Universiteit Utrecht

TNO

UTRECHT UNIVERSITY

Department of Information and Computing Science

master Artificial Intelligence

Exploring the utility of large language models for achieving semantic interoperability in data ecosystems

Author:
Levi van Hees
Student number: 6767508

UU supervisors:
Dr. Marijn Schraagen
Dr. Dong Nguyen

TNO supervisors:
Wouter van den Berg, MSc.
Cornelis Bouter, MSc.

September 27, 2024

Abstract

This thesis investigates the challenges of data interoperability in data sharing, highlighting the difficulty of aligning non-ontological data to ontologies used in data exchange. This issue causes businesses and organizations to struggle to share vital data with each other. This thesis aims to reduce the challenges of data interoperability by using language models (LMs) to align non-ontological data and ontologies, automating a step in the process of data interoperability and easing participation in data sharing. To accomplish this, the thesis looks at the effectiveness of LMs when directly aligning the non-ontological data to ontologies. The data used in this thesis comes from the ENERSHARE project, a European project on data interoperability in different fields of the energy domain, and is aligned to ontologies from the Semantic TreeHouse (STH), a vocabulary hub made by TNO containing various ontologies describing the energy domain. For this task the GPT-3.5 and GPT-4 models are used. The models receive a prompt describing the task, and are fine-tuned on some examples to improve performance. Then to analyze the results of the models, the resulting predictions were analyzed on three categories: target ontology, label structure and pilot of the labels. The results show that the method is not yet ready to be used in the STH but it does show that the method can work and can be improved. And with improvements of the prompt and a change in the evaluation method the current model might perform well enough to be used as a suggestion tool instead of a mapping tool.

Contents

1. Introduction

This chapter will discuss the problem of data interoperability and the different levels of interoperability that can be distinguished. Section 1.1 discusses the problem and its relevance to artificial intelligence. Section 1.1.1 will introduce TNO and the Semantic Treehouse, a vocabulary hub that aims to improve interoperability by reusing existing ontologies. Lastly, in section 1.2 I will introduce the research questions this thesis explores.

1.1 Background

In today's connected and data-driven world, maintaining smooth data interoperability across multiple systems and platforms remains a big problem. Causes for the lack of interoperability stem from the constant development of different data sources, these sources use different formats creating difficulties when exchanging this data. Projects tackling these difficulties (Blavin et al., 2022; IDSA, 2024) aim to provide manual data mapping, transformation, and reconciliation operations to projects that are time-consuming, error-prone, and resource costly. Furthermore, a lack of semantic interoperability, in which data meanings and relationships are not explicitly specified and understood, complicates the issue. As a result, organizations struggle to fully leverage their data assets for decision-making, analytics, and innovation. Part of this struggle comes from the time investment it takes to properly translate the data to the right format. Which leads to the problem that this thesis aims to solve, can language models be used to create accurate mappings between data and ontologies creating a shared semantic data space. Language Models (LMs) offer an exciting possibility for improving data interoperability, with the capabilities of LMs we can attempt to automate semantic understanding, mapping, and integration of heterogeneous data sources, allowing for seamless data exchange and interoperability across multiple systems and domains. LMs are powerful tools that are able to make accurate estimations on the semantic meaning of phrases, which is a crucial part for the task of creating a mapping.

First we must understand what data interoperability is. According to Pagano et al. (2013), there are multiple different data interoperability levels. On an organizational level, data interoperability means that a business' goals and processes operate on every piece of data involved in the data exchange. On the technical level, it means the diversity of technology that supports the data exchange; and on a semantic level, data interoperability involves the understanding of the exchanged data including its contextual information. This thesis aims to offer a method for improving the technical and semantic level of interoperability, by creating mappings between data formats allowing organizations to easily exchange data and attain the same meaning behind their data.

The current European data strategy focuses on the creation of a single market for data (European Commission, 2019). This strategy focuses mostly on the technical and semantic levels of interoperability, but it also promotes organizational interoperability through encouragement of collaboration of public and private stakeholders. Data is supposed to flow freely within the EU and across sectors through the creation of so-called data spaces. According to the Data Space Support Center (Robles et al., 2023), a data space is a framework that supports data sharing within a data ecosystem, a network that allows collaboration between independent users and organizations. Via these data spaces it is possible to share data with other organizations, while remaining in control over your own data. It provides a clear structure for participants to share, trade, and collaborate on data assets in a way that is compliant with relevant laws and regulations and ensures fair treatment for all involved.

For more advanced levels of platform interoperability, a shared semantic space is needed. This shared space can function as a joint vocabulary into which any specific concepts and terminology can be mapped. Rehm et al. (2020) wrote that a first step toward interoperability can be achieved by mapping two schemes onto each other and creating a converter. However, this method creates a new problem, for each new platform that is added to this shared space a new mapping needs to be created. In summary, such a method does not scale. An alternative solution could be for each platform to add a joint ontology for their semantic metadata, but even for general categories communities tend to use different terms for concepts. Thus the creation of such a joint ontology is a difficult task (Labropoulou et al., 2020).

