquema, na forma de um diagrama de atividade, representativo de todas estas interações.

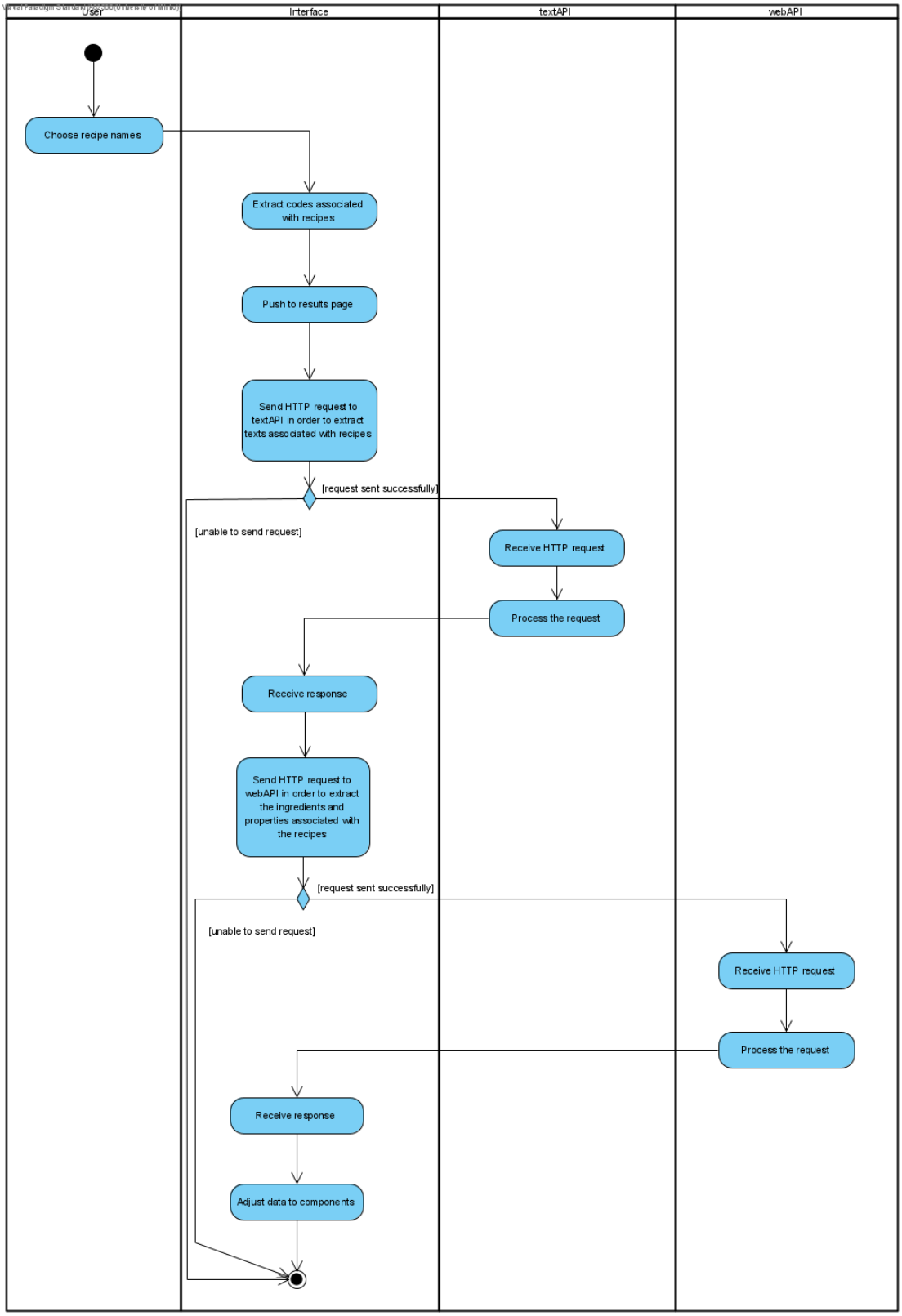


Figura 22: Diagrama de atividade relativo à pesquisa por nome da receita.

Por sua vez, no segundo tipo de pesquisa poderá ser efetuada uma pesquisa por nome dos ingredientes, na qual será possível consultar as receitas nas quais os ingredientes escolhidos são utilizados na sua conceção. Após terem sido escolhidos os nomes dos ingredientes, o utilizador é reencaminhado para a página *results*, onde na sua criação é efetuado um pedido *GET* ao micro serviço *webAPI* através da rota *multiplosIngredientesNoRemedio* de forma a conseguir obter a informação relativa às receitas onde os ingredientes são incorporados na sua confeção. Com este pedido é retirada a informação, para cada ingrediente, acerca do código e o nome do remédio. Com base no código da receita do medicamento, é efetuado um segundo pedido *GET* ao micro serviço *textAPI* através da rota *findMultipleTexts* com os códigos das receitas, extraídas através do processo anterior, de forma a extrair os textos associados às receitas onde se podem verificar os ingredientes escolhidos. De seguida, é efetuado um último pedido de forma a obter a lista total de ingredientes, e propriedades, para o conjunto de códigos, através de um pedido *GET* ao micro serviço *webAPI* com a rota *MultiplosRemedios*. Com a obtenção da informação completa acerca de cada receita, onde são incorporados os ingredientes escolhidos, segue-se um processo semelhante ao anterior, onde os dados são formatados de forma a preencher os componentes gráficos de visualização.

Na Figura 23 é possível ver, de uma forma esquematizada, o conjunto de ações efetuadas numa pesquisa por nomes de ingredientes.

