UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA

COLEGIADO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO

Rayane de Souza do Nascimento

Busca de Padrões em um Repositório de Padrões Ontológicos Orientados a Objetivos

Projeto de Graduação apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação.

Orientadora: Monalessa Perini Barcellos
Coorientador: Cássio Chaves Reginato

Vitória

2022

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA

COLEGIADO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO

Rayane de Souza do Nascimento

Busca de Padrões em um Repositório de Padrões Ontológicos Orientados a Objetivos

COMISSÃO EXAMINADORA

_____

Prof. Monalessa Perini Barcellos, D. Sc.

_____

Prof. Vítor Estêvão Silva Souza, D. Sc.

_____

Prof. Camila Zacché de Aguiar, D. Sc.

Vitória, 04 de agosto de 2022

"All you really need to know for the moment is that the universe is a lot more complicated than you might think, even if you start from a position of thinking it's pretty damn complicated in the first place"

(Douglas Adams)

AGRADECIMENTOS

Agradeço à minha mãe, Léia, e ao meu pai, Renato, por terem criado em mim a curiosidade, inquietude e desejo de entender o mundo, e por sempre me ensinarem que a dedicação ao estudo era o único caminho possível.

À minha tia Laice, por me ensinar o que é o amor verdadeiro, e que sem ele eu jamais chegaria até aqui.

Aos meus amigos, Laisa, Rísia e Rafael, por serem uma companhia perfeita tanto nos momentos de estudo quanto de diversão, e por nunca terem me deixado desacreditar de mim.

À minha irmã Renata, por ser minha maior admiradora e incentivadora e ao meu cunhado Gabriel, por toda a ajuda no desenvolvimento deste trabalho e todo o companheirismo na vida.

Aos bons professores, vocês fizeram toda a diferença na minha caminhada. Para além de transmitir conhecimento, me fizeram acreditar no poder da educação e me motivaram a continuar mesmo quando eu pensei em desistir, em especial, à minha orientadora Monalessa e meu coorientador Cássio.

Aos amigos da faculdade, onde inúmeras vezes trocamos conhecimentos, sorrisos, angústias, e muita colaboração. Saibam que sem vocês isso não seria possível.

Por fim, ao meu amado Dener, por seu apoio incondicional em uma parte crucial dessa trajetória, e por ser sempre paciente, acolhedor e motivador.

RESUMO

Ontologias têm sido aplicadas em diversas áreas como Engenharia de *Software*, Inteligência Artificial, *Web* Semântica entre outras, possibilitando a criação de modelos conceituais corretos, claros e que possam ser compartilhados e reutilizados. O desenvolvimento de ontologias é uma tarefa difícil mesmo para um especialista. Uma das formas de simplificar esse processo é reusar ontologias existentes no desenvolvimento de uma nova ontologia.

A reutilização de ontologias contribui para acelerar o processo de desenvolvimento, utilizando menos tempo e recursos, além de estimular o uso de boas práticas, o que contribui para a qualidade da ontologia resultante. Para favorecer o reúso, o uso de padrões (*patterns*) tem se mostrado uma abordagem promissora na Engenharia de Ontologias. Um padrão pode ser definido como uma solução bem-sucedida para um problema recorrente.

No entanto, construir uma ontologia a partir do reúso de padrões ontológicos também não é uma atividade simples. Um dos desafios é a obscuridade da lógica do design (i.e., *design rationale*) das ontologias disponíveis. Uma abordagem que pode facilitar o entendimento da lógica por trás de uma decisão é a Engenharia de Requisitos Orientada a Objetivos (*Goal-Oriented Requirements Engineering* - GORE), que advoga que ao utilizar objetivos para derivar requisitos, as motivações para os requisitos tornam-se mais claras. Nesse sentido, no Núcleo de Estudos em Modelagem Conceitual e Ontologias (NEMO), foi proposto GO-FOR (*Goal-Oriented Framework for Ontology Reuse*), que utiliza objetivos como elemento central para promover reúso de *Goal-Oriented Ontology Patterns* (GOOPs), que são fragmentos de ontologias associados a objetivos. O *framework* é composto por uma arquitetura conceitual, um processo e uma ferramenta chamada *GoopHub*, que permite o armazenamento e seleção de GOOPs. Neste trabalho foi realizado um aprimoramento no mecanismo de seleção de GOOPs em *GoopHub*, através da implantação de uma estrutura gramatical que padroniza a terminologia usada para nomear os GOOPs, e da aplicação de técnicas de busca automatizada, utilizando processamento de linguagem natural.

Palavras-chave: Ontologia, Reúso, Modelagem de Objetivos, GORE, *Pattern*.

LISTA DE FIGURAS

SUMÁRIO

Capítulo 1

Introdução

Este capítulo apresenta uma breve introdução ao tema do trabalho, seus objetivos, histórico do desenvolvimento e a organização deste documento.

1.1 Introdução

Ontologia pode ser definida como uma descrição formal de um domínio. Tal descrição pode ser compartilhada entre diferentes aplicações (NOY, 2004). Para Guarino (1997), uma ontologia é um artefato, constituído por um vocabulário próprio usado para descrever uma determinada realidade, contendo, ainda, uma série de inferências nítidas acerca do significado das palavras deste vocabulário. Ontologias têm sido aplicadas em diversas áreas como Engenharia de *Software*, Inteligência Artificial e *Web* Semântica, entre outras para tratar problemas relacionados à semântica.

Mendonça e Soares (2017) afirmam que o processo de desenvolvimento de uma ontologia é complexo, pois envolve a criação de modelos semânticos ou descrições simplificadas da realidade de um dado domínio e exige dos desenvolvedores conhecimentos técnicos em modelagem conceitual, em lógica formal e em alguns aspectos filosóficos. Embora hoje em dia os engenheiros de ontologia sejam apoiados por uma ampla gama de métodos e ferramentas de engenharia de ontologias, o desenvolvimento de ontologias ainda é uma tarefa difícil mesmo para um especialista (POVEDA-VILLALÓN; SUÁREZ-FIGUEROA; GÓMEZ-PÉREZ, 2010). Uma das maneiras de simplificar esse processo é reutilizar ontologias existentes na construção de uma nova ontologia.

No entanto, ainda é difícil construir uma ontologia através do reúso de ontologias existentes. Uma das principais dificuldades está em encontrar ontologias realmente adequadas que se alinhem com o propósito que se quer cumprir. Presutti et al. (2009) afirmam que um dos desafios da reutilização de ontologias é a obscuridade da lógica do design (i.e., *design rationale*) das ontologias disponíveis. A lógica do design refere-se às razões das decisões tomadas durante um processo de design (JARCZYK; LÖFFLER; SHIPMAN, 1992). Desconhecer a lógica por trás do desenvolvimento dificulta a seleção das ontologias a serem reutilizadas e a compreensão delas, o que é crucial para integrá-las adequadamente.

Uma abordagem que pode auxiliar no entendimento da lógica por trás de uma decisão é a Engenharia de Requisitos Orientada a Objetivos (*Goal-Oriented Requirements Engineering* - GORE), que advoga que o uso de objetivos para derivar requisitos contribui para o entendimento das motivações para os requisitos tornam-se mais claras. GORE pode ser utilizada no âmbito da Engenharia de Ontologias para auxiliar no levantamento dos requisitos que a ontologia deve atender, bem como no entendimento das motivações para tais requisitos e para a ontologia em si. Nesse sentido, em Reginato et al. (2019), é proposto GO-FOR (*Goal-Oriented Framework for Ontology Reuse*). Os autores discutem como o uso de GORE pode ser útil no reúso de ontologias e propõem uma arquitetura na qual objetivos são o elemento central para promover reúso. Os elementos básicos de GO-FOR são padrões ontológicos orientados a objetivos (*Goal-Oriented Ontology Patterns* - GOOPs). Um GOOP consiste em um fragmento de ontologia envolvido por um objetivo. Em um GOOP, o objetivo estabelece o escopo abordado pelo fragmento de ontologia. Assim, os GOOPs podem ser reutilizados com base no objetivo a que se relacionam. Os GOOPs são armazenados em um repositório de padrões ontológicos orientados a objetivos (*Goal-Oriented Ontology Patterns Repository* - GOOPR). Em GOOPR, os GOOPs se relacionam de acordo com as relações entre seus objetivos (REGINATO et al., 2019). Para nomear os GOOPs, GO-FOR orienta que seja utilizado um verbo no infinitivo seguido de um substantivo ou uma frase substantiva (e.g., caracterizar localização). GO-FOR possui uma ferramenta de apoio chamada *GoopHub* (NOGUEIRA, 2019), que permite aos engenheiros de ontologia registrar e recuperar GOOPs em seu repositório.

Um aspecto importante na recuperação de padrões ontológicos é a terminologia adotada para dar nome aos dos padrões. Nos repositórios de ontologias disponíveis na internet, encontram-se muitas ontologias que tratam dos mesmos conceitos, porém com visões e propósitos distintos. A nomeação arbitrária dos padrões ontológicos dificulta a seleção de uma ontologia adequada para reúso em um contexto particular (GUIMARÃES, 2015). No contexto da organização do conhecimento, a representação da informação possui mecanismos para representar os objetos a partir de um conjunto de termos que caracterizem o seu conteúdo e permitam, assim, a sua busca. Nesse sentido, uma das preocupações é a padronização da terminologia utilizada para se encontrar e se classificar a informação. Essa padronização visa facilitar o processo de representação de informação, evitando redundâncias e inconsistências no conjunto de termos utilizados (GUIMARÃES, 2015), possibilitando nomear uma ontologia ou fragmento de ontologia de tal forma que seu nome seja o mais descritivo possível acerca do que a ontologia representa.

Considerando a importância de nomear adequadamente ontologias para favorecer o reúso, este projeto de graduação aprimora a seleção de GOOPs em *GoopHub* através (i) da implantação de uma estrutura gramatical para padronizar a terminologia usada para nomear padrões ontológicos, e (ii) da aplicação de técnicas de busca automatizada, utilizando processamento de linguagem natural para fornecer contexto à busca. Com esse aprimoramento, espera-se contribuir para o reúso de padrões ontológicos e apoiar o desenvolvimento de ontologias de qualidade, uma vez que encontrar ontologias adequadas é um ponto central no reúso de ontologias.

1.2 Objetivos

O objetivo geral deste projeto de graduação é *aperfeiçoar a busca por padrões ontológicos orientados a objetivos (GOOPs) em GoopHub através da implementação da estrutura gramatical proposta em (REGINATO et al., 2022) para padronizar sua nomeação e de estratégias para realizar busca por proximidade semântica*. Dessa forma, o objetivo geral do trabalho pode ser decomposto nos seguintes objetivos específicos:

(i) Implementar em *GoopHub* a estrutura gramatical proposta em (REGINATO et al., 2022).

(ii) Determinar as técnicas de busca automatizada de GOOPs a serem utilizadas.

(iii) Implementar a solução de busca proposta no *GoopHub*.

(iv) Realizar testes para verificar os resultados das melhorias realizadas no *GoopHub*.

1.3 Método de Desenvolvimento do Trabalho

Para o desenvolvimento deste trabalho, foram realizadas as seguintes etapas:

i) Revisão Bibliográfica: Consistiu na revisão de literatura acerca de ontologias, padrões ontológicos, GO-FOR e técnicas de busca automatizada, a fim de adquirir conhecimento teórico para a realização do trabalho.

ii) Estudo das Tecnologias: Consistiu no estudo das tecnologias utilizadas, destacando-se: linguagens de programação Java, JavaScript e Python, *framework* para desenvolvimento *web* Spring Boot e banco de dados Stardog.

iii) Definição das estratégias: Consistiu na definição das estratégias de busca a serem avaliadas para uso e, então, utilizadas.

iv) Implementação: Consistiu na implementação, no *GoopHub*, da estrutura gramatical proposta e das estratégias de busca definidas na etapa anterior.

v) Testes das melhorias realizadas: Consistiu em realizar testes para verificar a eficácia na recuperação de GOOPs antes e depois das modificações realizadas no *GoopHub*.

vi) Escrita da monografia: Consistiu na escrita desta monografia para registrar os resultados produzidos em todas as etapas retratadas neste anteprojeto.

1.4 Organização do texto

Esta monografia é organizada em quatro capítulos e contém, além deste capítulo de introdução que conta com objetivos e histórico de desenvolvimento, os seguintes capítulos:

Capítulo 2 – Fundamentação Teórica: Apresenta uma revisão da literatura acerca dos temas relevantes ao contexto deste trabalho, a saber: ontologias, reúso de ontologias, GO-FOR e técnicas de busca automatizada. Também apresenta algumas tecnologias que foram utilizadas no desenvolvimento e aprimoramento da ferramenta *GoopHub*.

Capítulo 3 – Evolução de GoopHub: Apresenta a solução proposta e as alterações realizadas em *GoopHub* para implementar a solução e um exemplo para comparação de resultados da busca por GOOPs antes e depois das alterações realizadas.

Capítulo 4 – Considerações Finais: Apresenta a conclusão do trabalho, incluindo algumas dificuldades encontradas, limitações e propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os principais aspectos teóricos que fundamentam este trabalho. Ele está organizado em seis seções, a saber: Ontologias (Seção 2.1), Reúso em Ontologias (Seção 2.2), GO-FOR (Seção 2.3), Processamento de Linguagem Natural (Seção 2.4), Tecnologias Envolvidas neste trabalho (Seção 2.5) e Considerações Finais do Capítulo (Seção 2.6).

2.1 Ontologias

Uma ontologia é uma descrição de conceitos e relacionamentos que existem entre estes conceitos (GRUBER, 1992). Studer, Benjamins e Fensel (1998) detalham que uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada, ou seja, algo entendível para os computadores, que define explicitamente seus conceitos, propriedades e relacionamentos e que representa um conceito sobre algum fenômeno cujo conhecimento é compartilhado entre um grupo de pessoas. Para Guarino (1998) uma ontologia se refere a um artefato de engenharia, que é constituído por um vocabulário específico utilizado para descrever certa realidade, mais um conjunto de suposições explícitas a respeito do significado pretendido para as palavras do vocabulário.

Guarino (1998) define ainda uma classificação das ontologias em quatro tipos, de acordo com seu nível de generalidade, conforme apresentado na Figura 1.