1.1.1 TNO and the Semantic Treehouse

TNO (Nederlandse organisatie voor toegepast natuurwetenschappelijk onderzoek) translates to Dutch organization for applied scientific research in English. TNO is an independent and non-profit organization that through its research aims to create a safer, healthier, and more sustainable life. Or as the website of TNO (2023) states, "TNO’s mission is to create impactful innovations for the sustainable wellbeing and prosperity of society."

Currently TNO has constructed a vocabulary hub called the Semantic Treehouse (STH) (van den Berg et al., 2022), which can be seen as a data provider: it offers a wide range of templates which can be used to represent data. To facilitate semantic interoperability, a wizard-like component was integrated in the vocabulary hub to further drive adoption of shared vocabularies. The wizard takes a user through a series of steps to design message schemata and API specifications based on a shared domain ontology. The final goal is to have the wizard create a translation for the user that allows them to translate their data to match with industry standards. The current functionality of the wizard uses a top-down approach, by starting from the ontologies already in the STH a profile is created that can be used to format a users data. The functionality this thesis aims to add is a bottom-up approach, where by looking first at the data of the user a model tries to create a mapping from the data of the user to the relevant concepts of ontologies in the STH.

1.1.2 The issue of semantics

As mentioned above the ability of IT systems to exchange data with unambiguous, shared meaning is called semantic interoperability. TNO tries to improve the semantic interoperability between organizations in Dutch economic sectors (e.g. flexible staffing, logistics or manufacturing industry), but even in these communities where everyone uses similar concepts it takes a lot of effort by everyone involved to use a single ‘language’ when exchanging data. Achieving this interoperability requires community building around a clear vision of interoperability, creating open semantic standards with that community, setting up and running open governance processes for these standards, regular communication about new developments, and constant implementation support services. While there have been European developments in semantic standardization, some with reasonable success, they have not been enough (Campmas et al., 2022; European Commission, 2017).

1.1.3 Problem statement

Meanwhile the field of artificial intelligence has given rise to state-of-the-art technologies such as large language models (LLMs) and other machine learning and NLP techniques. Could these provide an opportunity to change the way the issue of semantics has been approached in the last decades and boost semantic interoperability in data spaces? The problem can therefore be specified as such: how can we harness the power of LLMs and related technologies to foster semantic interoperability in the context of European data spaces? The specific context in which this problem will be explored is that of the ENERSHARE project (ENERSHARE, 2022), which aims to create a data space for the energy domain.

1.1.4 Relevance

The lack of data interoperability is a problem in many fields of research and work, the lack of being able to easily and accurately exchange data between organizations costs a lot of time and resources. Instead of being able to directly exchange data with other organizations, companies have to spend a lot of time and resources to adjust the data they send or receive. According to Brunnermeier and Martin (2002) the U.S. automotive sector spends an estimated one billion dollars extra and car design times are 2 months longer due to lack of interoperability. Similarly, Powell and Alexander (2019) describe that the lack of interoperability leads to redundant, disorganized, disjointed and inaccessible medical information, and that may affect the quality of care provided to patients and waste of financial resources. Solving this problem can save time and money in many varying sectors. AI techniques can help with the problem of interoperability as for example LMs are good at understanding the semantics of a dataset, this helps to identify patterns and overlaps which can be used to create a mapping between datasets. Furthermore AI techniques might be able to overcome eventual pillarization that might occur if individual sectors focus on interoperability within their own sector (Folmer, 2023), pillarized solutions to data interoperability are solutions that are very domain specific and have no effect on the problems in different domains.

1.1.5 Further background

As discussed earlier a main cause for the lack of data interoperability is the variety of ways in which data is formatted and represented, therefore I will give more details on how knowledge is represented in section 2.2. I will discuss several existing methods and models that can assist the task of data interoperability in section 2.3, this section is mostly about different methods for the task of ontology matching. And lastly I will discuss methods to evaluate the ontology matching models in section 2.6.

1.2 Research questions

The main question for this thesis will be:

How can LMs help improve data interoperability on a semantic and technical level, by creating mappings between data structures?

With this question I want to look at how the semantic meaning of data can be exchanged on a technical level which is accessible to businesses and organizations.

Sub-questions/goals:

1. How does a LM used to create mappings between ontologies fit in the current STH architecture?

    - What form of improvement does using a LM for ontology mapping add to the current STH architecture?

    With this question and sub-question I want to see if the model experimented with in this thesis has the potential to be used in a practical application like the STH. And if not what aspects of the model/method could help in future works.

2. Can the use of LMs for ontology/knowledge graph mapping provide a significant improvement over existing ontology mapping methods?