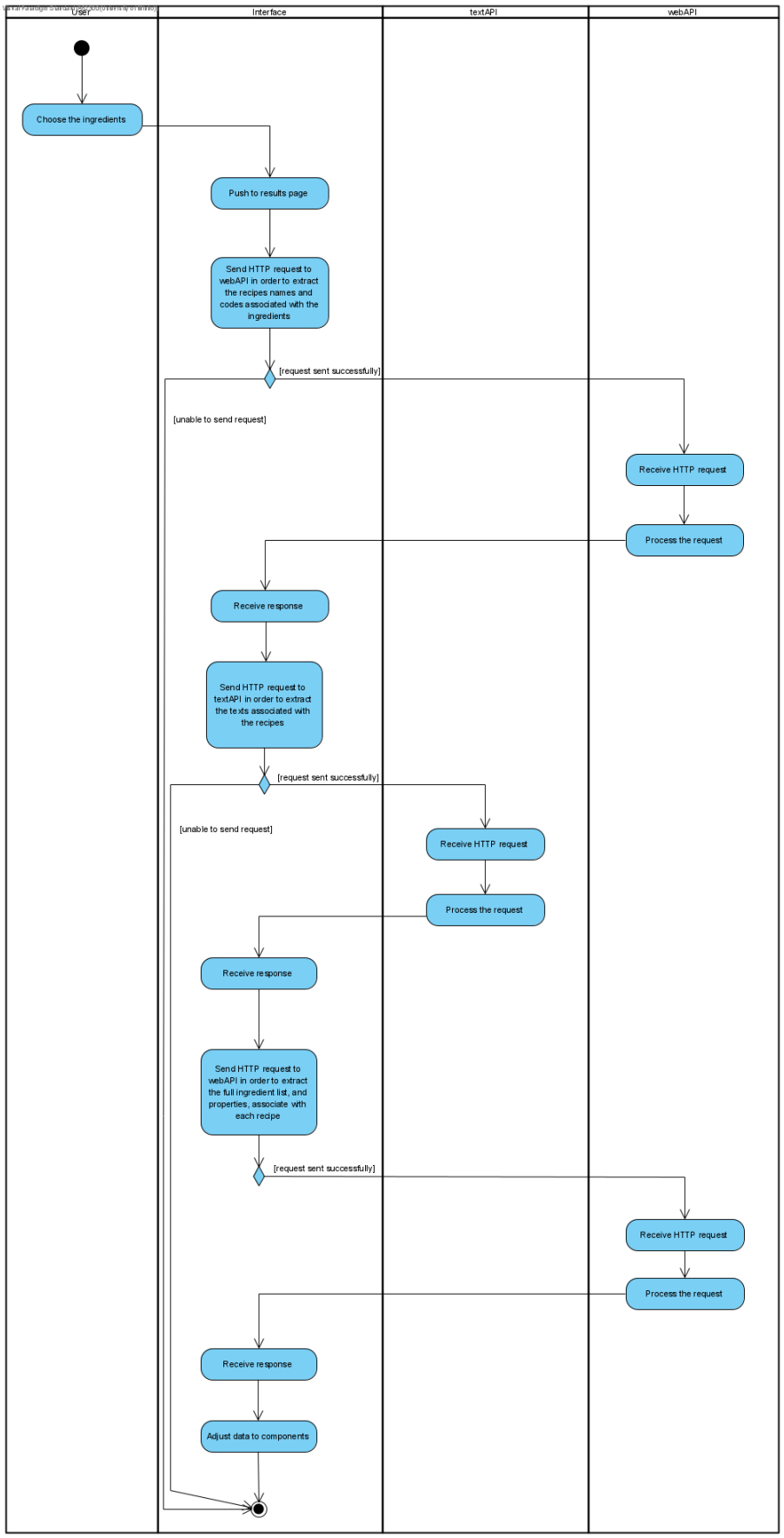


Figura 23: Diagrama de atividade relativo à pesquisa por nome do ingrediente.

O segundo tipo de interação, e tal como mencionado, é relativo à visualização de estatísticas acerca do processo de extração. No que concerne às estatísticas disponibilizadas, pode ser consultado o número de receitas, palavras e frases processadas no total. Podem ser consultadas outras estatísticas como a receita com o maior/menor número de ingredientes associados. Para tal, e como foi possível verificar anteriormente na explicação de cada uma das rotas, existem rotas criadas com o propósito de disponibilizar informação estatística. São utilizados pedidos *HTTP* para as rotas como a *mostLeastWordsLines* do micro serviço *textAPI* juntamente com as rotas *mostLeastIngredients*, *mostCommonIngredient*, *leastCommonIngredient*, *totalNumberOfIngredients* do micro serviço *webAPI*. De acordo com a resposta de cada uma destas rotas, os componentes utilizados para a visualização das estatísticas são atualizados com os dados extraídos a partir dos pedidos anteriores.

## 5.5 A APLICAÇÃO DE EXPLORAÇÃO

Tendo explicado todo o processo lógico, bem como estrutural, em relação à aplicação de exploração seguiu-se a conceção da parte visual da aplicação. Para tal, através do uso da biblioteca *Vuetify* foi possível obter um conjunto de componentes que foram utilizados para construir o esqueleto da página. Após ter sido construído o esqueleto visual da página, seguiu-se a ligação da lógica aplicacional com a parte visual da página. De forma a explicitar o aspeto visual de cada um dos componentes desenvolvidos para a aplicação, vão ser apresentados um conjunto de imagens relativas a cada funcionalidade desenvolvida.

Na Figura 24 podemos ver a forma como pode ser consultada a página *home* da aplicação de exploração. É nesta página que se pode realizar as duas formas de pesquisa referidas anteriormente, utilizando-se para a sua concretização dois campos de texto com a propriedade *autocomplete*. O primeiro campo de texto é relativo à pesquisa por nome da receita e o segundo campo de texto é, por sua vez, relativo à pesquisa por nome do ingrediente. Apenas poderá ser efetuado um tipo de pesquisa, ou seja, não pode ser efetuado duplamente uma pesquisa por nome do ingrediente e nome da receita. Após o preenchimento de um dos campos a pesquisa poderá ser efetuada através do uso do *button*, com a cor amarela.


Figura 24: A principal página do sistema de exploração.

De seguida, na Figura 25, podemos ver o resultado de um processo de pesquisa realizado para uma dada receita de um medicamento. Nessa figura, vemos uma tabela que contém os resultados da extração para um dado medicamento, na qual estão também apresentados os diferentes ingredientes que compõem a receita, bem como os seus modos de confeção e quantidades, caso existam. Nessa figura também podemos ver outra forma de visualização de resultados, através de um grafo interativo, no qual é possível aceder à informação presente na tabela, utilizando-se o cursor, o que permite uma visualização mais intuitiva e clara dos resultados obtidos. Para além disso, também se pode verificar a existência do texto associado à receita, no texto é possível observar cada um dos ingredientes da receita destacados a vermelho.