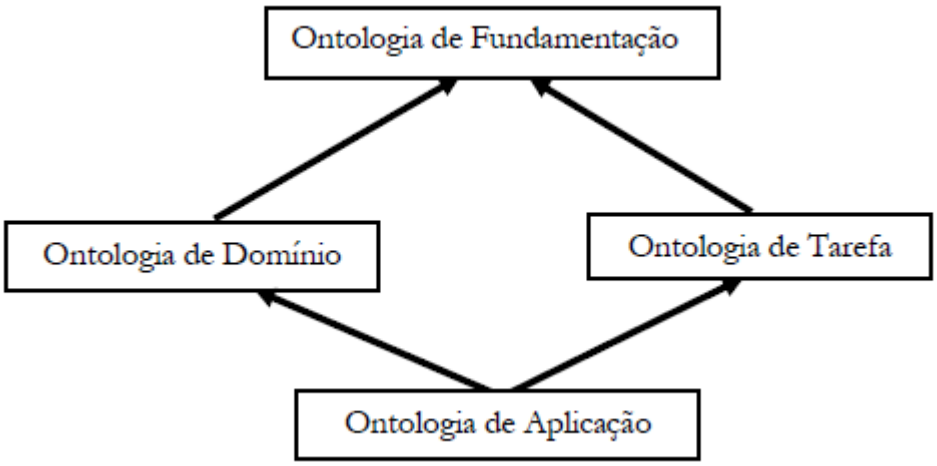

Figura 1 - Tipos de ontologias (GUARINO, 1998).

• *Ontologias de Fundamentação:* descrevem conceitos muito gerais, como espaço, tempo, matéria, objeto, evento, ação etc., que são independentes de um determinado problema ou domínio.

• *Ontologias de Domínio:* descrevem o vocabulário relacionado a um domínio genérico, como medicina ou automóveis, pela especialização dos termos introduzidos na ontologia de fundamentação.

• *Ontologias de Tarefa:* descrevem o vocabulário relacionado a uma tarefa ou atividade genérica, como diagnosticar ou vender, pela especialização dos termos introduzidos na ontologia de fundamentação.

• *Ontologias de Aplicação:* descrevem conceitos que dependem tanto de um domínio particular quanto de uma tarefa, que geralmente são especializações de ambas as ontologias relacionadas.

Guizzardi (2007) estabelece uma diferenciação entre duas classes de ontologia: *ontologias de referência* e *ontologias operacionais*. Ontologias de referência são ontologias construídas com o objetivo de descrever um domínio com o maior detalhamento possível. O foco dessa ontologia está na adequação da representação, uma vez que as especificações resultantes se destinam ao uso por humanos em tarefas como negociação de significado, análise de domínio e resolução de problemas. Já ontologias operacionais são projetadas com foco em garantir propriedades computacionais desejáveis, como eficiência computacional e facilidade de tradução para plataformas de implementação, importantes propriedades para áreas de aplicação de ontologia em Ciência da Computação como Web Semântica e Engenharia de Domínio. Como o uso de linguagens formais altamente expressivas em geral não é interessante do ponto de vista computacional, deve-se projetar e utilizar dentre estes tipos de ontologia aquele que mais se adequa a um determinado propósito.

Em domínios complexos, a representação de conhecimento como uma ontologia única pode resultar em um grande monolito, que se torna difícil de manipular, utilizar, manter e escalar. Em contrapartida, representar cada subdomínio isoladamente é uma tarefa custosa, com alta fragmentação, onde as mesmas complexidades da alternativa anterior acabam ocorrendo (RUY, 2016). Em vista desses fatores, é recomendado realizar-se a modularização de ontologias. A modularização de ontologias refere-se à atividade de identificar um ou mais módulos em uma

ontologia. Um módulo é considerado subparte significativa e independente de uma ontologia. Esta abordagem facilita o particionamento e a extração de partes de uma ontologia maior. (SUÁREZ-FIGUEROA, 2012).

2.2 Reúso em Ontologias

Reúso pode ser definido como um processo no qual ontologias disponíveis são utilizadas para gerar uma nova ontologia (BONTAS; MOCHOL; TOLKS-DORF, 2005). A reutilização é muitas vezes conceituada como um caso especial de design. Intuitivamente, refere-se à tarefa de selecionar algumas ontologias existentes e manipulá-las de alguma forma para satisfazer os requisitos do domínio da nova ontologia (KATSUMI; GRÜNINGER, 2016). Dada a importância do reúso no desenvolvimento de ontologias, algumas abordagens de engenharia de ontologias, como SABiO (FALBO, 2014) e NeOn (SUÁREZ-FIGUEROA et al., 2012), incluem atividades de reúso de ontologias ou recursos ontológicos como parte do processo de desenvolvimento de ontologias (SALAMON, 2018).

Na literatura encontram-se diversos métodos de reúso de ontologias. Durante o processo de criação da ontologia através de reúso, pode-se utilizar um ou mais desses métodos. A seguir são brevemente descritos alguns dos métodos de reúso mais utilizados:

i. Alinhamento: No alinhamento de ontologias mantém-se as duas ontologias originais separadas, sendo adicionadas ligações entre seus termos equivalentes. Estas ligações permitem que as ontologias alinhadas reúsem as informações umas das outras. O alinhamento normalmente é realizado quando as ontologias originais são de domínios complementares (NOY; MUSEN, 1999).

ii. Fusão *(Merging)*: Na fusão é criada uma única ontologia que é uma versão combinada das ontologias originais, sem haver alterações nas ontologias originais. A fusão normalmente é realizada quando as ontologias originais cobrem domínios semelhantes ou sobrepostos (NOY; MUSEN, 1999).

iii. Integração: Na integração de ontologias é criada uma ontologia única através do agrupamento, extensão, especialização ou adaptação de outras ontologias de assuntos diferentes. Na integração de ontologias é possível identificar as regiões

que foram criadas a partir das ontologias originais (PINTO; GÓMEZ-PÉREZ; MARTINS, 1999).

iv. Mapeamento: No mapeamento de ontologias é criada uma estrutura formal com expressões que ligam os termos de uma ontologia aos termos de uma outra ontologia. Este mapeamento pode ser usado para transferir instâncias de dados, esquemas de integração e de combinação, e outras tarefas similares (NOY; MUSEN, 2003).

Com o objetivo de facilitar o desenvolvimento de ontologias, alguns especialistas defendem a adoção de padrões de design ontológico (*Ontology Design Patterns* - ODPs). Padrões são veículos para encapsular conhecimento. São considerados um dos meios mais eficazes para nomear, organizar e raciocinar sobre a lógica do design (FALBO et al., 2013). Segundo Buschmann, Henney e Schmidt (2007), um padrão é uma solução de modelagem que resolve um problema recorrente, que surge em contextos específicos do projeto e apresenta uma solução bem fundamentada para o problema. Padrões ontológicos também podem ser descritos como modelos elaborados e que representam uma perspectiva de consenso sobre como resolver um problema em um domínio específico (GANGEMI; PRESUTTI, 2009).

No desenvolvimento de ontologias é possível fazer uso combinado de padrões ontológicos, o que contribui para acelerar o processo de desenvolvimento . O reúso de padrões ontológicos tem mostrado diversos benefícios, como facilitar o desenvolvimento de ontologias e produzir ontologias de melhor qualidade (POVEDA-VILLALÓN; SUÁREZ-FIGUEROA; GÓMEZ-PÉREZ, 2010).

Embora reúso tenha sido reconhecido como uma abordagem promissora, engenheiros de ontologias ainda enfrentam dificuldades para selecionar as ontologias (ou fragmentos de ontologias) mais adequadas para reúso e integrá-las em uma nova ontologia (PARK; OH; AHN, 2011). Dentre essas dificuldades destacam-se o tamanho e a complexidade das ontologias existentes e potenciais candidatas ao reúso, a obscuridade da lógica do design na maioria das ontologias e a fragilidade das ferramentas de apoio ao engenheiro de ontologias (GANGEMI; PRESUTTI, 2009).

GO-FOR (*Goal-Oriented Framework for Ontology Reuse*) é um framework orientado a objetivos para reúso de ontologias, no qual aplica-se GORE (*Goal-Oriented Requirements*

Engineering)[1] em Engenharia de Ontologias para expressar a lógica do design de fragmentos de modelo de ontologia (REGINATO et al., 2019). GO-FOR baseia-se em quatro princípios que buscam responder como identificar correspondências entre ontologias e estão relacionados a um subconjunto de questões recorrentes no design de ontologias, apontadas por Gruber (1991). Abaixo tem-se um resumo do que trata cada um desses princípios:

(P1) Padronize a terminologia e a pesquisa semântica: A falta de um padrão terminológico para nomear estruturas de ontologia favorece o uso de nomes arbitrários e inexpressivos, prejudicando a busca e recuperação de ontologias adequadas e, consequentemente, comprometendo o reúso de ontologias.

(P2) Aplique a lógica do design (design rationale): Uma das dificuldades para a reutilização de ontologias é a obscuridade da lógica do design. É importante explicitar as razões para o desenvolvimento de uma ontologia da maneira como foi desenvolvida (por exemplo, o que levou o engenheiro de ontologia a incluir certos conceitos na ontologia).

(P3) Resolva sobreposições: Quando os modelos de ontologia são desenvolvidos a partir do zero, sem qualquer preocupação com a reutilização, há uma grande chance de sobreposição com outros modelos de ontologia. Portanto, é importante identificar e resolver sobreposições entre ontologias para evitar redundância e aumentar a reutilização.

(P4) Concentre-se mais em Padrões do que em Modelos: Os modelos de ontologia podem abranger um amplo escopo, abarcando várias exigências. Nesses casos, pode ser difícil identificar um modelo que atenda a um requisito específico desejado para reutilização. Além disso, ao encontrar o modelo, é necessário identificar seu fragmento que atende ao requisito desejado para reutilização. Os padrões são uma abordagem melhor nesses casos. Eles promovem a reutilização, pois modularizam os modelos de ontologia de uma maneira mais fácil de encontrar e reutilizar.

---

[1] GORE visa ao uso de objetivos para auxiliar na identificação de requisitos e no entendimento das razões por trás de tais requisitos. Em GORE, objetivos são usados como uma conceituação útil para elicitar, modelar e analisar requisitos, capturando alternativas e conflitos (HORKOFF et al., 2019). No âmbito da Engenharia de Ontologias, GORE pode auxiliar modelar atores, seus objetivos e interdependências, e refinar o modelo conceitual até que a aplicabilidade da ontologia se torne clara (FERNANDES; GUIZZARDI; GUIZZARDI 2011).

Buscando atender o primeiro princípio, GO-FOR propõe uma terminologia para nomeação de ontologias que consiste no uso de um verbo no infinitivo, acrescido de um substantivo ou uma frase substantiva. Com relação ao verbo, ao analisar várias estruturas de modelos de ontologia e investigar objetivos que eles são capazes de atingir, percebeu-se que os objetivos podem ser geralmente associados a três tipos principais de ações: classificação, identificação e descrição. Assim, sugere-se um conjunto de verbos a serem utilizados para nomear um GOOP, de acordo com o tipo de ação a que seu objetivo se refere. Classificação se aplica quando o objetivo é classificar uma entidade (e.g., um espécime) de acordo com um sistema de classificação que é disposto como taxonomias. Assim, para fragmentos de modelos que são usados para classificar entidades, indica-se o uso do verbo "classificar" para nomear o GOOP (e.g., classificar peixe). Identificação ocorre quando o propósito é identificar um elemento conhecido (por exemplo, um elemento químico da tabela periódica) de acordo com uma taxonomia ou uma lista de valores (enumeração). Aqui, os verbos a serem usados são "identificar" ou "determinar" (e.g., identificar tipo sanguíneo). Finalmente, Descrição ocorre quando um fragmento de modelo se destina a descrever uma entidade (i.e., um evento ou um objeto) representando sua relação com outras entidades que representam suas características. Por exemplo, um nascimento pode ser descrito por sua data, a mãe e o bebê que participou daquele evento. Para descrição, sugere-se usar "descrever", "caracterizar", "definir", "especificar" ou "registrar" (e.g., descrever crime) (REGINATO et al., 2022).

Em GO-FOR, os modelos de ontologia são representados em fragmentos relacionados a objetivos. Esses fragmentos de modelo são estruturas ontológicas independentes, chamadas padrões ontológicos orientados a objetivos (*Goal-Oriented Ontology Patterns* - GOOPs). Assim, os objetivos podem ser usados como parâmetros para apoiar a capacidade de compartilhamento e reutilização do padrão ontológico. Os GOOPs são armazenados em um repositório de padrões ontológicos orientados a objetivos (*Goal-Oriented Ontology Patterns Repository* - GOOPR). Dentro do GOOPR, os GOOPs se relacionam de acordo com as relações entre seus objetivos. A ferramenta *GoopHub*, é uma materialização de um GOOPR e oferece suporte computacional ao GO-FOR. *GoopHub* permite aos engenheiros de ontologias registrar e recuperar GOOPs (REGINATO et al., 2019). A Figura 2 mostra uma visão geral de GO-FOR:

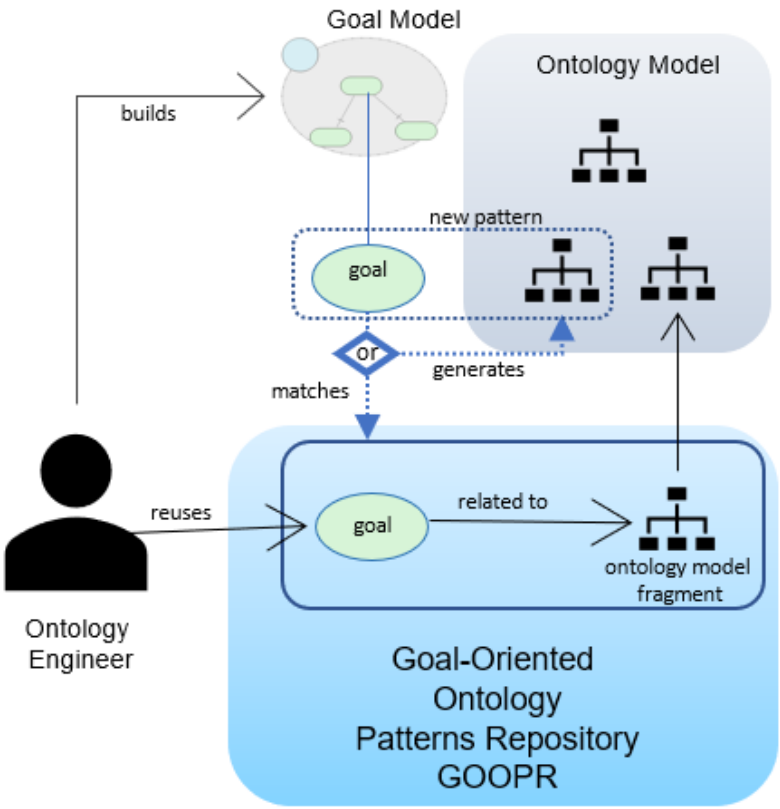

Figura 2 - Visão geral de GO-FOR (REGINATO et al., 2019).