    - Can the same method used for ontology mapping be used to create mappings between data that is not structured like an ontology and ontologies, by extracting ontological structures from the data? This would accommodate real world examples of businesses needing to digitize their data.

    - What steps are made in this thesis to improve or change methods of previous research on ontology matching?

    The aim of this question is simply if the method of this thesis can make improvements in some aspects of the task compared to existing methods. With the sub-questions aiming to investigate if the differences are caused by the differences in method or dataset and what the differences/improvements of the method are.

3. Can the model help with different domains in the STH structure, or is it bound to a specific domain?

    This question is important since the application is most useful when it can cover different domains, this would show that the method is applicable on a different domain and thus there would be no need to create specific models for each new domain that the model needs to be applied to.

2. Related work

This chapter presents more in dept information on the subjects discussed in the first chapter. Section 2.1 gives a more detailed example of how users can interact with the current STH architecture. Section 2.2 goes more in dept on the representation of knowledge through knowledge graphs. Section 2.3 explains what the process of ontology matching entails. Section 2.4 shows how an alignment can be constructed from non-ontological data to an ontology. Section 2.5 discusses different methods for aligning two ontologies. Section 2.7 shows how ontology alignment methods have been evaluated and how the methods perform. And lastly, section 2.8 looks at the different architectures in recent language models.

2.1 Semantic Treehouse

The Semantic Treehouse (STH) is a vocabulary hub, which means that it stores vocabularies and data standards, the STH can be used for the collaborative governance of the vocabularies. The first version of the STH was released on January 1st 2018. The main principle behind the STH is to reuse as much of an existing ontology as possible. The current STH architecture allows users to choose multiple interaction approaches. The first is a top-down approach, where the user creates an application profile. An application profile is a set of rules the data format should adhere to (for example what is required data or how many times certain label can occur), from an ontology that already exists within the STH. This profile allows the user to specify which parts of the ontologies in the STH are reused for their use case (van den Berg et al., 2022). The second approach is bottom-up and lets the user use their data to create an ontology based on some structure already present in the STH, see figure 2.1. In the first approach, users get assistance from a wizard tool that allows the user to pick relevant classes and extends existing application profiles in the STH. The second approach is more difficult for laymen as it is done by hand, it requires a good understanding of the structure and requirements of an ontology. This is where a model that creates a mapping for the user will truly help. This model can serve as a tool for users, increasing the ease to participate in data sharing and with it the willingness of businesses to exchange their data.

The Semantic Treehouse environment is applied to different domains, for example STPE (Standaardisatie platform e-facturatie) containing ontologies for virtual invoices or Energy contain ontologies from the energy domain, including the ENERSHARE project. Within the STH environment, these ontologies can be viewed in a tree or a graph view for better visualization of the ontologies. The STH also includes several references to external ontologies.

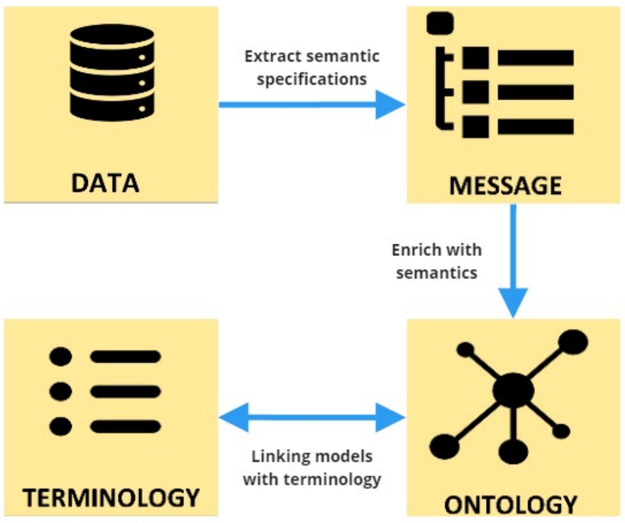

DATA
Extract semantic specifications
MESSAGE
Enrich with semantics
TERMINOLOGY
Linking models with terminology
ONTOLOGY

Figure 2.1: Semantic Treehouse Architecture

2.2 Knowledge representation

This section will take a look at how knowledge can be represented and specifically knowledge graphs, a hierarchical data structure which is used in for example RDF. The field of knowledge representation aims to represent information about a subject in forms that allow computers and machines to solve complex tasks. The field incorporates aspects of psychology on how humans represent knowledge and tackle problems to create conventions or formalisms, this helps to improve the construction of complex systems. Knowledge representation can also use logic to infer additional information from the knowledge (Schank & Abelson, 1977).