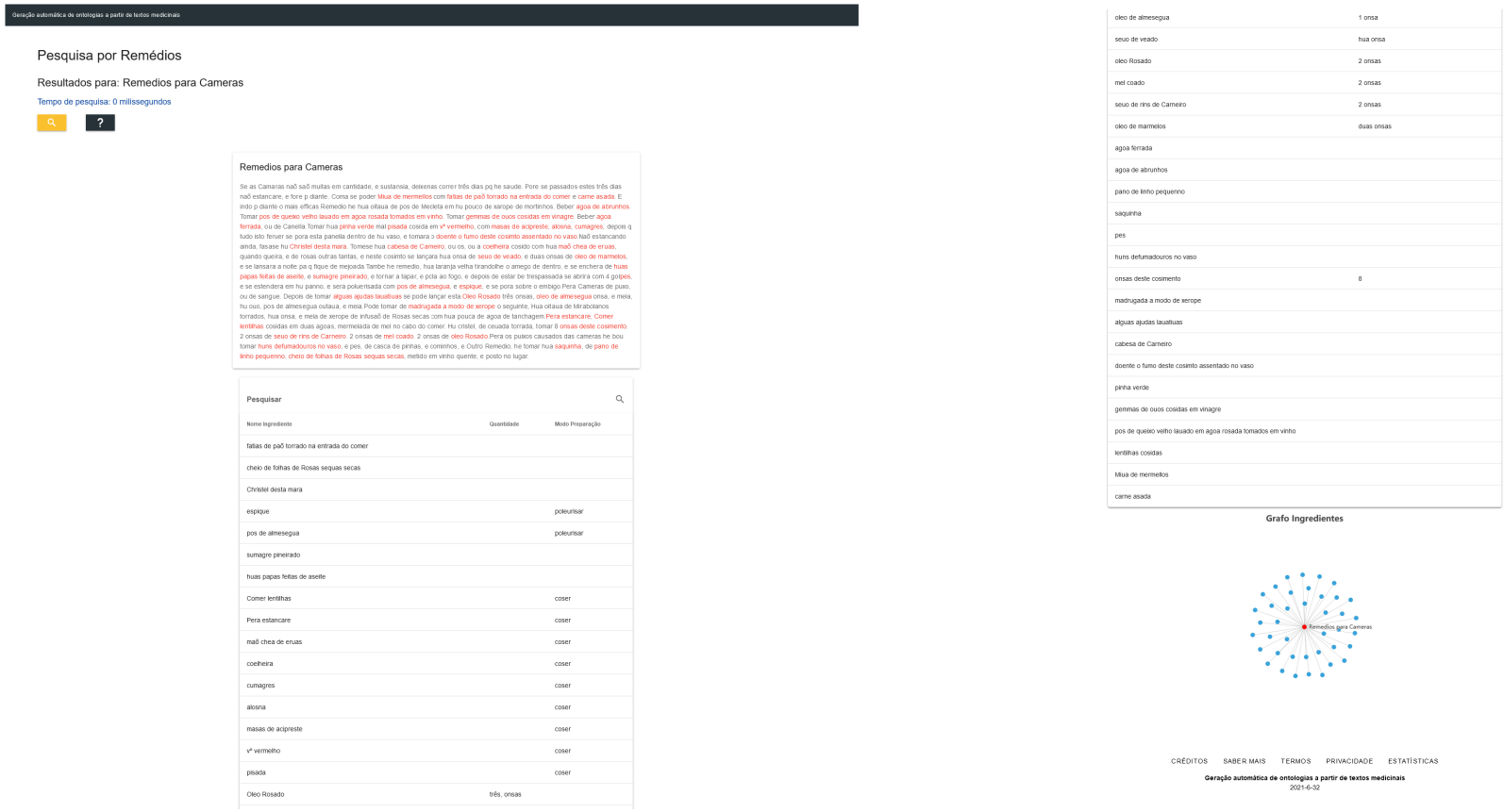


Figura 25: Resultados de uma pesquisa de uma receita de um medicamento.

Nos casos de pesquisas que envolvam mais do que um resultado, ou seja, que envolvam a procura de vários medicamentos (ou ingredientes), o sistema de exploração apresenta os resultados da forma como se pode observar na Figura 26. Também é possível verificar a existência de uma barra de navegação dos resultados, que permite percorrer cada um dos resultados de forma individual. Para além da barra de navegação, existe também um grafo geral no qual é possível navegar pelos resultados completos da pesquisa.

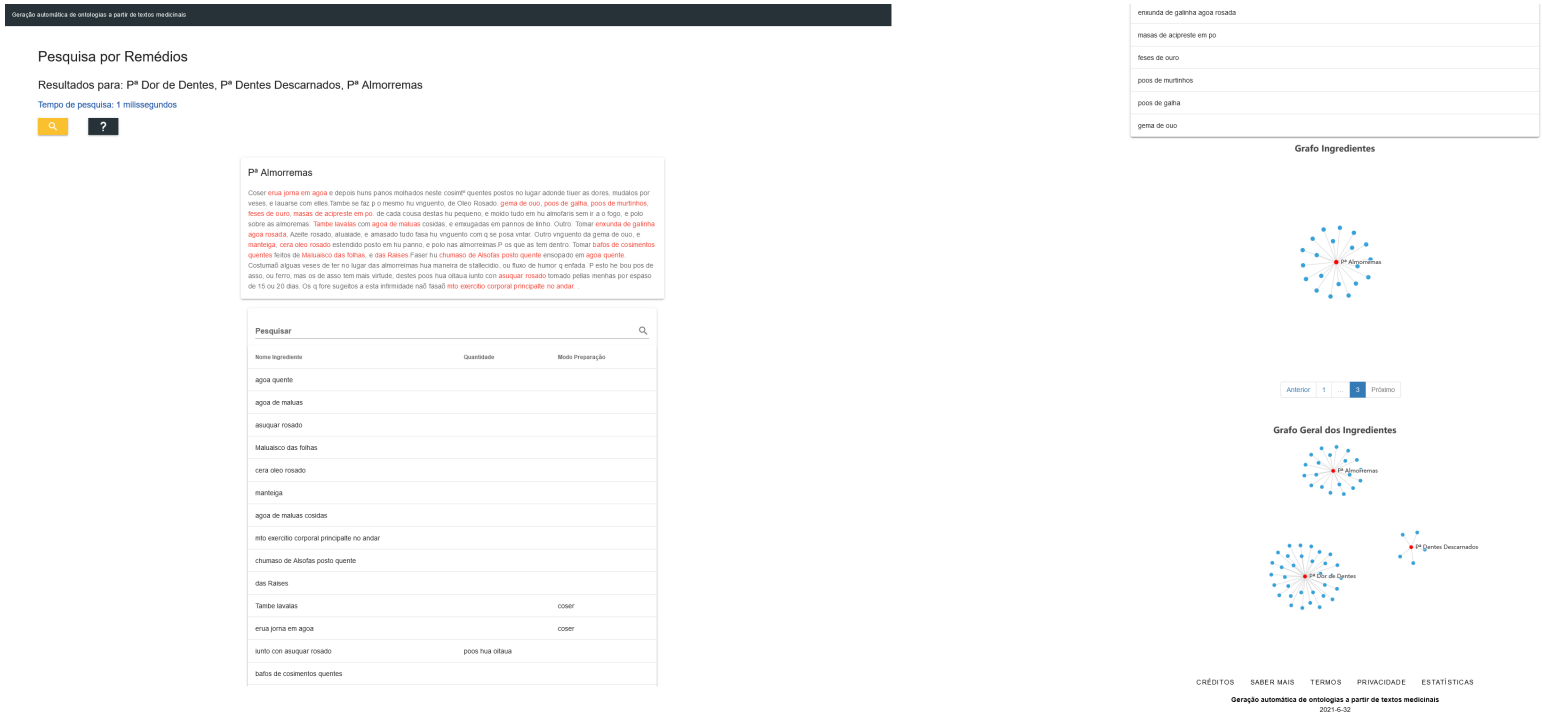


Figura 26: Página de resultados para pesquisas complexas.