Para que um engenheiro de ontologias reutilize os GOOPs no desenvolvimento de uma ontologia, ele deve iniciar identificando os atores do domínio e elaborando o modelo de objetivos da ontologia, como sugerido em (FERNANDES; GUIZZARDI; GUIZZARDI 2011) e (SALAMON, 2018). A Figura 3 apresenta, um exemplo de modelo de objetivos referente a uma ontologia para o domínio Oferta de Produtos na Web.

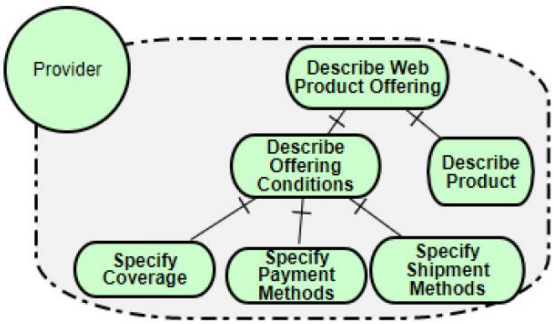

Figura 3 - Exemplo de modelo de objetivos (REGINATO et al., 2022)

Tomando o modelo de objetivos elaborado para a ontologia como base, o engenheiro de ontologias deve verificar se existe um GOOP no GOOPR associado a cada objetivo do modelo. Se existir, o engenheiro de ontologias pode reusar o GOOP realizando a integração

deste com o modelo conceitual da ontologia (*i.e.,* desenvolvimento *com* reúso). Caso não exista, o engenheiro pode criar um fragmento de ontologia que satisfaça aquele objetivo. O engenheiro de ontologias pode, então, relacionar o fragmento com o objetivo, dando origem a um novo GOOP e inseri-lo no GOOPR para uso futuro (*i.e.,* desenvolvimento *para* reúso). A Figura 4 apresenta um exemplo de GOOP. O GOOP apresentado na figura está relacionado ao objetivo *Describe Locality,* o qual tem dois subobjetivos (*Describe Geographic Point* e *Describe Place Name*). Assim, o GOOP associado a *Describe Locality,* é composto por outros dois GOOPs, associados aos subobjetivos. Na figura, os diferentes GOOPs são representados por diferentes cores e regiões delimitadas por linhas tracejadas.

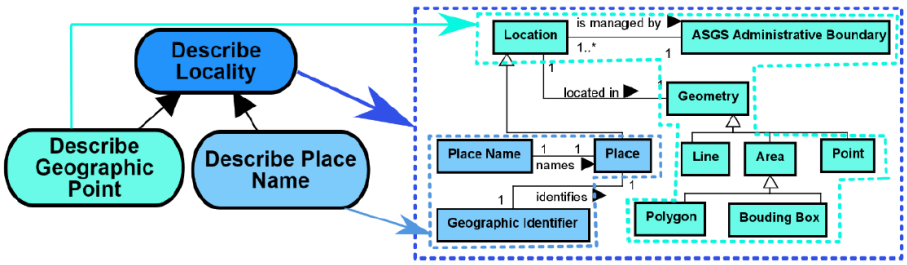

Figura 4 - Exemplo de GOOPs (REGINATO et al., 2022)

2.3 Processamento de Linguagem Natural

O processamento de linguagem natural, mais conhecido pelo termo em inglês *Natural Language Processing* (NLP), é um ramo da Ciência da Computação, Inteligência Artificial e Linguística preocupado com as interações entre computadores e a linguagem humana. Segundo Manning e Schutze (1999), o objetivo da ciência linguística é ser capaz de caracterizar e explicar a multidão de observações linguísticas que circulam ao nosso redor, seja em conversação, escrita ou outros meios. Parte disso tem a ver com o lado cognitivo de como os humanos adquirem, produzem e compreendem a linguagem, parte tem a ver com entender a relação entre enunciados linguísticos e o mundo, e parte tem a ver com compreender as estruturas linguísticas pelas quais a linguagem se comunica.

Linguagem natural é qualquer linguagem que os humanos aprendem no seu ambiente e usam para se comunicar uns com os outros. Qualquer que seja a forma de comunicação, as linguagens naturais são usadas para expressar conhecimentos e emoções e para transmitir respostas a outras pessoas. (RESHAMWALA; MISHRA; PAWAR, 2013).

Está disponível hoje na internet uma massiva quantidade de dados: trilhões de gigabytes, provenientes em grande parte de redes sociais, redes de dispositivos conectados, medições ou fontes publicamente disponíveis. Esta enorme quantidade de informação, porém, é principalmente não-estruturada, porque é produzida especificamente para consumo humano e, portanto, não diretamente processável pelas máquinas. A análise automática de texto envolve uma compreensão profunda da linguagem natural por máquinas, uma realidade do qual ainda estamos muito distantes (CAMBRIA; WHITE, 2014). O processamento de linguagem natural é a coleção de técnicas empregadas para tentar realizar esse objetivo.

O NLP tem sido a base de muitas aplicações ao longo dos últimos anos, como por exemplo extração e análise de grandes volumes de dados, aprimoramento de buscas, tradução e resumo de textos, captura de opiniões e comportamentos e diversas outras. Não é difícil perceber o valor que a automatização de tais tarefas tem para o ser humano, e o ganho que a evolução do NLP proporcionou nas mais diversas áreas. No entanto, a verdadeira compreensão da linguagem natural é uma tarefa desafiadora, pois requer capacidades simbólicas de alto nível (DYER, 1995), incluindo: (i) criação e propagação de ligações dinâmicas; (ii) manipulação de estruturas constitutivas recursivas; (iii) aquisição e acesso de memórias lexicais, semânticas e episódicas; (iv) controle de múltiplos módulos de aprendizado/processamento e roteamento de informações entre esses módulos; (v) fundamentação do nível básico da construção da linguagem (por exemplo, objetos e ações) nas experiências perceptivas/motoras; (vi) representação de conceitos abstratos.

No início, a maioria das pesquisas de NLP era focada na sintaxe, em parte porque o processamento sintático era evidentemente necessário, e em parte porque tinha aplicabilidade mais direta de técnicas de aprendizado de máquina. No entanto, alguns pesquisadores se concentraram na semântica porque a viram como um problema realmente desafiador ou assumiram que o processamento orientado semanticamente seria uma abordagem melhor. Assim, exploraram a correspondência de padrões semânticos usando categorias semânticas, e o conhecimento do mundo foi usado para estender a semântica linguística, juntamente com as redes semânticas como dispositivo de representação do conhecimento. Trabalhos posteriores reconheceram a necessidade de conhecimento externo na interpretação e enfatizaram explicitamente a semântica na forma de semântica com estruturas de caso para representação e processamento orientado semanticamente.

A semântica, no entanto, é apenas uma camada na escala que separa a NLP da compreensão da linguagem natural. Para alcançar a capacidade de processar informações com

precisão e sensatez, modelos computacionais também devem ser capazes de projetar semântica e sentimentos no tempo, compará-los em paralelo de forma dinâmica, de acordo com diferentes contextos e diferentes atores e suas intenções (HOWARD; CAMBRIA, 2013). Isso significa saltar da Curva Semântica para a Curva Pragmática, que permitirá que o NLP seja mais adaptativo e, portanto, de domínio aberto, sensível ao contexto e movido pela intenção (RESHAMWALA; MISHRA; PAWAR, 2013).

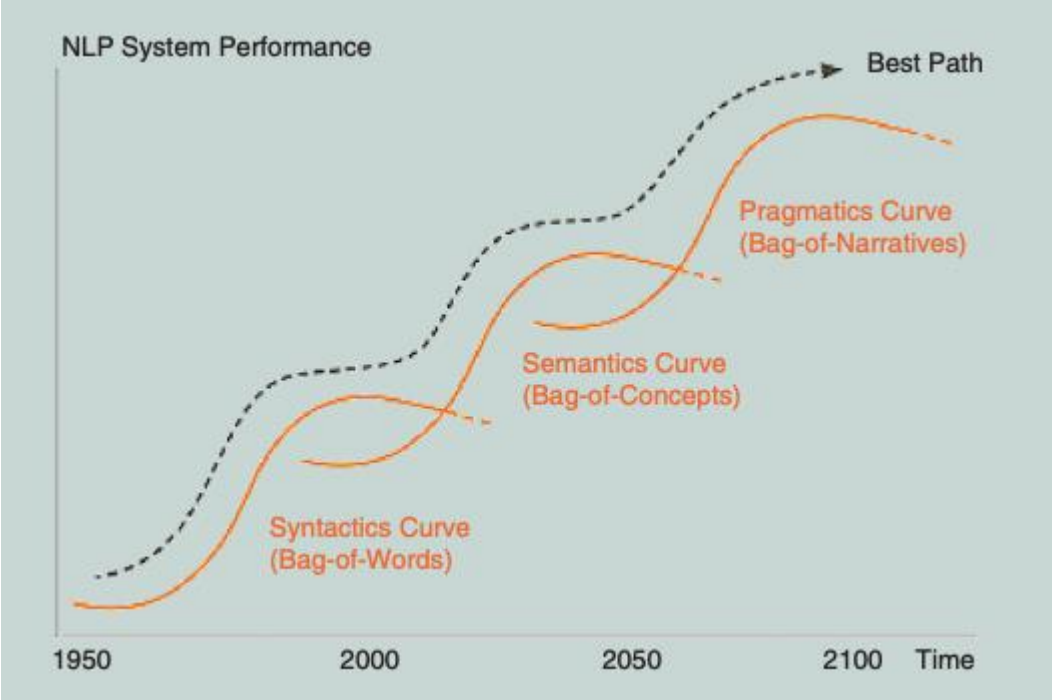

Figura 5 - Evolução prevista da pesquisa em NLP através de três eras diferentes (RESHAMWALA; MISHRA; PAWAR, 2013).

2.3.1 - Métodos de Processamento de Linguagem natural

Em geral, o processo de NLP pode ser dividido em quatro partes: pré-processamento de texto, representação de texto, treinamento de modelo e avaliação de modelo. O pré-processamento de texto visa obter um texto "limpo" para melhorar a eficiência e precisão das análises seguintes, removendo símbolos sem sentido, verificando erros de ortografia, fazendo remoção de palavras de parada, marcação de partes-do-discurso, tokenização e lematização.

Após o pré-processamento do texto, os pesquisadores devem escolher a melhor forma de representar as palavras de forma que possam ser processadas com precisão pelo computador. A representação de texto converte palavras em números, que serão vetores ou matrizes nesta etapa. Com base nestes vetores de palavras, é possível aplicar algoritmos - como classificação, análise de sentimentos, e extração de tópicos etc. - para treinar um modelo para resolver o real

problema. Depois de treinar o modelo, os pesquisadores precisam avaliar o modelo para garantir que ele tenha uma boa generalização (KANG et al., 2020).

A Figura 6 demonstra um fluxograma que descreve as etapas da condução do NLP, bem como exemplo de alguns modelos e algoritmos utilizados:

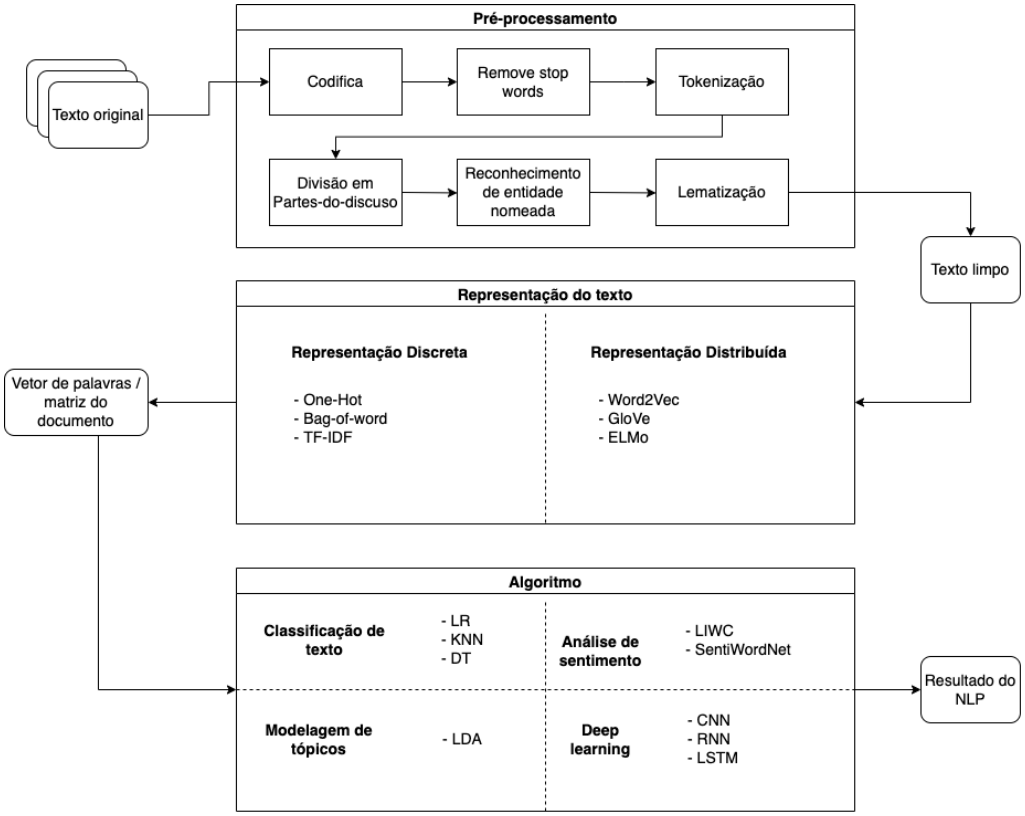

Figura 6 - Fluxograma do NLP.