Knowledge is often represented in a graph like structure where nodes and edges, in the graph, represent objects and their relations. This type of structure is called a knowledge graph or sometimes a semantic web, and it can encode the underlying meaning relations and semantics of entities (Ehrlinger & Wöß, 2016; Hogan et al., 2021; Noy et al., 2019; Ontotext, 2020). An example of a knowledge graph can be found in below, figure 2.2. Within a knowledge graph the graph structure can be deconstructed into individual links sometimes called facts. Each fact consists of a triple representing the relation between two nodes, each triple can be denoted as (h,r,t) $\in$ F (read as (head, relation, tail))(Ji et al., 2021; Rowland et al., 2022). The triple representation of facts allow for simple interpretation of the facts, but also make manipulation of the graph a difficult task. To solve this, recent research by Q. Wang et al. (2017), focuses on the embedding of knowledge graphs into vectors which allows for easier information retrieval and processing of semantic information.

As mentioned above, a knowledge graph can be represented by a set of triples, but this symbolic way of representing knowledge can make manipulating a knowledge graph a difficult task. When talking about knowledge graphs or other forms of knowledge representation the representation space or embedding space is a space which allows for the comparison of objects with other objects, through this embedding space it is possible to make claims about the similarity of objects. Most studies use real-valued pointwise spaces (e.g. matrix, tensor-spaces and complex vector spaces) as an embedding space, while Gaussian spaces and manifold spaces are used as well (Ji et al., 2021). With these different embedding spaces come a variety of similarity functions, Euclidean distance, TransE and Gaussian embeddings are among them. Recent research on knowledge graphs focuses on knowledge representation learning (KRL) or knowledge graph embedding (KGE), mapping objects and relations into vectors while maintaining their semantic meaning (Q. Wang et al., 2017). The main idea of KRL or KGE is to embed components of the knowledge base: objects and relations; into continuous vector spaces, to simplify the ability to manipulate the knowledge graph while maintaining its structure. A key issue of graph embedding is how to learn low-dimensional distributed embeddings of objects and relations

An example of practical use of knowledge graphs is RDF. RDF, or resource description framework, is a data exchange standard for the Web. RDF implements URIs, uniform resource identifiers, to name the relationship between entities using the head and tail of a link, using this it expands on the linked structure of the Web. It is important that the URIs are unique since if two resources refer to the same URI it would cause ambiguity in the RDF graph. This model allows for interchangeable use of both structured and semi-structured data. This linked structure can be viewed as directed, labeled knowledge graph. The graph view is easy to understand and is therefore ideal to use as an explanation tool. To put this into context, OWL (Web Ontology Language) is a standard for defining ontologies by using the RDF structure as a base. The OWL format is used in many of the existing models for ontology matching for both input ontologies and output mappings.

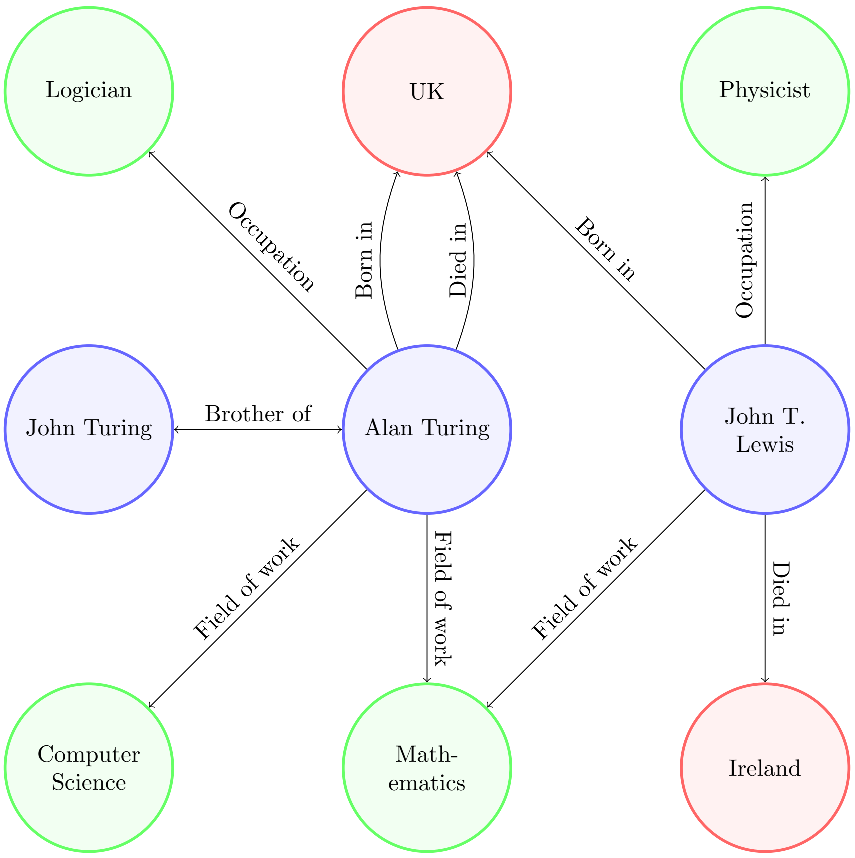


Figure 2.2: Example of a knowledge-graph

Figure 2.2 illustrates what a knowledge graph can look like, this particular example is based on the structure Google uses for their knowledge graph (Singhal, 2012). The figure shows how entities (nodes) are linked together through properties they have in common. It shows how an class is not limited to have one relation to another class, and can have different relations (edge) to the same class (for example Alan Turing was born in the United Kingdom and died there). Similarly, it also illustrates that entities can have the same relation with multiple different entities (for example, Alan Turing worked in both the field of computer science and in the field of mathematics).