No caso de se realizar uma pesquisa por ingredientes, a página de resultados obtidos mantém o mesmo aspeto, como se poderá observar na Figura 27, apenas variando um pouco no cabeçalho.

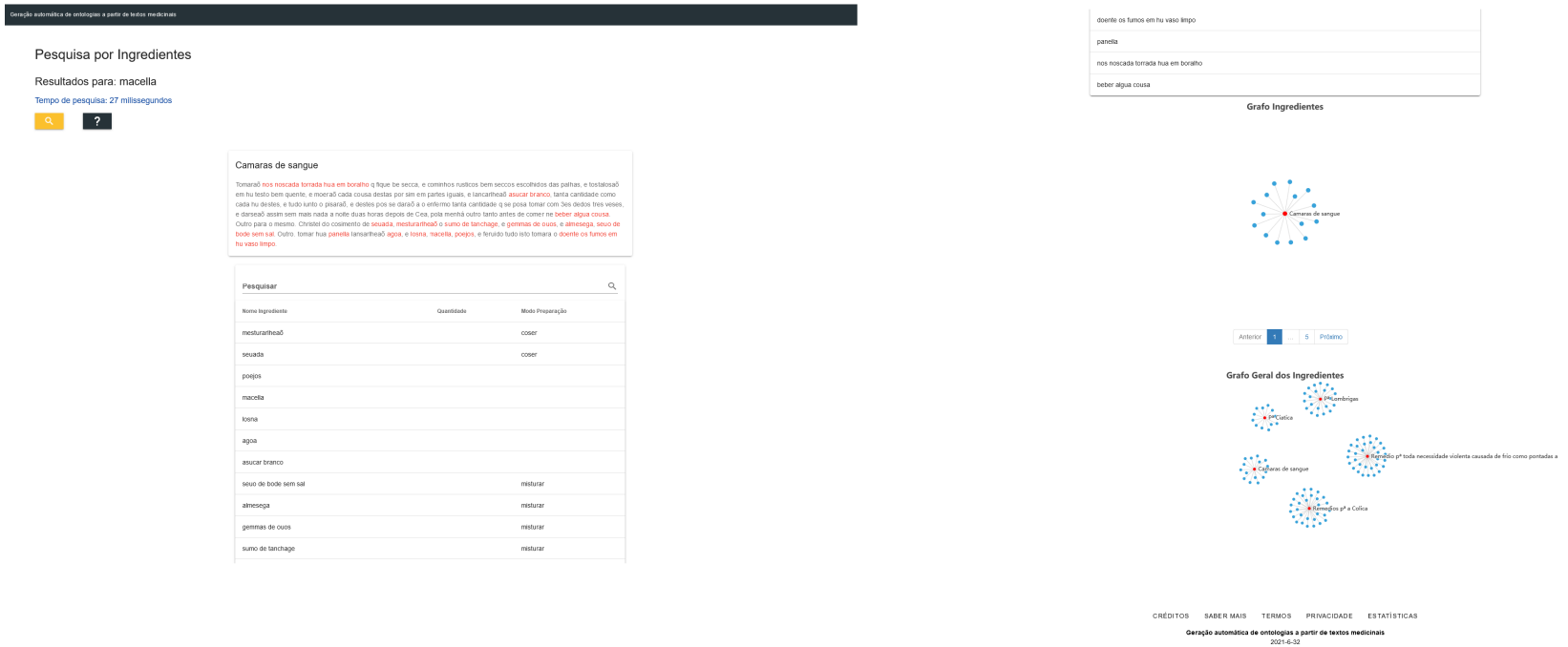


Figura 27: Página de resultados para pesquisa por ingredientes.

O seguinte grafo de ingredientes apresentado na Figura 28, foi extraído a partir da pesquisa por nome dos ingredientes efetuada para a Figura 27. É possível ver que existem cinco receitas de medicamentos ligadas entre si através de um ingrediente comum, o ingrediente “macella”. Usando o cursor do sistema, no grafo é possível verificar que o ingrediente anterior encontra-se na composição de cada uma das receitas presentes no grafo.

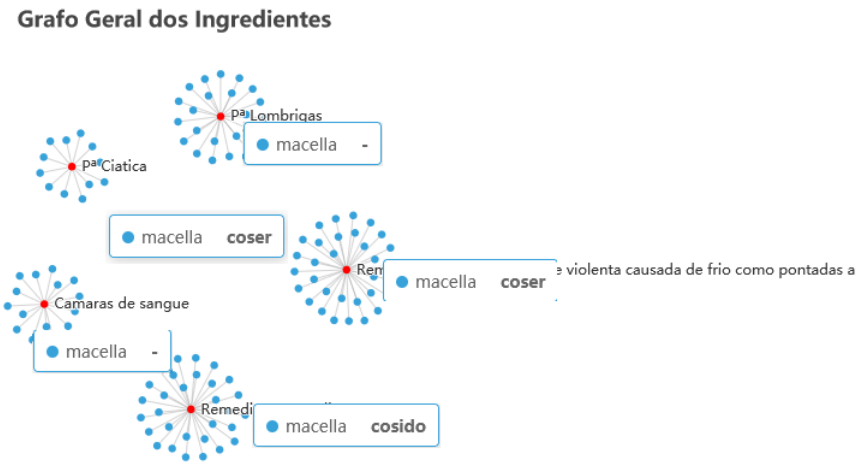


Figura 28: Grafo geral dos vários medicamentos anotado.

Por fim, e através do rodapé da página podemos aceder a um diálogo no qual é possível aceder ao conjunto bastante diverso de estatísticas, estatísticas essas que foram mencionadas na secção 5.4, relativas ao processo de extração, como se pode ver na Figura 29.


Figura 29: Diálogo das estatísticas.

Dentro das estatísticas, é possível consultar o número de receitas de medicamentos processadas, palavras e frases. Para além disso, é também possível ver a receita com o maior/menor número de palavras e linhas. Pode também ser consultada informação referente aos ingredientes, onde se pode ver o remédio que possui um maior/menor número de ingredientes bem como o ingrediente mais comum e incomum, assim como o número total de ingredientes. Na Figura 30 pode-se ver o conjunto de estatísticas mencionadas anteriormente.