A seguir algumas das principais etapas são brevemente descritas :

i. Tokenização (*Tokenization)*: É o processo de quebrar o texto em unidades, muitas vezes na forma de palavras e sentenças. Ao tokenizar, os delimitadores que definem um token (espaço, ponto, ponto e vírgula etc.) precisam ser determinados. Existem muitos pacotes de tokenização que fornecem procedimentos de tokenização inteligentes, por exemplo, NLTK e *StanfordCoreNLP* (KANG et al., 2020).

ii. Reconhecimento de entidade nomeada (*Named Entity Recognition* - NER): O reconhecimento de entidade nomeada, às vezes chamado de fragmentação, extração ou identificação de entidade, é a tarefa de identificar e categorizar informações-chave (entidades) no texto. Uma entidade pode ser qualquer palavra ou série de palavras que se refiram consistentemente à mesma coisa. Cada entidade detectada é classificada em uma categoria

predeterminada. Por exemplo, um modelo de aprendizado de máquina NER pode detectar a palavra "Exemplo.SA" em um texto e classificá-lo como uma "Empresa".

iii. Partes-do-discurso (*Parts of Speech*): No método de pré-processamento, as partes do discurso são aplicadas para a identificação de substantivos, verbos ou adjetivos e a frequência da palavra resultante é usada para análise posterior. Os pesquisadores escolhem diferentes partes do discurso de acordo com seus contextos de pesquisa, por exemplo, frases de avaliação (adjetivos e advérbios) para medir as características do serviço (GHOSE; IPEIROTIS; Li, 2012).

iv. Lematização (*Lemmatization*): Lematização é o processo de agrupar as formas flexionadas de uma palavra para que possam ser analisadas como um único item (*Collins English Dictionary. Glasgow: HarperCollins Publishers*, 1994). A lematização é aplicada principalmente em mineração de texto e é usada para análise de expressões de texto mais precisas, por exemplo, em tradução automática. Existem vários lematizadores pré-empacotados na maioria das ferramentas de mineração de texto, como, por exemplo, WordNet, SpaCy e Snowball em Python.

v. Análise de sentimento (*Sentiment Analysis*): A análise de sentimento é a mineração contextual de texto que identifica e extrai informações subjetivas do material de origem, além de quantificar e estudar sistematicamente estados afetivos e informações subjetivas. A análise de sentimento é amplamente aplicada em avaliações e respostas a pesquisas, mídia online e social e textos de notícias onde os autores normalmente expressam sua opinião/sentimento menos explicitamente. Pode ser entendida como um problema de classificação e mineração de texto (HAMBORG et al., 2021).

2.3.2 - Similaridade de Cosseno

Uma medida de similaridade é uma ferramenta importante para determinar o grau de semelhança entre dois objetos. A similaridade de cosseno mede a similaridade entre dois vetores de um espaço de produto interno, ela é medida pelo cosseno do ângulo entre dois vetores e determina se dois vetores estão apontando aproximadamente na mesma direção. A equação matemática de semelhança de cosseno entre dois vetores diferentes de zero é:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figura 7 - Fórmula da similaridade de cossenos.

Medidas de similaridade entre conjuntos de texto de lógica difusa (*fuzzy*), têm ganhado atenção de pesquisadores por suas amplas aplicações em diversas áreas (BUSTINCE; BARRENECHEA; PAGOLA, 2006). A medida de similaridade de cosseno é uma das muitas medidas de similaridade existentes. Na aplicação para textos de lógica difusa, ela aplica o cosseno do ângulo entre as representações vetoriais dos dois conjuntos *fuzzy*. A similaridade de cosseno é uma medida clássica utilizada na recuperação de informação e é a medida de similaridade vetorial mais amplamente relatada (SALTON; MCGILL, 1983). A semelhança de cosseno é vantajosa porque mesmo que dois textos semelhantes estejam distantes por causa do tamanho, por exemplo, uma palavra aparece muitas vezes em um texto e poucas vezes no outro, eles ainda podem ter um ângulo menor entre eles. Quanto menor o ângulo, maior a semelhança.

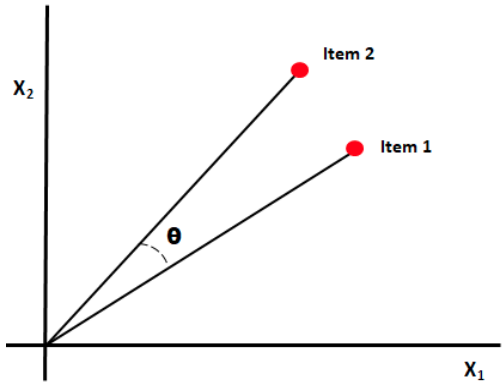

Figura 8 - Similaridade de cosseno exibida em ângulo.

2.3.3 - BERT

O modelo BERT foi proposto em "*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*" (DEVLIN, et al., 2018). É um transformador bidirecional pré-treinado usando uma combinação de *masked language modeling (MLM)* e previsão de próxima frase (*next sentence prediction - NSP*) em um grande repertório que compreende o *Toronto Book Corpus* e a *Wikipedia*.

O MLM é um objetivo de pré-treinamento auto supervisionado. É amplamente utilizado no processamento de linguagem natural para aprender representações de texto. O MLM treina um modelo para prever uma amostra aleatória de *tokens* de entrada que foram substituídos por um espaço reservado (*mask*) (YAMAGUCHI et al., 2021).

A previsão de próxima frase também é um objetivo de pré-treinamento auto supervisionado, ele fornece uma frase, e treina um modelo para prever a próxima frase. Muitas tarefas importantes de refinamento, como resposta de pergunta (*Question Answering - QA)* e Inferência de linguagem natural (*Natural Language Inference - NLI*), são baseadas na compreensão da relação entre duas frases, que não é capturada diretamente pelo MLM. Para treinar um modelo que entende as relações entre as sentenças, é necessário antes treiná-lo para uma tarefa binarizada de previsão da próxima sentença que pode ser gerada a partir de qualquer repertório de um único idioma. Enquanto o MLM ensina o BERT a entender as relações entre as palavras, o NSP ensina o BERT a entender as dependências de longo prazo entre as frases. Sem o NSP, o BERT tem um desempenho pior em todas as métricas (DEVLIN, et al., 2018).

O fato de BERT ser bidirecional quer dizer que ele mascara uma porcentagem (usualmente quinze por cento) da entrada de tokens aleatórios e então prevê os tokens que foram mascarados, dessa forma ele apenas prevê as palavras mascaradas ao invés de reconstruir toda a entrada. Isso porque caso contrário um modelo com transformação bidirecional permitiria que cada palavra, indiretamente, visse a si mesma, e o modelo poderia trivialmente prever a palavra alvo em um contexto de várias camadas. Essa é uma diferença significativa entre BERT e outros modelos de entendimento de linguagem pré-treinados anteriores, como por exemplo OpenAI GPT ou ELMo, e um dos fatores do porquê ele é considerado *state-of-the-art* na área atualmente (DEVLIN, et al., 2018).

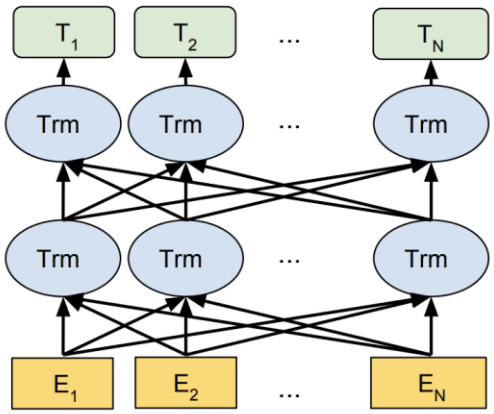

Figura 9 - BERT usa uma transformação bidirecional (DEVLIN, et al., 2018).

2.4 Tecnologias

Nesta seção são apresentadas breves descrições das tecnologias utilizadas no desenvolvimento deste trabalho.

As linguagens de programação Java e JavaScript, o *framework* Spring Boot e o banco de dados Stardog são as tecnologias utilizadas na construção de *GoopHub*. Para a implementação das melhorias tratadas neste projeto, utilizou-se ainda a linguagem de programação Python, o *framework* Flask e a plataforma Docker.

i. Java: Linguagem de programação compilada, orientada a objetos, estaticamente tipada e independente de plataforma. Utilizada no *back-end* de *GoopHub*.

ii. JavaScript: Linguagem de programação interpretada, orientada a objetos, dinamicamente tipada, multiparadigma e assíncrona. Utilizada no *front-end* de *GoopHub*.

iii. Spring Boot: Framework open-source para a plataforma Java. Trata-se de um *framework* não intrusivo, baseado nos padrões de projeto inversão de controle (IoC) e injeção de dependência. No Spring o *container* se encarrega de instanciar as classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do *framework*, o que é chamado de *auto-wiring* ou ainda anotações nas classes, métodos e propriedades. Dessa forma, o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.

iv. Stardog: Banco de dados baseado em triplas, não relacional, que é capaz de realizar transações, consultas SPARQL e contempla semântica formal e suporte ao raciocínio através da linguagem OWL (*Ontology Web Language*). Além disso, Stardog abarca os conceitos de grafo de conhecimento (*Knowledge graph*). Um grafo de conhecimento é um grafo multi-relacional composto de entidades como nós e relações como diferentes tipos de bordas. Um grafo de conhecimento fornece uma estrutura para integração, análise e compartilhamento de dados (WANG et al., 2014).

v. Python: Linguagem de programação interpretada, multiparadigma e de tipagem dinâmica. Utilizada na API responsável por aplicar os algoritmos de busca e processamento de linguagem natural, e nos próprios algoritmos utilizados. Python foi escolhida por possuir uma série de fatores que fazem dela uma ótima candidata para projetos que incluem processamento de linguagem natural, como por exemplo a sintaxe

simples e semântica transparente, e pela variedade de bibliotecas disponíveis, por estar sendo amplamente utilizada com estes propósitos atualmente.

vi. Flask: F*ramework web* escrito em Python. É classificado como um *micro framework* porque não requer ferramentas ou bibliotecas particulares, possuindo um núcleo simples, porém extensível.

vii. Docker: Conjunto de produtos de plataforma como serviço que usam virtualização a nível de sistema operacional para entregar *software* em pacotes chamados contêineres. Os contêineres são isolados uns dos outros e agrupam seus próprios *softwares,* bibliotecas e arquivos de configuração. Os benefícios da utilização de Docker em um contexto geral e neste trabalho são, dentre outros, maior disponibilidade do sistema, visto que *GoopHub* já possui dois projetos, *front-end* e *back-end*, que necessitam serem compilados separadamente, e acrescentou-se mais um projeto com diferentes necessidades.

2.5 Considerações Finais do Capítulo

Este capítulo apresentou fundamentos teóricos e tecnológicos relevantes para o desenvolvimento deste trabalho. No que diz respeito a aspectos teóricos, foram abordados tópicos relacionados a ontologias e reúso de ontologias, GO-FOR e processamento de linguagem natural, que fornecem a base teórica para o aprimoramento da ferramenta *GoopHub* proposto neste trabalho. No âmbito dos aspectos tecnológicos, foram apresentadas as tecnologias envolvidas no processo de construção e aprimoramento da ferramenta, descrevendo as linguagens e *frameworks* que foram utilizados para o desenvolvimento e implementação da solução proposta.

Capítulo 3

Evolução de *GoopHub*

Este capítulo apresenta as evoluções realizadas em GoopHub e discute aspectos relacionados ao seu desenvolvimento. Na Seção 3.1 é apresentado o objetivo de GoopHubo Sistema. Na Seção 3.2 é apresentada uma visão geral da solução. Na Seção 3.3 é apresentada a arquitetura da solução. A Seção 3.4 aborda a implementação da solução e a Seção 3.5 apresenta algumas das telas do sistema. Por fim, a Seção 3.6 apresenta um exemplo de uso da solução desenvolvida.

3.1 Objetivo de *GoopHub*

Atualmente, a busca por ontologias para reúso no desenvolvimento de novas ontologias pode ser bastante custosa, pois, embora haja muitos repositórios de ontologias na Web, muitos não são atualizados com frequência e são insuficientes no que diz respeito às funcionalidades de busca pela ontologia mais adequada para reúso em uma situação particular. Isso contribui para tornar o reúso de ontologias pouco vantajoso, pois o tempo gasto para buscar ontologias adequadas ser alto (PARK; OH; AHN, 2011).

Visando apoiar o reúso de padrões ontológicos e promover o uso de GO-FOR, *GoopHub* foi desenvolvido para apoiar a criação, armazenamento e busca de GOOPs (*Goal-Oriented Ontology Patterns*). *GoopHub* consiste em uma interface web para um repositório que armazena GOOPs (i.e., o GOOPR de GO-FOR), permitindo que engenheiros de ontologias realizem *uploads* e downloads de GOOPs.

A partir do modelo de objetivos construído para a ontologia a ser desenvolvida, o engenheiro de ontologias pode utilizar *GoopHub* para buscar GOOPs seguindo uma abordagem *top-down*, buscando por objetivos mais gerais, decompostos em subobjetivos; ou *bottom-up*, procurando por objetivos mais específicos e integrando GOOPs a eles associados até que os objetivos mais gerais sejam atendidos.

Na versão de *GoopHub* anterior a este trabalho (NOGUEIRA, 2019), a busca por GOOPs retornava apenas GOOPs que tinham em seu nome exatamente os mesmos termos fornecidos pelo engenheiro de ontologias, o que é uma forma bem limitada de busca. Os aprimoramentos realizados em *GoopHub* no escopo deste projeto de graduação, visam otimizar a identificação e seleção de GOOPs através da implementação da estrutura gramatical para

nomear padrões ontológicos proposta em (REGINATO et al., 2019) e, também, através da aplicação de técnicas de busca automatizada, utilizando processamento de linguagem natural, para fornecer contexto à busca, e dessa forma facilitar o reúso de padrões ontológico.