A function that knowledge graphs offer is the alignment of classes also known as entity alignment. Entity alignment (EA) aims to combine knowledge from different knowledge graphs. Given two entities from different graphs, EA needs to find the alignment set where the two entities hold an equivalence relation $\equiv$. This technique is similar to ontology alignment which will be discussed in the next section.

There are two ways to approach entity alignment, the first is to match the names of nodes in the graphs. This process relies on the semantic similarity of the node names to find a fitting alignment. The second approach is match the graphs structure wise, this entails that a node is matched based on the relationships it has within the graph, and if there is a node in the second graph with similar relations (Fallatah et al., 2022; Gallagher, 2006).

In the paragraphs above it was mentioned that knowledge graphs can be represented in RDF, figure 2.3 illustrates this as figure 2.2 has been translated to RDF.

```
<#alan-turing>
    rel:brotherOf <#john-turing> ;
    ex:occupation ex:logician ;
    ex:fieldOfWork [
        ex:mathematics
        ex:computerScience
    ] ;
    ex:bornIn ex:unitedKingdom ;
    ex:diedIn ex:unitedKingdom .

<#john-turing>
    rel:brotherOf <#alan-turing>.

<#john-t-lewis>
    ex:occupation ex:physicist ;
    ex:fieldOfWork ex:mathematics ;
    ex:bornIn ex:unitedKingdom ;
    ex:diedIn ex:ireland .
```

Figure 2.3: Knowledge graph on Alan Turing in RDF format

2.3 Ontology matching

Solutions to the issues of data interoperability should aim to unify comparable data formats, allowing for these formats to be aligned with each other. When this alignment process occurs between ontologies, it is also known as ontology mapping, and this method has the potential to solve data interoperability issues.

To understand the problem of ontology matching we first need to define what ontologies are, what a mapping or alignment is and how ontology alignment methods create such mappings. Within the field of data and computer science, an ontology is a representation of a shared understanding or a convention of some domain interest (Euzenat et al., 2011; Feilmayr & Wöß, 2016; Uschold & Gruninger, 1996). This representation allows a computer to use abstract concepts, which they otherwise could not work with. Within the literature on ontologies there are many articles that make distinction between types of ontologies. There are often two important categories in which an ontology can fall. There are top- or upper-level ontologies, this type of ontology provides a high-level of abstraction. Upper-level ontologies are ontologies that allow for definitions of concepts that are widely used across different domains (Euzenat & Shvaiko, 2013; Mascardi et al., 2007), an example of an upper-level ontology framework is SUMO (Niles & Pease, 2001).

The second big category in which ontologies can be classified are the task-specific or application ontologies. These ontologies are focused on defining knowledge of a specific domain. They encapsulate concepts, processes and relations needed for specific tasks (Euzenat & Shvaiko, 2013; Sadegh-Zadeh, 2012), an example of application ontologies is semantic web service ontologies (Battle et al., 2005).

The next question is, how we can represent the mapping or alignment we are trying to create. An alignment can be defined as a set of links between classes or properties in an ontology. An example of a formal definition of this link is given by Euzenat and Shvaiko (2013), here a link is defined as the quintuple: $\langle id, e, e', r, n \rangle$. Here the id is a unique identifier for the link, e and e' are the entities of different ontologies between which the link is formed. r expresses the similarity relation between the two entities, and n expresses the confidence of the method used in the link. Figure 2.4 below illustrates what a mapping between two ontologies might look like. This example uses an ontology for human (figure A.2) and person (figure A.3). This mapping was made by looking for equivalence relations between the properties of the ontologies. In this case, the human ontology consists of 4 labels while the person ontology only has 3 labels. A human is defined to have a separate first name and last name, while the person only has a name to accommodate for both of these labels. In a mapping, it is possible to map different labels from the input to a single label of the target, which is how this conflict between the ontologies can be resolved.