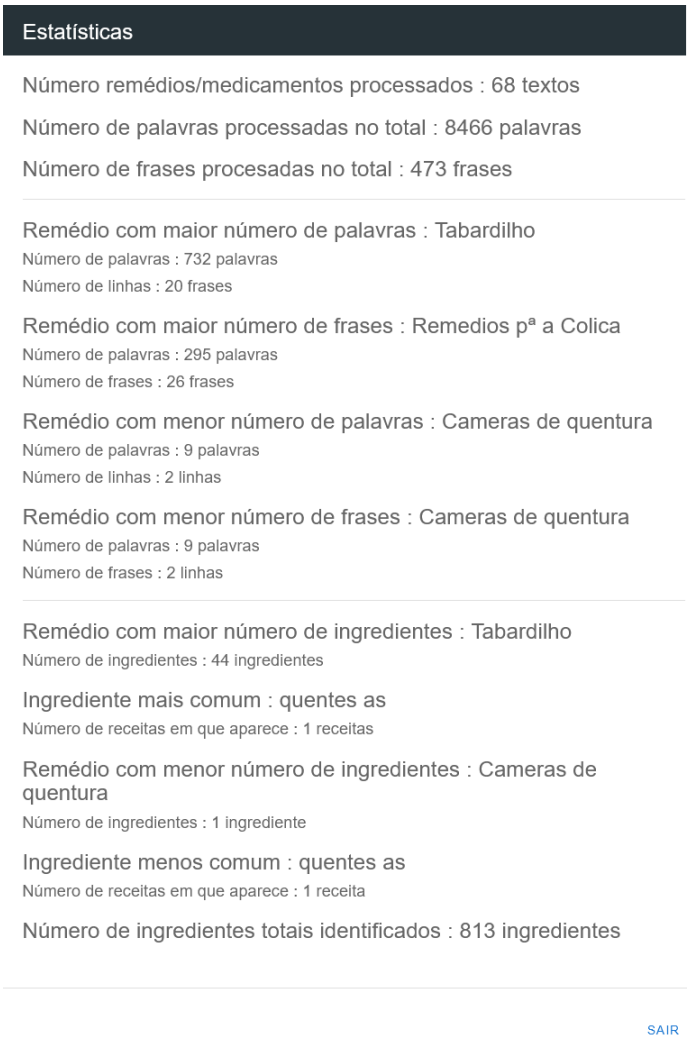


Figura 30: Diálogo completo das estatísticas.

# 6

# CONCLUSÕES E TRABALHO FUTURO

## 6.1 CONCLUSÕES

A extração semiautomática de ontologias é altamente vantajosa, especialmente no que concerne à redução de custos associados com a criação de ontologias para um domínio de conhecimento grande. Assim, com a utilização de algoritmos que permitem extrair ontologias de forma semiautomática é possível, principalmente, reduzir os custos temporais associados à modelação e captação do conhecimento para representar na forma de uma ontologia.

De forma a proceder com a aprendizagem semiautomática de ontologias, tipicamente é empregue um esquema composto por cinco fases. Antes de se aplicarem os mecanismos de extração dos elementos para compor a ontologia, é primeiro necessário pré processar o texto, através da eliminação das inconsistências semânticas e estruturais, até à etiquetação do texto através da aplicação de algoritmos como o *Part of Speech Tagging* e o *Named Entity Recognition*. De seguida, podem ser aplicados os algoritmos de extração de conceitos, como o *Seed Words*, *C/NC-Value* ou os padrões léxico-sintáticos. Logo após, os relacionamentos, que ligam os conceitos extraídos anteriormente, podem ser extraídos através de métodos como o *Recency Vector*. Para extrair os axiomas, podem ser utilizadas técnicas de programação lógica indutiva. A última fase corresponde à fase de avaliação da ontologia, onde, por exemplo, poderá ser elaborada uma avaliação *application* ou *data driven*.

Nesta dissertação, foi aplicado um processo de extração composto por três fases. Na primeira fase realizaram-se as operações de pré processamento dos textos de trabalho. Optou-se por simplificar esta fase, aplicando apenas metodologias simples de pré processamento, basicamente, com o intuito de não distorcer o significado dos textos, por exemplo, através da substituição de caracteres indesejáveis como os apóstrofes. De seguida, após uma análise de vários métodos e algoritmos, como os mencionados anteriormente, passíveis de serem aplicados com o intuito de extrair os principais elementos para compor uma ontologia, definimos e aplicámos um conjunto específico de padrões léxico-sintáticos. A utilização deste tipo de padrões permitiu implementar um sistema de fácil compreensão, que pode ser ajustado e refinado por diferentes pessoas. A aplicação dos padrões léxico-sintáticos permitiu fazer a identificação e a extração dos pares hiperónimo/hipónimo e, consequentemente, a hierarquia de conceitos. Após a extração dos *clusters* de elementos, seguiu-se o pós processamento e a representação e construção dos elementos obtidos na forma de uma ontologia. Para tal, optou-se por transformar os elementos extraídos para um sistema de grafos implementado em *Neo4J*. Com o processo de extração

concluído passámos à última fase de avaliação da ontologia. Em termos gerais, verificámos que com a aplicação deste processo, num conjunto de seis textos, obtivemos resultados positivos.

O conhecimento que o sistema permite recolher está relacionado com os ingredientes presentes nas receitas dos medicamentos, podendo-se, em certos casos, destacar também as quantidades e os modos de confeção associados. Desta forma, será possível destacar, por exemplo, as espécies botânicas utilizadas nas receitas ou os ingredientes de origem animal, bem como os minerais e entre outros tipos de ingredientes. Para além disso, consegue-se também evidenciar algumas das práticas medicinais, agrónomas e veterinárias associadas com a preparação dos medicamentos.

Assim, foi então possível criar uma rede semântica sob a forma de uma ontologia de medicamentos, capaz de responder, e sumarizar, um conjunto de receitas escritas em Português antigo, que remontam do século dezasseis ao século dezassete. Através da exploração dos resultados da extração é possível, de uma forma sustentada e efetiva, consultar a informação mais relevante e pertinente relacionada com um dado conjunto de medicamentos antigos. Este tipo de exploração facilita o processo de descoberta de novo conhecimento que esteja de alguma forma refletido nos textos de medicamentos.

No que concerne à exploração do conhecimento obtido através da ontologia gerada, optámos por utilizar uma plataforma específica, que foi desenvolvida recorrendo a tecnologias *Web*, de maneira a disponibilizar um conjunto de mecanismos de pesquisa que permitissem, de forma simples e amigável, aceder à informação contida nos textos dos medicamentos que foram processados. Nesse sentido, desenvolvemos uma plataforma *Web* para disponibilizar o acesso a esses mecanismos de pesquisa, que permitem explorar e evidenciar particularidades relacionadas com os medicamentos, através da informação presente na base de dados relativa à ontologia. De forma a criar um sistema efetivo e de alta disponibilidade, implementámos um sistema baseado em micro serviços, desenvolvido em *Node.js* com o apoio da *framework Express.js*. Neste sistema destacamos dois micro serviços, com alguma criticidade ao nível do *backend* do sistema, que são responsáveis por aceder aos dados presentes nas bases de dados utilizadas pelo sistema, bem como providenciar estes mesmos dados ao cliente. Para além do componente do *backend* do sistema destacamos, também, o componente de *frontend* que foi desenvolvido, sendo bastante simples e intuitivo, criado através de *Vue.js* juntamente com a *framework Vuetify*.