3.2 Visão Geral da Solução

A Figura 10 apresenta uma visão geral da solução proposta, por meio de uma adaptação do diagrama de sequência da UML para representar o funcionamento da busca de GOOPs em *GoopHub*.

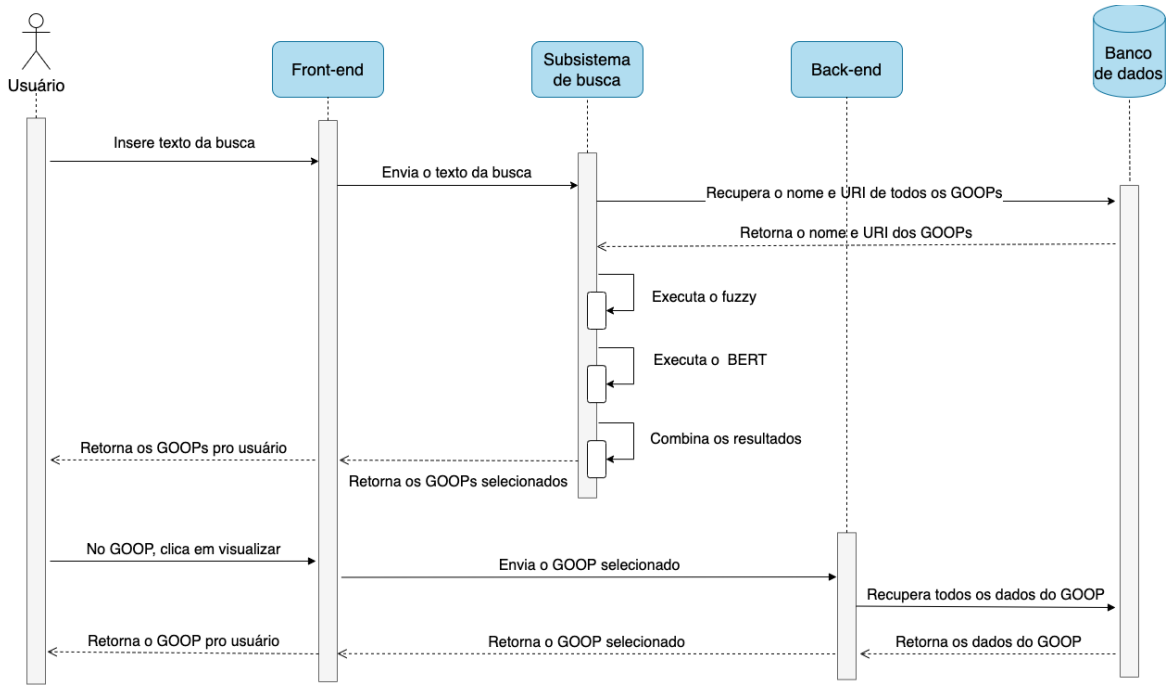

Figura 10 - Adaptação de um diagrama de sequência de Busca de GOOPs.

3.3 Arquitetura de *Software*

Arquitetura de *Software* de um sistema computacional é a estrutura (ou estruturas) do sistema que compreende elementos de *software*, propriedades externamente visíveis desses elementos e os relacionamentos entre elas (BASS; CLEMENTS; KAZMAN, 2003). Os aprimoramentos na ferramenta *GoopHub* foram feitos utilizando as tecnologias apresentadas na Seção 2.3. A seguir apresenta-se a arquitetura de *GoopHub* com essas modificações.

A ferramenta *GoopHub* foi desenvolvida seguindo a arquitetura cliente-servidor. Para o projeto de arquitetura da ferramenta, optou-se por dividi-la em três subsistemas: o subsistema do servidor e o subsistema de busca (*back-end*) e o subsistema do cliente (*front-end*). O subsistema

do servidor foi desenvolvido utilizando o framework Spring Boot e baseia-se no padrão de arquitetura de software MVC (*Model, View, Controller*), que é dividido nas três camadas que formam seu nome:

i. *Model*: Camada de modelo, responsável pela lógica de negócio e pela interação da aplicação com o banco de dados.

ii. *View*: Camada de visão, responsável pela interação com o usuário, exibindo os dados por meio de linguagens como HTML.

iii. *Controller:* Camada de controle, responsável pela comunicação entre as camadas anteriores. As requisições recebidas são processadas pelo *Controller*, que então envia o retorno para a *View*.

O subsistema de busca foi desenvolvido utilizando o *framework* Flask, e consiste em um *controller* API REST. API REST é uma interface de programação de aplicações que está em conformidade com as restrições do estilo de arquitetura REST, permitindo a interação com serviços *web RESTful*. REST é a sigla em inglês para "*Representational State Transfer*", que em português significa transferência de estado representacional.

O subsistema do cliente foi desenvolvido utilizando o *framework* Angular. Um *software* desenvolvido em Angular é composto por diversos elementos como: módulos, componentes, templates e serviços. Esses elementos fazem parte da arquitetura do Angular, essa arquitetura de *software* orientada a componentes implementa conceitos de dois padrões de arquitetura de *software*: MVC (*Model, View, Controller*) e MVVM (*Model, View, View-Model*) que é um padrão de *software* semelhante ao MVC, com a diferença de que o *View-Model* utiliza recurso de *data binding* para fazer com que a *View* seja atualizada automaticamente quando ocorrer uma modificação no *Model*:

i. Um *Model* define dados que serão apresentados no *Template;*

ii. Um *Template* (*View*) apresenta os dados do *Model;*

iii. Um *View-Model* determina o comportamento do *Template*.

Se o *View-Model* atualiza o *Model*, então o *Template* tem que ser atualizado automaticamente. Se o usuário atualiza o *Model* por meio do *Template* (usando um formulário,

por exemplo) o *View-Model* também precisa ter acesso ao *Model* atualizado. O *Data Binding* atua garantindo que esse processo ocorra corretamente. A Figura 11 apresenta um diagrama representando a divisão da arquitetura de *GoopHub* nos elementos descritos acima:

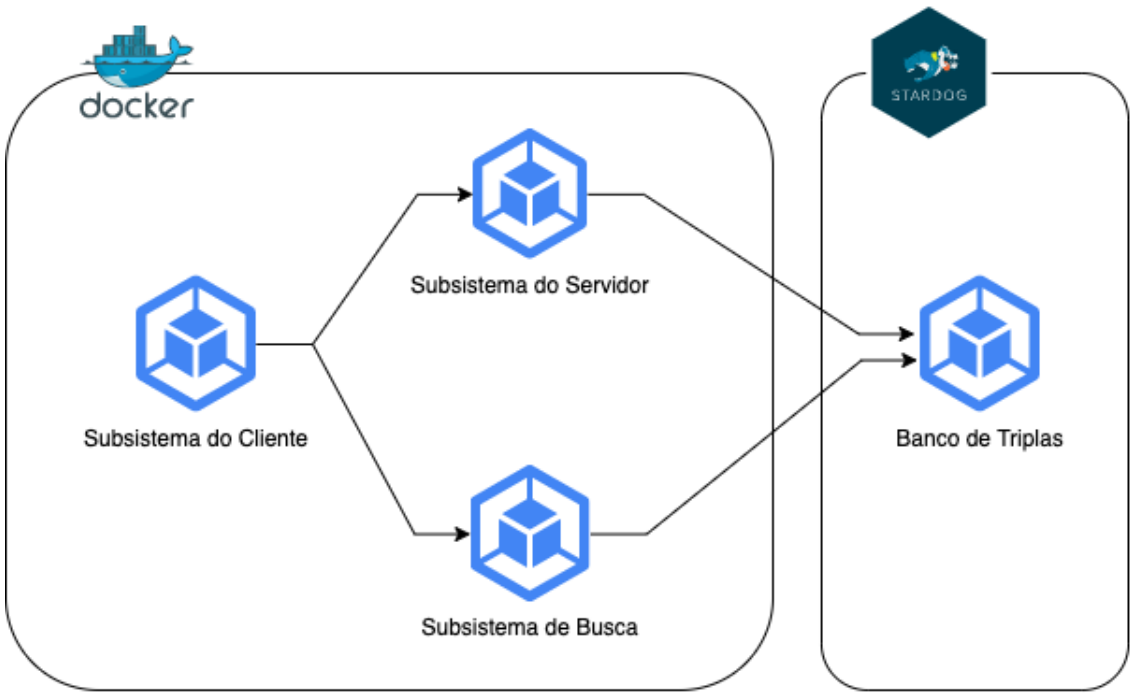

Figura 11 - Arquitetura do GoopHub.

i. Subsistema do Cliente: *front-end* do sistema, feito em Angular.

ii. Subsistema do Servidor: *back-end* do sistema, feito em Spring Boot. Responsável pelas regras de negócio e persistência dos dados.

iii. Subsistema de Busca: Faz parte do *back-end* do sistema, feito em Python. Responsável por recuperar as instâncias do banco de dados, aplicar os algoritmos de NLP e retornar os dados para o *front-end*.

iv. Banco de Triplas: Banco de dados do sistema. Banco Stardog.

3.4 Implementação

O projeto do *GoopHub* é dividido em quatro repositórios, que estão apresentados na Figura 12 a seguir. Para o aprimoramento da ferramenta foram feitas modificações em *goophub-*

backend e *goophub-frontend*, que serão apresentadas na próxima seção, e foram criados *goophub-docker* e *goophub-search*, cuja implementação será explicada nesta seção.

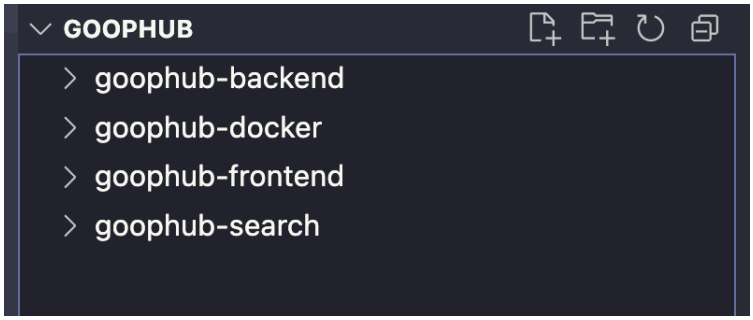

Figura 12 - Repositórios do projeto.

3.4.1 - *GoopHub-search*

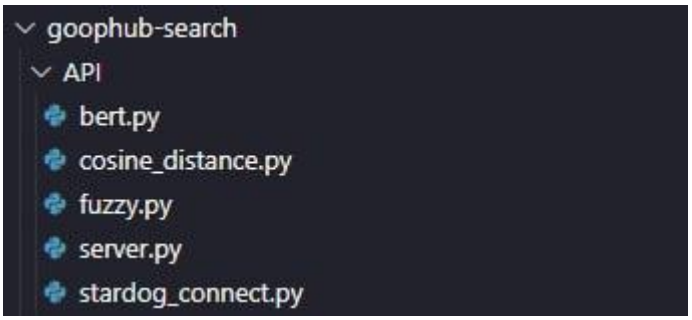

Figura 13 - Estrutura do subsistema de busca.

A Figura 13 mostra a estrutura de arquivos do subsistema de busca intitulado *goophub-search*. Primeiramente foi implementado no projeto o algoritmo de similaridade de cossenos, intitulado cosine_distance.py. Depois foi implementado o algoritmo de *fuzzy search* no arquivo intitulado fuzzy.py, e por fim o modelo BERT no arquivo bert.py. Nas seções que se seguem a implementação de cada um deles será detalhada.

3.4.1.1 - Similaridade de cosseno

Pensando em encontrar variações de texto, aplicamos o algoritmo de similaridade de cosseno. Nessa implementação, o algoritmo verifica apenas similaridade entre palavras completas. Sentenças que possuam as mesmas palavras, porém em ordem diferente, ou que possuam algumas palavras iguais são retornadas como similares. Na Figura 14, vemos alguns detalhes da implementação do algoritmo.

```python
from sklearn.feature_extraction.text import CountVectorizer
from scipy.spatial import distance

def cosine_distance_countvectorizer_method(s1, s2):
    allsentences = [s1 , s2]
    vectorizer = CountVectorizer(lowercase=True)
    all_sentences_to_vector = vectorizer.fit_transform(allsentences)
    text_to_vector_v1 = all_sentences_to_vector.toarray()[0].tolist()
    text_to_vector_v2 = all_sentences_to_vector.toarray()[1].tolist()

    cosine = distance.cosine(text_to_vector_v1, text_to_vector_v2)
    similarity = round((1-cosine)*100,2)

    return similarity
```

Figura 14 - Algoritmo de similaridade de cosseno.

3.4.1.2 - Fuzzy

Pensando em encontrar palavras que o usuário tenha digitado errado, flexões de palavras ou pequenas variações de texto, implementou-se o algoritmo de *fuzzy search*, que usa a Distância de Levenshtein, uma métrica para medir a diferença entre duas sequências. Neste algoritmo, as sequências são comparadas e retorna-se a porcentagem de semelhança entre elas, o *score*. Na Figura 15 vemos o *score* para a comparação entre duas frases que são títulos de GOOPs.

```
>>> fuzz.ratio("Teacher: Describe evaluation system", "Teachr: Describing system")
[out]: 80
```

Figura 15 - Fuzzy ratio entre duas sentenças.

No subsistema de busca, compara-se o termo enviado pelo usuário com o nome dos GOOPs no banco de dados e retornam-se aqueles que obtiveram *score* acima de setenta, pois possuem a escrita muito semelhante.

```python
from fuzzywuzzy import fuzz
import pandas as pd

def fyzzy_search(search, GOOP_DATA):
    result_list = []
    for s in GOOP_DATA:
        ratio = fuzz.ratio(search.lower(), s.lower())
        result_list.append((s, ratio))

    sorted_result = sorted(result_list, key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(sorted_result, columns=['sentence', 'score'])
    df = df[df['score'] > 70]
    return df['sentence'].to_list()
```

Figura 16 - Algoritmo de fuzzy search.

Na aplicação do *fuzzy search*, percebeu-se que este algoritmo trazia os mesmos ganhos que a aplicação da similaridade de cosseno, e mais alguns diferenciais, pois verifica a similaridade mesmo para palavras incompletas e flexões de palavras. No escopo deste projeto, não se enxergou que por si só a similaridade de cosseno trazia valor à busca, necessitando de algum refinamento ou uso combinado com outros algoritmos e modelos para auxiliar na seleção dos GOOPs, portanto, após os testes o algoritmo da similaridade de cosseno foi retirado da rotina de busca.