A general process for the matching or alignment of ontologies was described by Euzenat and Shvaiko (2013). The goal of ontology matching is to create a mapping or alignment A' between two ontologies o and o'. The process determines whether two classes within the ontologies can be matched by looking at their similarity, for this different similarity measures can be used, e.g. semantic similarity. The more similar two classes are, the more likely it is that these classes will be mapped onto each other. There are also different parameters that can be added to this process to increase effectiveness and efficiency. One is the use of an input alignment A, this is an alignment that already exists between the ontologies and is extended through the mapping process. The second method is the addition of extra matching parameters, for example weights or a threshold for the selection of links between classes. The last method of expanding a matching method is by making use of some external knowledge source that complements the ontologies, this could be some domain-specific thesaurus (Euzenat & Shvaiko, 2013; Euzenat et al., 2011).

Section 2.5 will discuss several existing models for ontology matching.

```
<map>
  <Cell cid='1'>
    <entity1 rdf:resource='http://example.org/human#Human'/>
    <entity2 rdf:resource='http://example.org/person#Person'/>
    <measure rdf:datatype='xsd:float'>1.0</measure>
    <relation>=</relation>
</map>
<map>
  <Cell cid='2'>
    <entity1 rdf:resource='http://example.org/human#firstName'/
    >
    <entity2 rdf:resource='http://example.org/person#hasName'/>
    <measure rdf:datatype='xsd:float'>1.0</measure>
    <relation>=</relation>
</map>
<map>
  <Cell cid='3'>
    <entity1 rdf:resource='http://example.org/human#lastName'/>
    <entity2 rdf:resource='http://example.org/person#hasName'/>
    <measure rdf:datatype='xsd:float'>1.0</measure>
    <relation>=</relation>
</map>
<map>
  <Cell cid='4'>
    <entity1 rdf:resource='http://example.org/human#currentAge'
    />
    <entity2 rdf:resource='http://example.org/person#hasAge'/>
    <measure rdf:datatype='xsd:float'>1.0</measure>
    <relation>=</relation>
</map>
```

Figure 2.4: Mapping between ontology of a human and a person

2.4 Data to ontology alignment

Aligning non-ontological data to an ontology differs a great deal from trying to align two ontologies. Unlike a predefined ontology, non-ontological data lacks explicit categorization and organization, making the alignment process intricate and nuanced. This section explores several methodologies, techniques, and considerations involved in bridging the gap between non-ontological data and ontological structures, aiming to use every piece of information embedded within non-ontological sources for enhanced semantic interoperability and knowledge extraction.

To create an alignment between non-ontological data and an ontology two routes can be considered: the first is to directly create the alignment between the data and the ontology, using a similarity measure to compare the data labels with the labels from the ontology. The second method would involve the construction or transformation of the data to an ontology, from this point the two ontologies can be aligned with methods as described in section 2.5.

The first approach involves the direct alignment of non-ontological data and ontologies (Bousquet et al., 2019; Poggi et al., 2008). There is a difference in how similarity can be measured between data and ontologies and ontologies and other ontologies. The main difference is that ontologies can be seen as graphs, therefore each node or label has a context, which are a node's neighbors. Non-ontological data can have such a context, but it is not always present making ambiguity of a labels name more of a problem. An example of this ambiguity is a label like 'loading time', if this is the only information on the label it could mean two things namely, the time it takes to load some object or the time at which some object was being loaded. This ambiguity present a real problem when trying to precisely align this label to an ontology. Ambiguity is arguably the biggest problem in all NLP tasks, and is easiest to solve by providing more context to a label, as said before a detailed description of each label is not available. In chapter 3 I will explain more about the data and what information is available to the models.

The second approach requires a processing step of extracting the ontology from the data. There are several approaches to this task: Top-down and bottom-up. The top-down approaches focus on reusing existing ontologies, mainly top-level ontologies, to create the coverage that is desired. Reasons to use top-level ontologies are to improve the overall quality of the ontology, by using principled design choices, but also to create interoperability to ontologies that use the same top-level ontologies (Keet, 2020). As mentioned earlier the current wizard tool within the STH architecture works in a top-down fashion. It allows users to reuse an existing ontology from the STH by picking specific elements from the ontology that get translated into an application profile (van den Berg et al., 2022).

The second approach, bottom-up, starts with a blank slate and aims to reuse not existing ontologies but existing data and knowledge (El Ghosh et al., 2017). Methods to accomplish this task range from manual to automated. There are various tools that can learn ontologies from data sources. For example Text2Onto (Cimiano & Völker, 2005) is able to extract terms, synonyms, concepts, taxonomies and even non-taxonomic relations from data source. It does this through linguistic processing, statistical text analysis, machine learning and association rules. By extracting all this information it can construct a hierarchy that is used to build the ontology.

Other methods to assist in this task are ontology recommenders. Ontology recommenders such as, NCBO ontology recommender (Martínez-Romero et al., 2017), rank a text on the basis of the presence of text tokens. The similarity of tokens compared to labels of ontology classes is calculated and informs which ontology might be best suited to map the text tokens to (Korel et al., 2023).