Portanto, conjugando os serviços da ferramenta de extração com os serviços do sistema de exploração de dados foi possível conceber uma ferramenta de análise que acolhe e realiza o processo de geração e análise de uma ontologia, desde a fase de extração dos conceitos mais pertinentes relacionados com os medicamentos, até à fase de visualização e análise dos dados extraídos. A partir dos conceitos extraídos, e com a ontologia já construída, é, então, possível fazer a exploração desses dados, contribuindo para a descoberta de conhecimento acerca dos medicamentos processados. Entre muitas outras coisas, essa exploração permitirá descobrir os ingredientes, suas quantidades e modos de confeção relacionados com os medicamentos processados, que remontam a séculos passados. Sem este sistema, não seria fácil fazer a aquisição desse conhecimento.

## 6.2 TRABALHO FUTURO

A ferramenta de extração produzida, juntamente com o sistema de exploração de dados, é vantajosa no que concerne à aquisição, e representação, rápida do conhecimento embebido nas receitas de medicamentos utilizadas. Desta forma, rapidamente se poderá proceder a uma análise da informação presente em cada uma das receitas, por meio da plataforma de exploração, onde se pode evidenciar, por exemplo, relações aparentemente escondidas entre medicamentos, através da utilização de ingredientes comuns às receitas.

Tendo isso em consideração, achamos que temos espaço para o desenvolvimento e aplicação de algumas medidas para melhorar, em particular, o desempenho da ferramenta de extração que desenvolvemos. Assim, de forma a reduzir o ruído, que é representativo dos conceitos incorretamente classificados, associado ao processo de extração, poderia ser aplicado um filtro relativamente a extração dos conceitos. No filtro poderiam ser aplicadas técnicas como o *LSA* (*Latent Semantic Analysis*)Cederberg and Widdows (2003), ou até mesmo através de um simples cálculo de frequência de aparecimento do conceito nos textos, de forma a garantir que apenas os conceitos, ingredientes, mais relevantes sejam mantidos na ontologia final. Neste cálculo poderia ser estabelecido um patamar mínimo de frequência, onde apenas os conceitos com uma frequência superior ao patamar seriam mantidos nos resultados finais. Desta forma, com o uso de um filtro como este, seria possível reduzir o ruído produzido pelo processo de extração de ingredientes por parte da ferramenta desenvolvida, e consequentemente seriam obtidos resultados mais precisos, consistentes e de melhor qualidade.

Uma outra melhoria em termos da qualidade dos resultados seria reduzir o grau de automatização do processo de extração da ontologia de medicamentos. Ou seja, ao invés do processo de extração ser automatizado, este poderia ser convertido num processo supervisionado, em que, através da intervenção de um especialista, seria possível aceitar, remover ou até mesmo ajustar os termos extraídos em tempo real. A apresentação dos termos extraídos, e a sua posterior validação e alteração, poderia ser efetuada, por exemplo, no final do processamento de cada medicamento. De forma a automatizar um pouco mais este processo, também poderia ser estabelecida uma *cache* na qual estivessem contidos todos os termos aceites, assim como os termos que foram removidos pelo especialista. Depois, esta *cache* seria acedida para que se pudesse verificar se os termos sugeridos não se encontram dentro da *cache*, uma vez que representam conhecimento que já foi processado. Assim, poderíamos garantir, dentro dos padrões léxico-sintáticos definidos, resultados mais corretos e precisos, contribuindo também para a garantia da qualidade da própria ontologia produzida.

Por fim, pensamos que a documentação produzida para a ferramenta de extração poderia ter sido mais geral e rigorosa. Nesta altura, pensamos que seria uma boa ideia, a curto prazo, utilizar uma ferramenta para a ajuda à produção de documentação que permitisse esclarecer e partilhar de forma simples todos os aspetos do processo de extração, em particular do esquema das tarefas de extração que foi implementado.

# BIBLIOGRAFIA


Abeer Al-Arfaj and Abdulmalik Al-Salman. Ontology Construction from Text: Challenges and Trends. *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, 6(2):15–26, 2015.

Ayesha Ameen, Khaleel Ur Rahman Khan, and B. Padmaja Rani. Creation of Ontology in Education Domain. In *2012 IEEE Fourth International Conference on Technology for Education*, pages 237–238. IEEE, jul 2012. ISBN 978-1-4673-2173-0. doi: 10.1109/T4E.2012.50. URL http://ieeexplore.ieee.org/document/6305981/.

Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. Automated Extraction and Clustering of Requirements Glossary Terms. *IEEE Transactions on Software Engineering*, 43(10):918–945, oct 2017. ISSN 0098-5589. doi: 10.1109/TSE.2016.2635134. URL http://ieeexplore.ieee.org/document/7765062/.

Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. A survey of ontology learning techniques and applications. *Database*, 2018(2018), jan 2018. ISSN 1758-0463. doi: 10.1093/database/bay101. URL https://academic.oup.com/database/article/doi/10.1093/database/bay101/5116160.

Ali Ayadi, Ahmed Samet, François de Bertrand de Beuvron, and Cecilia Zanni-Merk. Ontology population with deep learning-based NLP: a case study on the Biomolecular Network Ontology. *Procedia Computer Science*, 159: 572–581, 2019. ISSN 18770509. doi: 10.1016/j.procs.2019.09.212. URL https://linkinghub.elsevier.com/retrieve/pii/S1877050919313961.

Anabela Barros. *As receitas de cozinha de um frade português do século XVI, com Prefácio de Raquel Seiça e Colaboração de Joana Veloso e Micaela Aguiar*. Coimbra: Imprensa da Universidade de Coimbra, jun 2013. ISBN 978-989-26-0613-2. URL https://www.researchgate.net/publication/323245332_As_receitas_de_cozinha_de_um_frade_portugues_do_seculo_XVI.

Anabela Barros. Remédios vários e receitas aprovadas (ms. 142 do Arquivo Distrital de Braga): entre a História da Medicina e a História da Língua, a Ecdótica. In *As Humanidades e as Ciências: Disjunções e Confluências. XV Colóquio de Outono*, pages 25–57. 2014. URL https://www.researchgate.net/publication/325256242_Remedios_varios_e_receitas_aprovadas_ms_142_do_Arquivo_Distrital_de_Braga_entre_a_Historia_da_Medicina_e_a_Historia_da_Lingua_a_Ecdotica.

Anabela Barros. *Remédios vários e receitas aprovadas. Segredos vários - Edição semidiplomática e edição interpretativa do Caderno II do manuscrito 142 do Arquivo Distrital de Braga*. Imprensa da Universidade de Coimbra / Coimbra University PressCoimbra/Fundação Calouste Gulbenkian, dec 2016. ISBN 978-989-26-1281-2. doi: 10.14195/978-989-26-1282-9.

Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009. ISBN 978-0-596-51649-9.

Willem Nico Borst. *Construction of engineering ontologies for knowledge sharing and reuse*. Twente, sep 1997.

Janez Brank, Marko Grobelnik, and D Mladenic. A survey of ontology evaluation techniques. In *In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, Ljubljana, jan 2005. Jozef Stefan Institute.

Christopher Brewster. Ontology Learning from Text: Methods, Evaluation and Applications Paul Buitelaar, Philipp Cimiano, and Bernado Magnini (editors) (DFKI Saarbrücken, University of Karlsruhe, and ITC-irst), Amsterdam: IOS Press (Frontiers in artificial intelligence and appl. *Computational Linguistics*, 32(4):569–572, dec 2006. ISSN 0891-2017. doi: 10.1162/coli.2006.32.4.569. URL https://direct.mit.edu/coli/article/32/4/569-572/1926.

Christopher Brewster, Harith Alani, Srinandan Dasmahapatra, and Yorick Wilks. Data driven ontology evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation, LREC 2004*, pages 641–644. European Language Resources Association (ELRA), 2004. ISBN 2951740816.

Scott Cederberg and Dominic Widdows. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 -*, volume 4, pages 111–118, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119191. URL http://portal.acm.org/citation.cfm?doid=1119176.1119191.

B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, 14(1):20–26, jan 1999. ISSN 1094-7167. doi: 10.1109/5254.747902. URL http://ieeexplore.ieee.org/document/747902/.

Anjali Chauhan. A Review on Various Aspects of MongoDb Databases. *International Journal of Engineering Research Technology*, 8(5), may 2019. ISSN 2278-0181. URL https://www.stackchief.com/search/mongodb.

Philipp Cimiano and Johanna Völker. Text2Onto. In *Lecture Notes in Computer Science*, volume 3513, pages 227–238. Springer Verlag, 2005. doi: 10.1007/11428817_21. URL http://link.springer.com/10.1007/11428817_21.

Philipp Cimiano, Alexander Mädche, Steffen Staab, and Johanna Völker. Ontology Learning. In *Handbook on Ontologies*, pages 245–267. Springer Berlin Heidelberg, Berlin, Heidelberg, may 2009. doi: 10.1007/978-3-540-92673-3_11.

Matteo Cristani and Roberta Cuel. A Survey on Ontology Creation Methodologies. *International Journal on Semantic Web and Information Systems*, 1(2):49–69, apr 2005. ISSN 1552-6283. doi: 10.4018/jswis.2005040103. URL http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jswis.2005040103.

Mirna El Ghosh, Hala Naja, Habib Abdulrab, and Mohamad Khalil. Ontology Learning Process as a Bottom-up Strategy for Building Domain-specific Ontology from Legal Texts. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, pages 473–480. SCITEPRESS - Science and Technology Publications, jan 2017. ISBN 978-989-758-219-6. doi: 10.5220/0006188004730480. URL http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006188004730480.

Soumaya El Kadiri, Walter Terkaj, Esmond Neil Urwin, Claire Palmer, Dimitris Kiritsis, and Robert Young. Ontology in Engineering Applications. In *Lecture Notes in Business Information Processing*, volume 225, pages 126–137. Springer Verlag, 2015. ISBN 9783319215440. doi: 10.1007/978-3-319-21545-7_11. URL http://link.springer.com/10.1007/978-3-319-21545-7_11.

Mariano Fernandez, A. Gómez-Pérez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Madrid, mar 1997. Universidad Politécnica de Madrid.

G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. ISSN 0018-9219. doi: 10.1109/PROC.1973.9030. URL http://ieeexplore.ieee.org/document/1450960/.

Blaz Fortuna, Marko Grobelnik, and Dunja Mladenic. OntoGen: Semi-automatic Ontology Editor. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4558, pages 309–318. Springer Verlag, 2007. ISBN 9783540733539. doi: 10.1007/978-3-540-73354-6_34. URL http://link.springer.com/10.1007/978-3-540-73354-6_34.

Hermine Fotzo and Patrick Gallinari. Learning « Generalization/Specialization » Relations between Concepts: Application for Automatically Building Thematic Document Hierarchies. In *RIAO 2004*, pages 143–155, Paris, jan 2004. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE. ISBN 905450096.

Alvaro L Fraga, Marcela Vegetti, and Horacio P Leone. Semi-Automated Ontology Generation Process from Industrial Product Data Standards. In *III Simposio Argentino de Ontologías y sus Aplicaciones (SAOA) - JAIIO 46 (Córdoba, 2017)*, pages 53–66, aug 2017.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1433–1445, New York, NY, USA, may 2018. ACM. ISBN 9781450347037. doi: 10.1145/3183713.3190657. URL https://dl.acm.org/doi/10.1145/3183713.3190657.

Katerina T. Frantzi, Sophia Ananiadou, and Junichi Tsujii. The C-value/NC-value Method of Automatic Recognition for Multi-word Terms. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1513, pages 585–604. aug 1998. ISBN 9783540651017. doi: 10.1007/3-540-49653-X_35. URL http://link.springer.com/10.1007/3-540-49653-X_35.

Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5-6):907–928, nov 1995. ISSN 10715819. doi: 10.1006/ijhc.1995.1081. URL https://linkinghub.elsevier.com/retrieve/pii/S1071581985710816.

Anselmo Guedes, Fernanda Baião, and Kate Revoredo. On the identification and representation of ontology correspondence antipatterns. In *CEUR Workshop Proceedings*, volume 1248, jan 2014.

Cornelia Gyorodi, Robert Gyorodi, George Pecherle, and Andrada Olah. A comparative study: MongoDB vs. MySQL. In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, pages 1–6. IEEE, jun 2015. ISBN 978-1-4799-7649-2. doi: 10.1109/EMES.2015.7158433. URL http://ieeexplore.ieee.org/document/7158433/.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics -*, volume 2, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics. doi: 10.3115/992133.992154. URL http://portal.acm.org/citation.cfm?doid=992133.992154.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy, 2020.

Nilesh Jain, Ashok Bhansali, and Deepak Mehta. Angularjs: A modern mvc framework in javascript. *Journal of Global Research in Computer Science*, 5(12):17–23, 2014.

Krzysztof Janowicz, Patrick Maué, Marc Wilkes, Sven Schade, Franca Scherer, Matthias Braun, Sören Dupke, and Werner Kuhn. Similarity as a quality indicator in ontology engineering. In *Frontiers in Artificial Intelligence and Applications*, volume 183, pages 92–105. IOS Press, 2008. ISBN 9781586039233. doi: 10.3233/978-1-58603-923-3-92.