3.4.1.3 - BERT

Pensando em fornecer busca com similaridade de contexto para o sistema, escolheu-se implementar o modelo BERT, pois, atualmente, ele é considerado o *state-of-the-art* de NLP. O BERT *Tokenizer* é um tokenizador que funciona com o BERT e possui muitas funcionalidades para qualquer tipo de tarefa de tokenização. Ele foi importado na linha um e aplicado na linha quatro da Figura 17. Nas linhas dois e cinco, vemos a importação e utilização do model*o 'distilbert-base-uncased'* utilizado no MLM do BERT.

```
1 from transformers import BertTokenizer
2 from transformers import BertModel
3
4 tokenizer = BertTokenizer.from_pretrained('distilbert-base-uncased')
5 model = BertModel.from_pretrained('distilbert-base-uncased')
```

Figura 17 - Tokenizador e MLM do BERT.

Na Figura 18, vemos a API que executa o modelo BERT receber a query que contém a sentença informada pelo usuário no *frontend,* na linha um, e aplicar a função de vetorização, que cria um vetor de tokens. Esse vetor de tokens é comparado com os vetores de tokens que contêm os nomes dos GOOPs presentes no banco de dados, e então é calculado o nível de proximidade de contexto entre esses vetores baseado no pré-treinamento do BERT. Após essa verificação o algoritmo retorna de zero até 'k' (que foi definido como dez) vetores mais próximos contextualmente.

```
1 query = vectorize(query)
2 ids, distances = index_bert.knnQuery(query, k=10)
```

Figura 18 - Vetorização do BERT.

```
1 from flask import Flask
2 from bert import bert
3
4 @app.route('/bert', methods=['POST'])
5 def bert_search():
6     if request.method == 'POST':
7         query = request.form['query']
8         result = bert(query, GOOP_DATA)
9         return json.dumps(result)
10
11 app.run(host='0.0.0.0', port=5000, debug=False)
```

Figura 19 - Rota de execução do BERT.

A API de busca faz a combinação dos resultados do algoritmo *fuzzy search* com a dos resultados da aplicação do modelo BERT. Primeiro ranqueia os possíveis GOOPs retornados

pelo *fuzzy*, pois os nomes de GOOPs com *score* acima de 70 são muito parecidos entre si, e depois combina a esses GOOPs os GOOPs ainda não selecionados que tenham sido retornados pelo algoritmo do BERT, que são próximos em contexto, e então retorna uma resposta em formato *json* contendo os dados dos GOOPs selecionados para o *frontend*.

3.4.2 - *GoopHub-docker*

O Compose é uma ferramenta para definir e executar aplicativos Docker de vários contêineres. Com o Compose, utiliza-se um arquivo YAML para configurar os serviços de aplicativos. A Figura 20 mostra a configuração do arquivo *docker-compose.yaml*.

```yaml
version: '3'
services:
  goophup-frontend:
    image: node:11.15.0
    entrypoint: sh ./goophub-frontend-entrypoint.sh
    ports:
      - 4200:4200
    networks:
      - goophub
  goophub-backend:
    image: openjdk:8-jdk-alpine
    entrypoint: sh ./goophub-backend-entrypoint.sh
    ports:
      - 8080:8080
    networks:
      - goophub
  goophub-search:
    image: python:3.7-alpine
    working_dir: /goophub-search
    entrypoint: sh ./goophub-search-entrypoint.sh
    ports:
      - 5000:5000
networks:
  goophub:
    driver: bridge
    external: true
```

Figura 20 - Código de configuração dos recursos no Docker.

No arquivo estão declarados os três serviços da aplicação, cada serviço possui suas configurações: imagem, *script* de *entrypoint* e a porta que será utilizada para comunicação do serviço. O *entrypoint* executa as configurações para instalação de dependências e compilação da

imagem de cada ambiente, por exemplo, no *goophub-backend*, gera o arquivo Java compilado e instala os pacotes maven. Finalizadas as configurações, o docker cria e inicia os serviços nas portas referenciadas, para que se comuniquem pela rede. Dessa forma podemos executar todo o projeto rodando apenas o comando '*docker-compose up*'.

3.5 A Ferramenta *GoopHub*

GoopHub é uma aplicação *web* que funciona como um intermediador entre o usuário e o repositório, facilitando a busca por GOOPs. Ela foi desenvolvida inicialmente em(NOGUEIRA, 2019) e evoluída neste trabalho. Ao acessar o endereço da aplicação (dev.nemo.inf.ufes.br/goophub), o usuário é direcionado à página principal, como mostra a Figura 21. Partindo dela é possível acessar as funcionalidades que o *GoopHub* provê, sendo elas: busca, visualização e *download* de GOOPs, *upload* de modelos e um *endpoint* para realizar consultas SPARQL. A seguir são apresentadas informações sobre a utilização da ferramenta.

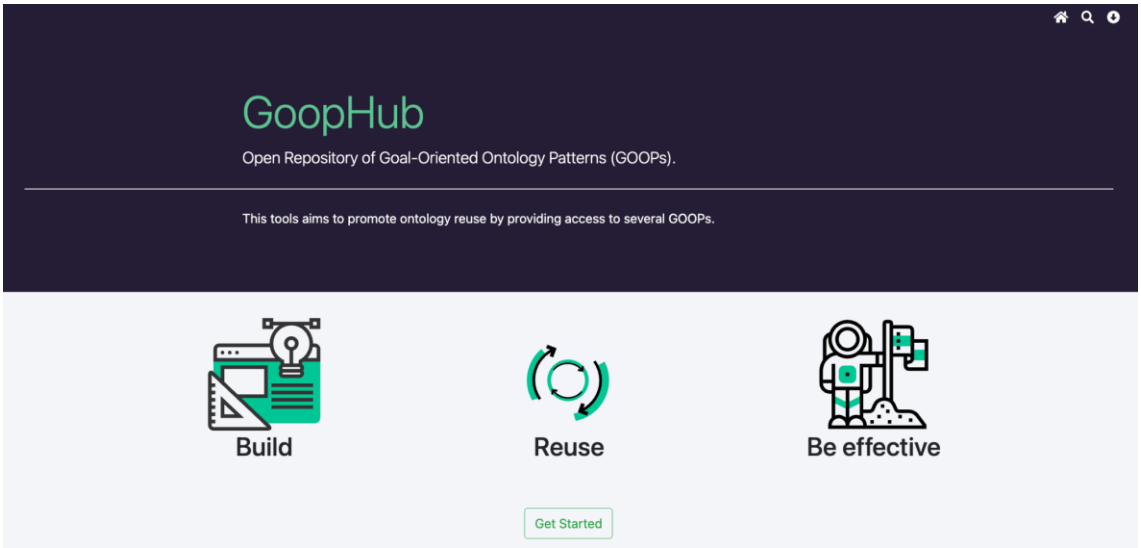

Figura 21 - Página inicial do GoopHub (NOGUEIRA, 2019).

A página de *Upload* de GOOPs permite que o usuário faça o registro de um GOOP associando-o a um Objetivo Complexo (*Complex Goal)* ou um Objetivo Atômico *(Atomic Goal)*. Um Objetivo Complexo é composto de um outro Objetivo associado por uma decomposição AND ou OR. Decomposições AND são usadas quando todos os subobjetivos devem ser satisfeitos para que o objetivo principal seja alcançado. Uma decomposição OR ocorre quando apenas um subobjetivo necessita ser satisfeito.

Para realizar o *upload* o usuário insere seus dados e nomeia o GOOP a ser enviado. Para nomeação dos GOOPs, foram realizadas modificações na versão desenvolvida por Nogueira

(2019) de acordo com os princípios de padronização da terminologia propostos em GO-FOR (REGINATO et al., 2019). Dessa forma, na nova versão, o usuário precisa informar o Ator ao qual o objetivo do GOOP se refere, selecionar um Verbo dentre os disponíveis (*Classify, Define, Describe, Specify*) e informar um Substantivo ou Frase Substantiva.

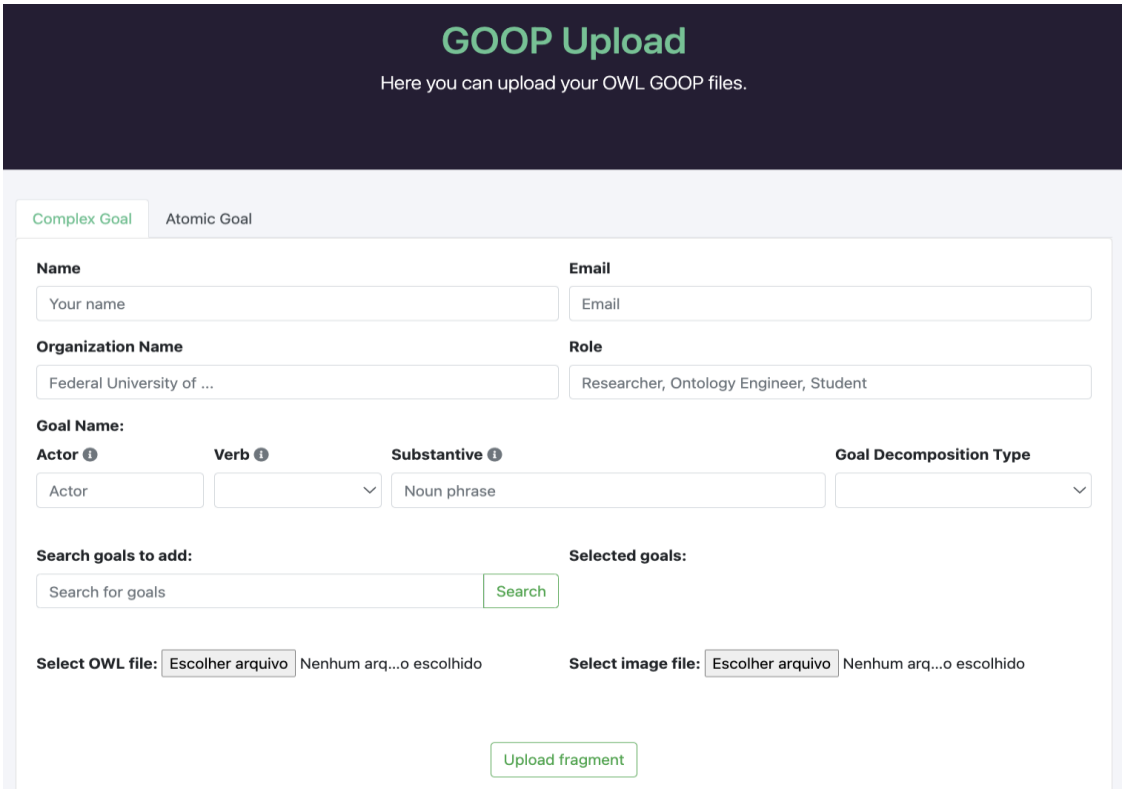

Figura 22 - Página de Upload de Objetivo Complexo.

Também foram incluídos ícones de informação relacionados a cada parte da nomeação do GOOP. Ao passar o mouse sobre o ícone de informação, o usuário pode ver uma breve descrição do que deve ser inserido naquele campo, de forma a auxiliá-lo na escolha da nomeação do GOOP que será registrado.

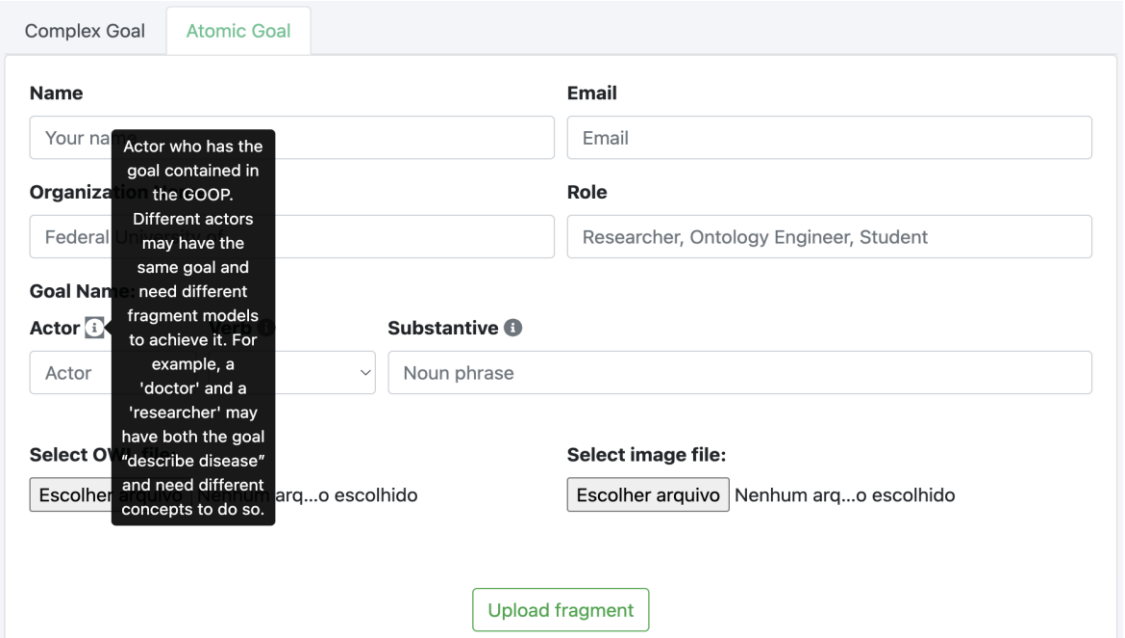

Figura 23 - Página de Upload de Objetivo Atômico com detalhe de dica.

A página de busca permite que o usuário informe o objetivo para o qual deseja encontrar GOOPs. A ferramenta retorna para o usuário, na forma de *cards*, os GOOPs relacionados ao objetivo informado, exibindo para cada um deles sua descrição e informando se são relacionados a um objetivo atômico ou complexo. Além disso, foi incluído na exibição dos GOOPs o nome do ator ao qual o objetivo do GOOP se refere, isso é importante pois diferentes atores podem ter um mesmo objetivo, mas precisarem de modelos de ontologia diferentes para alcançá-lo. GOOPs associados a objetivos complexos podem ser decompostos em outros. A Figura 24 ilustra um exemplo, considerando a versão proposta em (NOGUEIRA, 2019), no qual o usuário buscou por "*evaluation*" e teve como retorno um *card* representado um GOOP que tem como ator "*Teacher*" e satisfaz o objetivo "*Describe evaluation system*".