2.5 Ontology alignment

The subject of ontology alignment is getting increasingly popular (see appendix, figure A.1), and new methods to match ontologies are created each year. The following section will summarize some of the prominent methods of the past decade, results of these models on ontology alignment tasks will be discussed in section 2.7.3. Table 2.1 show methods that will be reviewed and which extra parameters are added to each method, these parameters are described in section 2.3.

LOOM (Ghazvinian et al., 2009) is an algorithm for creating mappings between two ontologies represented in OWL, returning pairs of related concepts. LOOM compares names and synonyms of the ontologies, identifying two concepts as similar if their names or synonyms are equivalent based on a modified string-comparison function. This function removes delimiters and compares strings based on a Levenshtein distance of one.

Logmap, proposed by Jiménez-Ruiz and Cuenca Grau (2011), addresses scalability issues in ontology alignment methods. Many alignment methods excel with moderately-sized ontologies but struggle with larger ones.

	Initial mapping	Extra threshold	Extra knowledge
LOOM		✓	
LogMap	✓	✓	
AML		✓	✓
OntoEmma		✓	✓
DeepAlignment		✓	✓
VeeAlign		✓	
Truveta		✓	
Mapper GPT	✓	✓	✓

Table 2.1: Existing models and which extra parameters they incorporate as specified by Euzenat and Shvaiko (2013)

LogMap tackles this by employing optimized data structures for lexical and structural indexing, creating initial mappings between ontologies. These mappings are then improved through a series of iterations using a process of mapping repair and discovery, see figure 2.5.

The repair step resolves logical inconsistencies within anchor mappings, while the discovery step expands contexts for each anchor, leveraging class hierarchies to create new mappings based on similarity scores calculated using ISUB (Stoilos et al., 2005).

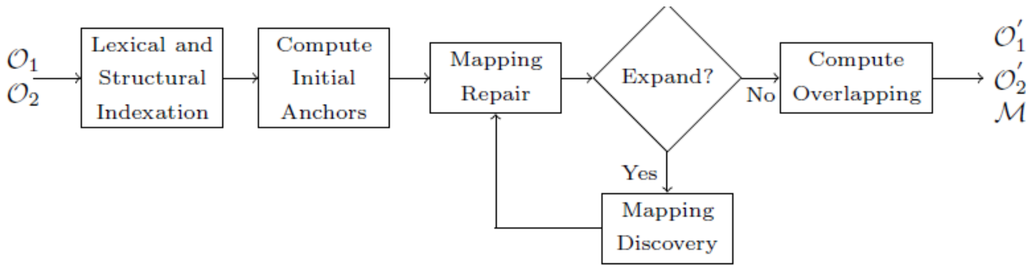


Figure 2.5: Architecture of LogMap (Jiménez-Ruiz & Cuenca Grau, 2011)

AML (Faria et al., 2013), an evolution of AgreementMaker, offers a solution for aligning large ontologies effectively. Its matching module, depicted in figure 2.6, is composed of matchers, selectors, and an alignment data structure. The matchers compare ontologies, selectors refine mappings, and the alignment data structure finalizes mappings. Any matching method can be used as a matcher and selectors trim mappings by excluding parts that fall below a certain similarity threshold.

Iyer et al. (2020) criticized AML for its reliance on handcrafted rules and manually assigned weights, hindering scalability and overlooking semantic relatedness.

OntoEmma, introduced by L. L. Wang et al. (2018), makes use of a neu-

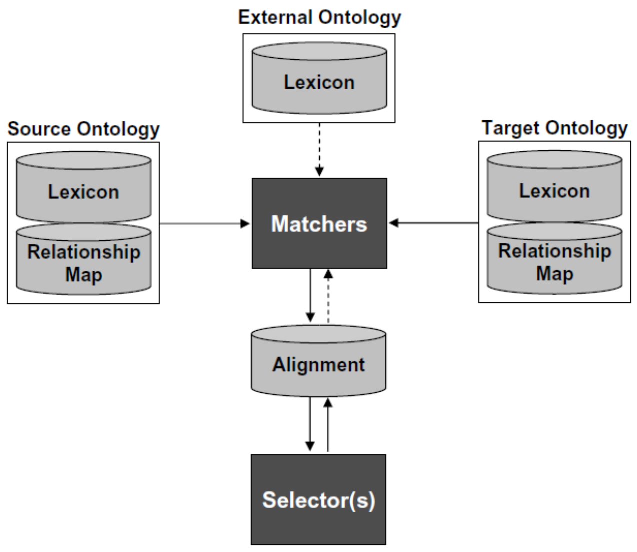


Figure 2.6: Architecture of AML's matcher module (Faria et al., 2013)

ral architecture, capable of encoding additional information when it is available. OntoEmma consists of three stages: candidate selection, feature generation, and prediction. Numerous ontologies have a large amount of classes and properties, this makes it computationally expensive to consider all possible pairs of source and target entities. To reduce the amount of pairs to consider OntoEmma uses the inverse document frequency (idf) of word tokens to select possible candidates that need to be considered during the process. These candidates then form pairs for which a set of features is generated, these features are often measures of similarity between the pairs, features include: Jaccard distance, root word equivalence, and other boolean and probability values.