Azanzi Jiomekong, Gaoussou Camara, and Maurice Tchuente. Extracting ontological knowledge from Java source code using Hidden Markov Models. *Open Computer Science*, 9(1):181–199, aug 2019. ISSN 2299-1093. doi: 10.1515/comp-2019-0013. URL https://www.degruyter.com/document/doi/10.1515/comp-2019-0013/html.

Dean Jones, Trevor Bench-Capon, and Pepijn Visser. Methodologies for Ontology Development. 1998.

Neha Kaushik and Niladri Chatterjee. Automatic relationship extraction from agricultural text for ontology construction. *Information Processing in Agriculture*, 5(1):60–73, mar 2018. ISSN 22143173. doi: 10.1016/j.inpa.2017.11.003. URL https://linkinghub.elsevier.com/retrieve/pii/S2214317317300227.

H.M. Kim, M.S. Fox, and Michael Gruninger. An ontology of quality for enterprise modelling. In *Proceedings 4th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '95)*, pages 105–116. IEEE Comput. Soc. Press, 1995. ISBN 0-8186-7019-3. doi: 10.1109/ENABL.1995.484554. URL http://ieeexplore.ieee.org/document/484554/.

Holger Knublauch. Travel Ontology, 2016. URL https://protegewiki.stanford.edu/wiki/Importing_Ontologies_in_P41.

Deepika Kumawat and Vinesh Jain. POS Tagging Approaches: A Comparison. *International Journal of Computer Applications*, 118(6):32–38, may 2015. ISSN 09758887. doi: 10.5120/20752-3148. URL http://research.ijcaonline.org/volume118/number6/pxc3903148.pdf.

John Leider and Heather Leider. Vuetify - A Material Design Framework for Vue.js, 2016. URL https://vuetifyjs.com/en/.

Renars Liepins, Mikus Grasmanis, and Uldis Bojars. OWLGrEd ontology visualizer. In *CEUR Workshop Proceedings*, volume 1268, pages 37–42. CEUR-WS.org, 2014. doi: 10.5555/2878379.2878386.

Rinaldo Lima, Bernard Espinasse, and Fred Freitas. A logic-based relational learning approach to relation extraction: The OntoILPER system. *Engineering Applications of Artificial Intelligence*, 78:142–157, feb 2019. ISSN 09521976. doi: 10.1016/j.engappai.2018.11.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0952197618302276.

Pablo Neves Machado and Vera Lúcia Strube de Lima. Extração de relações hiponímicas em um corpus de língua portuguesa. *REVISTA DE ESTUDOS DA LINGUAGEM*, 23(3):599–640, dec 2015. ISSN 2237-2083. doi: 10.17851/2237-2083.23.3.599-640. URL http://www.periodicos.letras.ufmg.br/index.php/relin/article/view/8893.

Alexander Maedche and Steffen Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, mar 2001. ISSN 15411672. doi: 10.1109/5254.920602. URL http://ieeexplore.ieee.org/document/920602/.

Pratik Sharad Maratkar and Pratibha Adkar. React JS - An Emerging Frontend JavaScript Library. *IRE Journals*, 4(12):99–102, jun 2021. ISSN 2456-8880. URL https://www.irejournals.com/.

Mark A. Musen. The protégé project. *AI Matters*, 1(4):4–12, jun 2015. ISSN 2372-3483. doi: 10.1145/2757001.2757003. URL https://dl.acm.org/doi/10.1145/2757001.2757003.

NeOn Technologies Foundation. NeOn Wiki, jul 2014.

Natalya Noy and Deborah Mcguinness. Ontology Development 101: A Guide to Creating Your First Ontology. *Knowledge Systems Laboratory*, 32, mar 2001.

Annika Öhgren. Ontology Development and Evolution: Selected Approaches for Small-Scale Application Contexts. 2004. ISSN 1404-0018.

Assuncion Pérez, Mariano López, and Oscar Corcho. METHONTOLOGY. In *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web*, volume 1, pages 125–142. 1 edition, 2004. ISBN 1849968845.

Peter Pšenák and Matúš Tibenský. The usage of Vue JS framework for web application creation. *Mesterséges intelligencia*, 2(2):61–72, 2020. ISSN 26769611. doi: 10.35406/MI.2020.2.61.

Cecilia Reyes-Peña and Mireya Tovar-Vidal. Ontology: Components and Evaluation, a Review. *Research in Computing Science*, 148(3):257–265, dec 2019. ISSN 1870-4069. doi: 10.13053/rcs-148-3-21. URL http://rcs.cic.ipn.mx/2019_148_3/Ontology_ComponentsandEvaluation_aReview.pdf.

Christophe Roche. Ontology: A Survey. *IFAC Proceedings Volumes*, 36(22):187–192, sep 2003. ISSN 14746670. doi: 10.1016/S1474-6670(17)37715-7. URL https://linkinghub.elsevier.com/retrieve/pii/S1474667017377157.

Félix Melchor Santos López and Eulogio Guillermo Santos De La Cruz. Literature review about Neo4j graph database as a feasible alternative for replacing RDBMS. *Industrial Data*, 18(2):135–139, dec 2015. ISSN 1810-9993. doi: 10.15381/idata.v18i2.12106. URL http://revistasinvestigacion.unmsm.edu.pe/index.php/idata/article/view/12106.

Hezbullah Shah and Tariq Soomro. Node.js Challenges in Implementation. *Global Journal of Computer Science and Technology*, 17:72 –83, may 2017. ISSN 0975-4172.

Maria Josefa Somodevilla, Darnes Vilariño Ayala, and Ivo Pineda. An Overview on Ontology Learning Tasks. *Computación y Sistemas*, 22(1):137–146, mar 2018. ISSN 2007-9737. doi: 10.13053/cys-22-1-2790. URL http://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/2790.

Sanju Tiwari and Sarika Jain. Automatic Ontology Acquisition and Learning. *International Journal of Research in Engineering and Technology*, 03(26):38–43, nov 2014. ISSN 23217308. doi: 10.15623/ijret.2014.0326008.

Mike Uschold and Michael Gruninger. Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, 11(2):93–136, jun 1996. ISSN 0269-8889. doi: 10.1017/S0269888900007797. URL https://www.cambridge.org/core/product/identifier/S0269888900007797/type/journal_article.

Markos Viggiato, Ricardo Terra, Henrique Rocha, Marco Tulio Valente, and Eduardo Figueiredo. Microservices in Practice: A Survey Study. aug 2018. URL http://arxiv.org/abs/1808.04836.

S. Vijayarani, J. Ilamathi, and S. Nithya. Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science  Communication Networks*, 5(1):7–16, 2015. ISSN 2249-5789.

Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text. *ACM Computing Surveys*, 44(4):1–36, aug 2012. ISSN 0360-0300. doi: 10.1145/2333112.2333115. URL https://dl.acm.org/doi/10.1145/2333112.2333115.