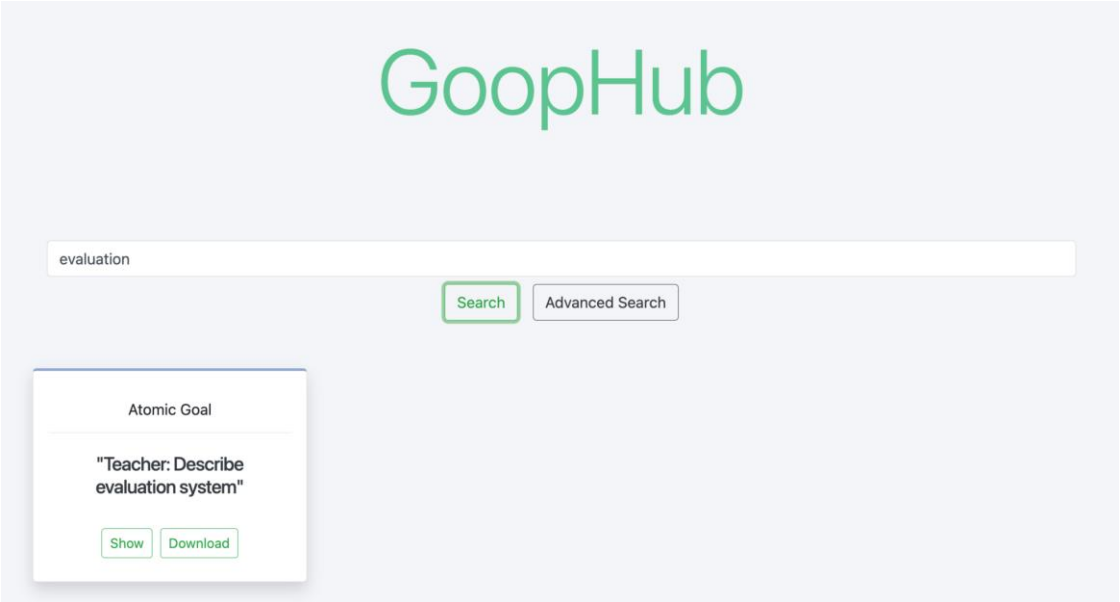

Figura 24 - Página de busca.

A partir do resultado da busca, o usuário pode visualizar os GOOPs (botão "*Show*" no *card* representando o GOOP) ou fazer o *download* do arquivo OWL/RDF (botão "*Download*" no *card* representando o GOOP) correspondente.

3.6 - Comparação de Uso da Busca por GOOPs Antes e Depois do Aprimoramento

Nesta seção iremos fazer a comparação de uma seleção realizada no *GoopHub* sem e com a utilização do subsistema de busca implementado neste trabalho. A seleção implementada anteriormente em *GoopHub* (NOGUEIRA, 2019) buscava apenas pelas palavras contidas no nome do GOOP, com a aplicação da busca literal do *Stardog*, que internamente utiliza o Apache Lucene para realizar essa pesquisa. Com a utilização apenas da busca literal, foi inserida a palavra *"school"* no campo de busca, e a busca não retornou nenhum resultado, pois o banco de dados não possuía nenhum GOOP que continha a palavra *school* em seu nome.

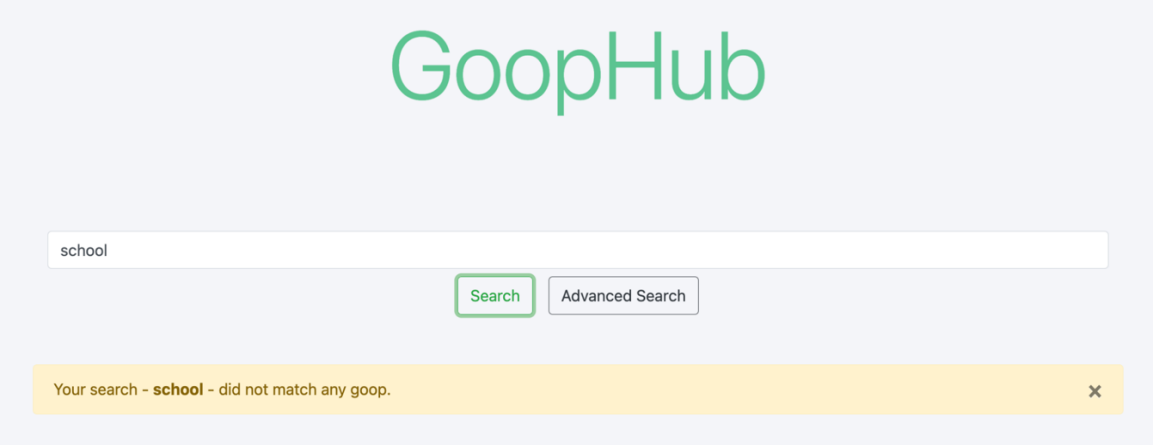

Figura 25 – Resultado da busca sem o aprimoramento.

Com a utilização do subsistema de busca aprimorado, aplicando o modelo BERT, foi inserida a palavra *"school"* no campo de busca e foram retornados os seis GOOPs mostrados na Figura 28.

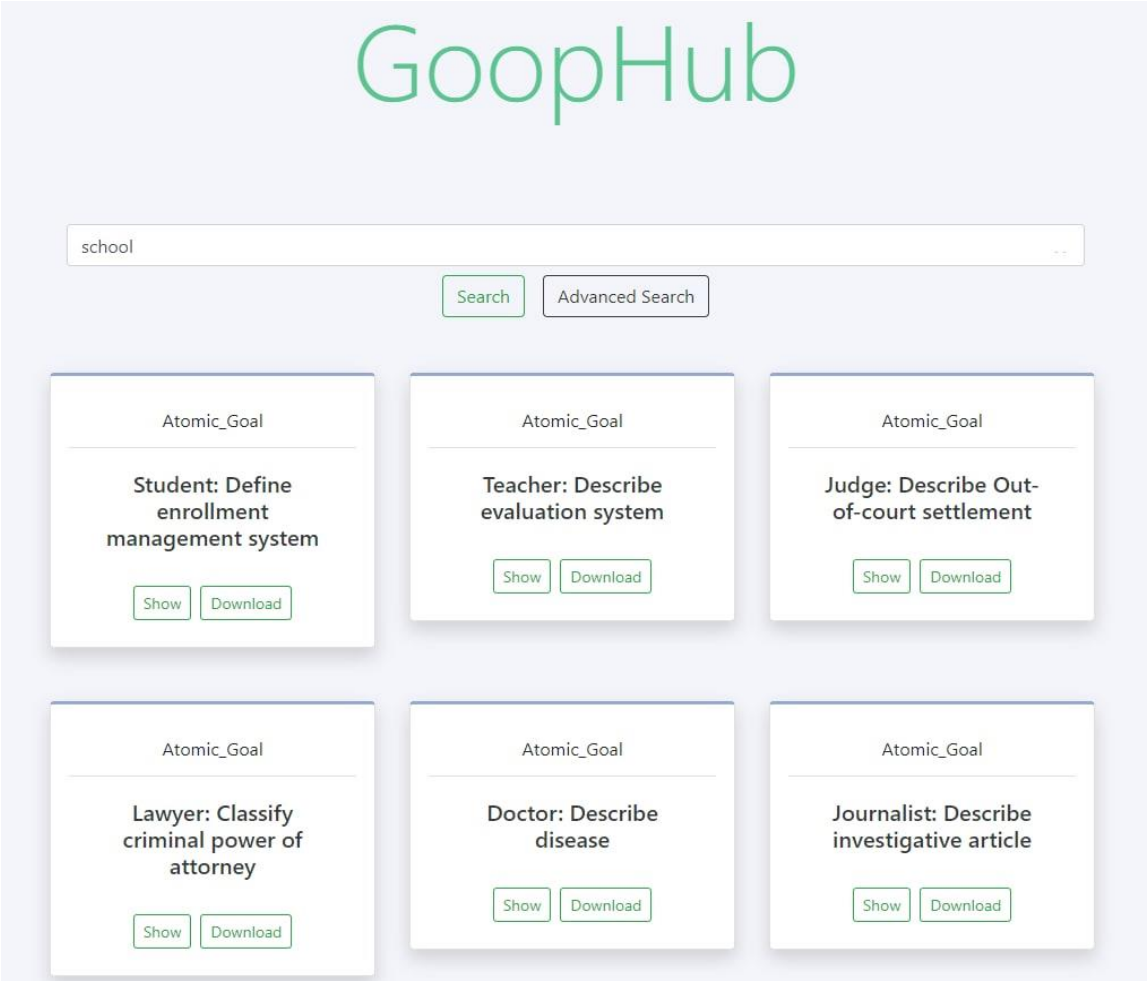

Figura 26 – Resultado da busca com o aprimoramento realizado neste trabalho.

Conforme mostrado na Seção 3.4.1.3, o modelo foi orientado a retornar os dez GOOPs mais próximos semanticamente dentro do modelo BERT, com uma distância semântica de no máximo 0,02 u.m. Essa distância foi arbitrária e escolhida de acordo com a observação dos resultados retornados pelo algoritmo. Foram utilizados os comandos da Figura 29 para traçar um gráfico de distância semântica entre a query inserida pelo usuário e o nome dos GOOPs no banco de dados.

```
1 import pandas as pd
2
3 pd.options.plotting.backend = "plotly"
4
5 df = pd.DataFrame({'id': ids, 'distance': distances, 'title': titles})
6
7 df.plot.line(x='title', y='distance', title="QUERY: " + query_str)
```

Figura 27 - Código para geração de gráfico de distância semântica.

A distância semântica entre a busca e os GOOPs pode ser observada na Figura 30.

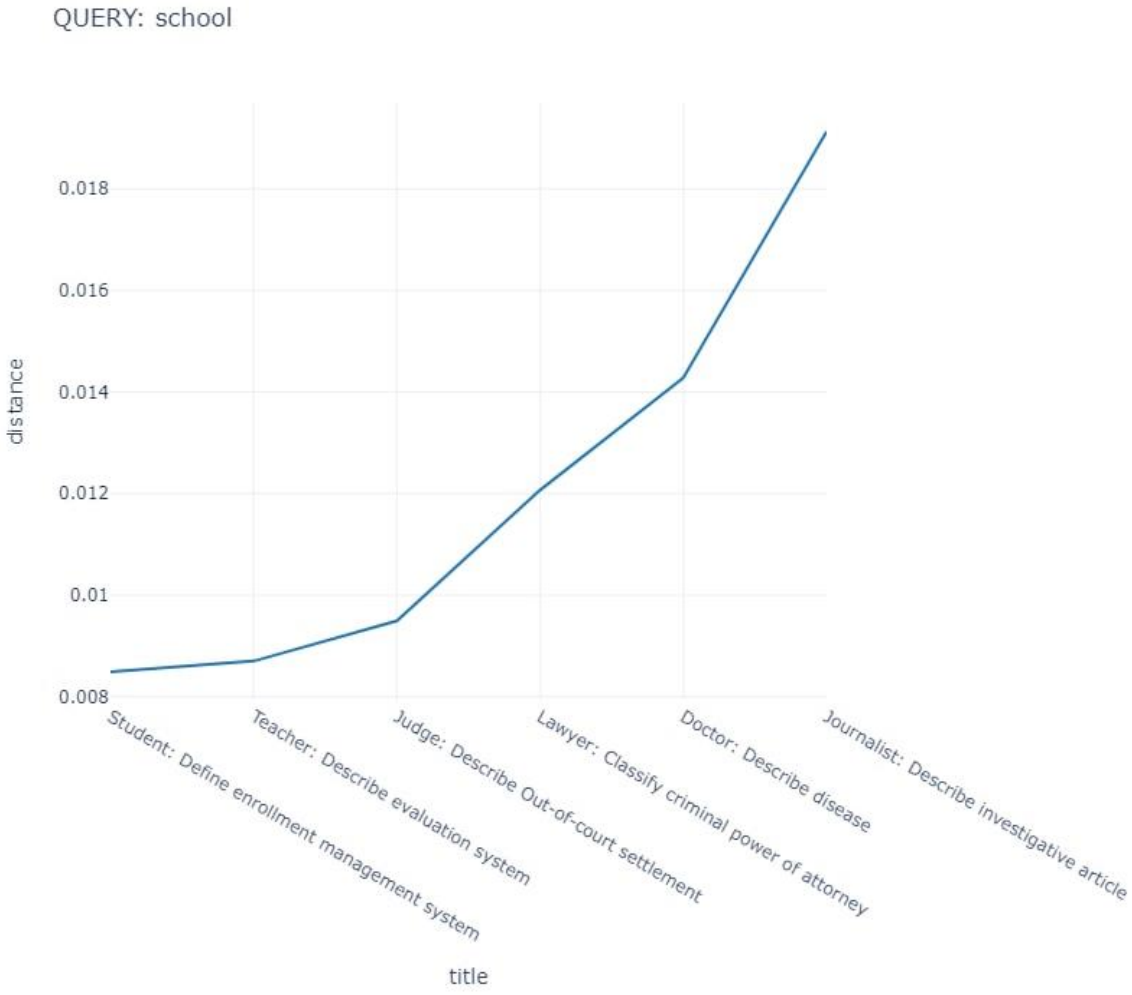

Figura 28 - Gráfico de distância semântica.

O BERT analisou a distância semântica das frases, baseado em seu pré-treinamento, e retornou os seis GOOPs listados na Figura 30, pois eles obtiveram uma distância semântica dentro do limite definido, ordenados de forma crescente pela distância semântica. A similaridade desejada entre duas sentenças pode variar de acordo com a finalidade da aplicação, não havendo um valor padrão a ser utilizado. Para este trabalho, foi feita análise inserindo-se GOOPs de contextos similares e analisando-se qual era a distância calculada pelo modelo entre o texto buscado e o título do GOOP. Foi observado que até a distância de 0.02 ainda eram retornados GOOPs de contexto similar ao buscado. Assim, esse valor foi definido como ponto de corte. No entanto, com a eventual inserção de um grande volume de GOOPs em GoopHub, e com um estudo mais aprofundado da utilidade dos GOOPs retornados para o usuário, pode-se fazer um ajuste fino nesse valor.

É possível verificar-se que sem o aprimoramento realizado, a busca é limitada a retornar apenas GOOPs que atendem literalmente aos termos dados como entrada pelo usuário. Isso pode ser positivo para encontrar GOOPs alinhados aos termos informados. Por outro lado, reduz significativamente a quantidade de GOOPs retornados, excluindo do resultado da busca GOOPs que poderiam ser úteis mesmo tendo nomes ligeiramente diferentes e que podem, inclusive, serem sinônimos. Ao utilizar a busca aprimorada, a busca retorna mais resultados e, com isso, o engenheiro de ontologias tem a oportunidade de analisar mais GOOPs para verificar qual é o mais adequado à sua necessidade. Embora a busca aprimorada traga benefícios, também deve-se considerar que ela pode trazer GOOPs desalinhados ao objetivo do engenheiro de ontologias, o que pode requerer dele maior esforço para identificar, dentre os GOOPs selecionados, qual realmente é útil para sua necessidade. Pensando em amenizar isso, exibe-se o retorno do *fuzzy search* primeiro, dessa forma caso haja algum GOOP cadastrado que tenha termos muito semelhantes ao que o engenheiro de ontologias procura, este terá uma visualização de destaque.

Capítulo 4

Conclusão

Neste *capítulo são realizadas as considerações finais deste trabalho, sendo apresentadas suas principais contribuições e perspectivas de trabalhos futuros.*

4.1 - Considerações Finais

GO-FOR (*Goal-Oriented Framework for Ontology Reuse*) foi proposto com o objetivo de facilitar e apoiar o reúso de ontologias. *GoopHub* promove o uso de GO-FOR apoiando o armazenamento, busca e recuperação de GOOPs (*Goal-Oriented Ontology Patters*). Uma vez que *GoopHub* esteja povoado, reúso poderá ser prática mais frequente, gerando um ciclo virtuoso no desenvolvimento de ontologias com e para reúso (REGINATO et al., 2019). *GoopHub* é uma ferramenta em desenvolvimento, dessa forma, é muito importante identificar nela pontos de aprimoramento, que possam torná-la mais eficaz e contribuir para seu uso e de GO-FOR.

No escopo deste trabalho, foram realizadas duas modificações principais em *GoopHub*: (i) a implementação da estrutura de nomeação de GOOPs proposta em (REGINATO et al., 2022) e (ii) a implantação de algoritmos de NLP para fornecer contexto à busca por títulos de GOOPs. Foi identificada a necessidade de tais refinamentos visto que a nomeação e o mecanismo de busca interferem diretamente em um dos principais objetivos de *GoopHub,* que é a facilitar a recuperação de ontologias para reúso.

Para o desenvolvimento do trabalho, foram realizadas atividades do processo de desenvolvimento de *software,* envolvendo projeto, implementação, testes e implantação. Para a etapa de implementação foi importante estudar as tecnologias que seriam usadas na modificação da aplicação, sendo necessário estudar *frameworks*, bancos de dados, entre outras tecnologias. O aprendizado na área de processamento de linguagem natural, desenvolvimento e reúso de ontologias também foi importante para execução do projeto.

Entre as dificuldades encontradas, destaca-se a compreensão e implementação do algoritmo BERT, pois é um algoritmo relativamente complexo que possui algumas variáveis que precisam ser ajustadas para que funcione bem em cada caso, e o entendimento do código pré-existente na ferramenta para alteração, visto que continha pouca documentação.

No Capítulo 1 foram identificados os objetivos a serem alcançados neste trabalho. O objetivo geral do trabalho foi *aperfeiçoar a busca por padrões ontológicos orientados a objetivos (GOOPs) em GoopHub através da implementação da estrutura gramatical proposta em (REGINATO et al., 2022) para padronizar sua nomeação e de estratégias para realizar busca por proximidade semântica*. Na Tabela 1 é apresentado para cada objetivo específico o produto que serve como evidência de seu alcance.

Tabela 1 – Objetivos específicos e sua situação na conclusão da monografia

Objetivo	**Status**	**Resultado**
Implementar em *GoopHub* a estrutura gramatical proposta em (REGINATO et al., 2022).	Atendido	Implementação da estrutura gramatical proposta (vide Seção 3.5).
Determinar as técnicas de busca automatizada de GOOPs a serem utilizadas.	Atendido	Pesquisa e seleção das técnicas a serem implementadas (vide Seções 2.4 e 2.5).
Implementar a solução proposta de busca no *GoopHub*.	Atendido	Implementação da solução proposta (vide Seções 3.4 e 3.5).
Realizar testes para verificar os resultados das melhorias realizadas no *GoopHub*.	Atendido	Realização de testes, comparando os resultados obtidos antes e depois das modificações realizadas (vide Seção 3.6).

4.2 - Contribuições e Trabalhos Futuros

A principal contribuição deste trabalho é o aprimoramento da seleção de padrões na ferramenta G*oopHub,* que apoia a utilização de GO-FOR fornecendo um repositório para armazenamento, busca e recuperação de GOOPs. A busca por trechos de nomes de GOOPs em G*oopHub* mostrou que a recuperação de GOOPs está mais eficiente, analisando-se não somente cada palavra contida no nome dos GOOPs, mas mostrando-se também efetiva apesar de erros ou variações gramaticais, e analisando-se o contexto e o valor semântico das sentenças pesquisadas.

Como trabalhos futuros, identificou-se que é necessário povoar o repositório com mais GOOPs. Além disso, associar propriedades aos GOOPs como taxa de reúso, avaliação de qualidade pelo usuário, dentre outras, pode ser importante para auxiliar a ranquear os GOOPs, utilizando-se esses resultados alinhados à proximidade semântica. Outro trabalho futuro diz respeito à inclusão de funcionalidades para suportar a integração de GOOPs, com o objetivo de gerar novos GOOPs e auxiliar o engenheiro de ontologias na integração destes em sua ontologia.

Por fim, um trabalho futuro que merece destaque é a implementação (ou integração) de um editor de ontologias em *GoopHub* para facilitar a visualização dos fragmentos e gerar os arquivos das ontologias modeladas utilizando a integração de diversos GOOPs.

De modo geral, a perspectiva é que *GoopHub* seja evoluído para ser adotado por engenheiros de ontologias que tenham interesse em adotar uma abordagem orientada a objetivos. Espera-se que o apoio automatizado provido pela ferramenta contribua para difundir o desenvolvimento de ontologias com e para reúso, melhorando a qualidade e a produtividade do processo de engenharia de ontologias.

Referências Bibliográficas

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software architecture in practice. [S.l.]: Addison-Wesley Professional, 2003.

BONTAS, E. P.; MOCHOL, M.; TOLKSDORF, R. Case studies on ontology reuse. In: CITESEER. Proceedings of the IKNOW05 International Conference on Knowledge Management. [S.l.], 2005. v. 74, p. 345.

BUSTINCE, H.; BARRENECHEA, E.; PAGOLA, M. Restricted equivalence functions. Fuzzy Sets and Systems, Elsevier, v. 157, n. 17, p. 2333–2346, 2006.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research. IEEE Computational intelligence magazine, IEEE, v. 9, n. 2, p. 48–57, 2014.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

DYER, M. G. Connectionist natural language processing: A status report. In: Computational architectures integrating neural and symbolic processes. [S.l.]: Springer, 1995. p. 389–429.

FALBO, R. Knowledge integration in a software development environment. Doctoral Thesis, COPPE/UFRJ, 1998.

FALBO, R. de A. Sabio: Systematic approach for building ontologies. In: ONTO. COM/ODISE@ FOIS. [S.l.: s.n.], 2014.

FALBO, R. de A. et al. Organizing ontology design patterns as ontology pattern languages. In: SPRINGER. Extended Semantic Web Conference. [S.l.], 2013. p. 61–75.

FERNANDES, P. C. B.; GUIZZARDI, R. S.; GUIZZARDI, G. Using goal modeling to capture competency questions in ontology-based systems. Journal of Information and Data Management, v. 2, n. 3, p. 527–527, 2011.

GANGEMI, A.; PRESUTTI, V. Ontology design patterns. In: Handbook on ontologies. [S.l.]: Springer, 2009. p. 221–243.

GHOSE, A.; IPEIROTIS, P. G.; LI, B. Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content. Marketing Science, INFORMS, v. 31, n. 3, p. 493–520, 2012.

GRUBER, T. R. The role of common ontology in achieving sharable, reusable knowledge bases. Kr, v. 91, p. 601–602, 1991.

GUARINO, N. Understanding, building and using ontologies. International journal of human-computer studies, Elsevier, v. 46, n. 2-3, p. 293–310, 1997.

GUARINO, N. Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy. [S.l.]: IOS press, 1998. v. 46.

GUIMARÃES, R. C. M. Nomeação de elementos ontológicos para criação de ontologias: uma proposta metodológica. 2015.

GUIZZARDI, G. On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. Frontiers in artificial intelligence and applications, IOS Press, v. 155, p. 18, 2007.

HAMBORG, F. et al. Newsmtsc: a dataset for (multi-) target-dependent sentiment classification in political news articles. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (ACL). [S.l.], 2021.

HORKOFF, J. et al. Goal-oriented requirements engineering: an extended systematic mapping study. Requirements Engineering, Springer, v. 24, n. 2, p. 133–160, 2019.

HOWARD, N.; CAMBRIA, E. Intention awareness: improving upon situation awareness in human-centric environments. Human-centric Computing and Information Sciences, SpringerOpen, v. 3, n. 1, p. 1–17, 2013.

JACKSON, M. Software Requirements & Specifications: a lexicon of practice, principles and prejudices. [S.l.]: ACM Press/Addison-Wesley Publishing Co., 1995.

JARCZYK, A. P.; LÖFFLER, P.; SHIPMAN, F. M. Design rationale for software engineering: a survey. In: CITESEER. Proceedings of the Hawaii International Conference on System Sciences. [S.l.], 1992. v. 25, p. 577–577.

KANG, Y. et al. Natural language processing (nlp) in management research: A literature review. Journal of Management Analytics, Taylor & Francis, v. 7, n. 2, p. 139–172, 2020.

KATSUMI, M.; GRÜNINGER, M. What is ontology reuse? In: FOIS. [S.l.: s.n.], 2016. p. 9–22.

LAMSWEERDE, A. V. Goal-oriented requirements engineering: A guided tour. In: IEEE. Proceedings fifth ieee international symposium on requirements engineering. [S.l.], 2001. p. 249–262.

MANNING, C.; SCHUTZE, H. Foundations of statistical natural language processing. [S.l.]: MIT press, 1999.

MENDONCA, F. M.; SOARES, A. L. Construindo ontologias com a metodologia ontoforinfoscience: uma abordagem detalhada das atividades do desenvolvimento ontológico. Ciência da Informação, v. 46, n. 1, 2017.

MINSKY, M. Semantic information process. [S.l.]: MIT Press, 1968.

NOGUEIRA, G. G. Um Repositório para Reúso de Ontologias Orientado a Objetivos, Projeto de Graduação, Colegiado do Curso Ciência da Computação, Universidade Federal do Espírito Santo, Vitória-ES, Brasil, 2019.

NOY, N. F. Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record, ACM New York, NY, USA, v. 33, n. 4, p. 65–70, 2004.

NOY, N. F.; MUSEN, M. A. Smart: Automated support for ontology merging and alignment. In: CITESEER. Proc. of the 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW'99), Banf, Canada. [S.l.], 1999.

NOY, N. F.; MUSEN, M. A. The prompt suite: interactive tools for ontology merging and mapping. International journal of human-computer studies, Elsevier, v. 59, n. 6, p. 983–1024, 2003.

PARK, J.; OH, S.; AHN, J. Ontology selection ranking model for knowledge reuse. Expert Systems with Applications, Elsevier, v. 38, n. 5, p. 5133–5144, 2011.

PINTO, H. S.; GÓMEZ-PÉREZ, A.; MARTINS, J. P. Some issues on ontology integration. In: CITESEER. Proceedings of the IJCAI. [S.l.], 1999. v. 99, p. 7–1.

POVEDA-VILLALÓN, M.; SUÁREZ-FIGUEROA, M. C.; GÓMEZ-PÉREZ, A. Reusing ontology design patterns in a context ontology network. Proc. WOP, p. 35–49, 2010.

PRESUTTI, V. et al. extreme design with content ontology design patterns. In: Proc. Workshop on Ontology Patterns. [S.l.: s.n.], 2009. p. 83–97.

REGINATO, C. et al. Go-for: A goal-oriented framework for ontology reuse. In: IEEE. 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI). [S.l.], 2019. p. 99–106.

REGINATO, C. C. ; SALAMON, J. S. ; NOGUEIRA, G. G. ; BARCELLOS, M. P. ; SOUZA, V. E. S. ; MONTEIRO, M. E. ; GUIZZARDI, R. S. S. . A Goal-Oriented Framework for Ontology Reuse. Applied Ontology, 2022. In press.

RESHAMWALA, A.; MISHRA, D.; PAWAR, P. Review on natural language processing. IRACST Engineering Science and Technology: An International Journal (ESTIJ), v. 3, n. 1, p. 113–116, 2013.

SALAMON, J. S. Uma abordagem orientada a objetivos para desenvolvimento de ontologias baseado em integração. 2018.

SALTON, G.; MCGILL, M. J. Introduction to modern information retrieval. [S.l.]: mcgraw-hill, 1983.

SCHANK, R. C. Conceptual information processing. [S.l.]: Elsevier, 2014. v. 3.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge engineering: Principles and methods. Data & knowledge engineering, Elsevier, v. 25, n. 1-2, p. 161–197, 1998.

SUÁREZ-FIGUEROA, M. C. et al. Introduction: Ontology engineering in a networked world. In: Ontology engineering in a networked world. [S.l.]: Springer, 2012. p. 1–6.

WANG, Z. et al. Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence. [S.l.: s.n.], 2014. v. 28, n. 1.

YAMAGUCHI, A. et al. Frustratingly simple pretraining alternatives to masked language modeling. arXiv preprint arXiv:2109.01819, 2021.