The last step is to predict the probability of the semantic equivalence of two entities. This is done by first creating an entity embedding, this embedding is created by encoding different parts of the entity separately and concatenating these parts. After getting the embeddings they are fed to two subnetworks, each network is a two layer feed-forward network. The outputs are again concatenated, and then fed to one final feed-forward network, after this the model estimates the equivalence between the pair, which is saved as a link in the mapping.

Figure 2.7 shows the architecture of OntoEmma's Siamese network and the process of entity embedding. The figure shows that the embedding is created by combining the name, aliases, the definition and the context of an entity vector.

Created by Kolyvakis et al. (2018), DeepAlignment is an unsupervised

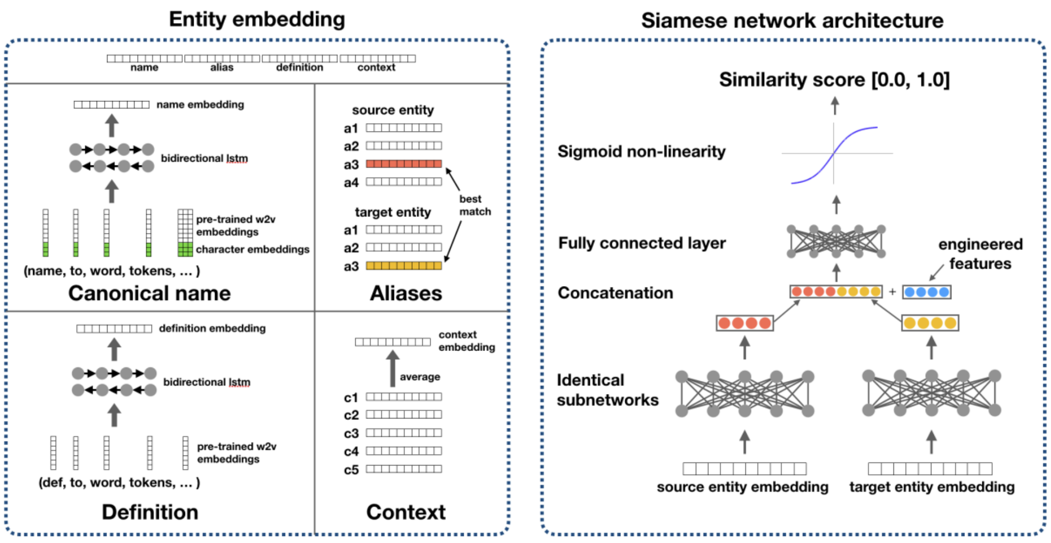


Figure 2.7: Architecture of OntoEmma (L. L. Wang et al., 2018)

deep learning method for ontology matching. The algorithm uses extra knowledge sources to extract synonyms/antonym relations, which are used to refine the pre-trained word vectors. This extra knowledge source is a set of synonyms and antonym relations extracted from semantic lexicons. Each ontological class is represented by a bag-of-words, which is complemented by the refined word embeddings. For the ontology matching the Stable marriage algorithm is used. This algorithm calculates the one-to-one mappings based on pairwise distances of entities. These distances are calculated using a variant of the document similarity metric. This similarity metric calculates the normalized average distance between the word embeddings for the pair.

Iyer et al. (2020) have created a deep learning based method for the task of ontology matching. When Iyer et al. (2020) wrote their paper, deep learning approaches (e.g. DeepAlign) for alignment tasks were often very domain specific and performed worse than rule-based systems. To change this the method called VeeAlign uses a dual-attention mechanism to compute contextualized representations of a class to learn mappings.
VeeAlign makes use of a Siamese network that creates both positive and negative alignment pairs. VeeAlign can make use of the context of a concept, namely the neighboring concepts in the ontology, for a better similarity computation. Figure 2.8 shows the architecture of VeeAlign, focusing on the Siamese network it is visible how the entities and their context pass through each layer to create an embedding. To do this the network first creates individual embeddings for all parts of the context and then from this extracts a single contextualized embedding by combining the entity vector and the weighted sum of the individual embeddings. The authors find that

context plays an important role in creating alignments. Ontologies consist of concepts and the relationships between these concepts. VeeAlign is based on the computation of the representations of both a concept and its context. In VeeAlign this context are the concepts that are connected to the concept in the ontology, and within the context a differentiation is made between: ancestor nodes, child nodes, nodes connected through datatype properties and nodes connected through object properties.

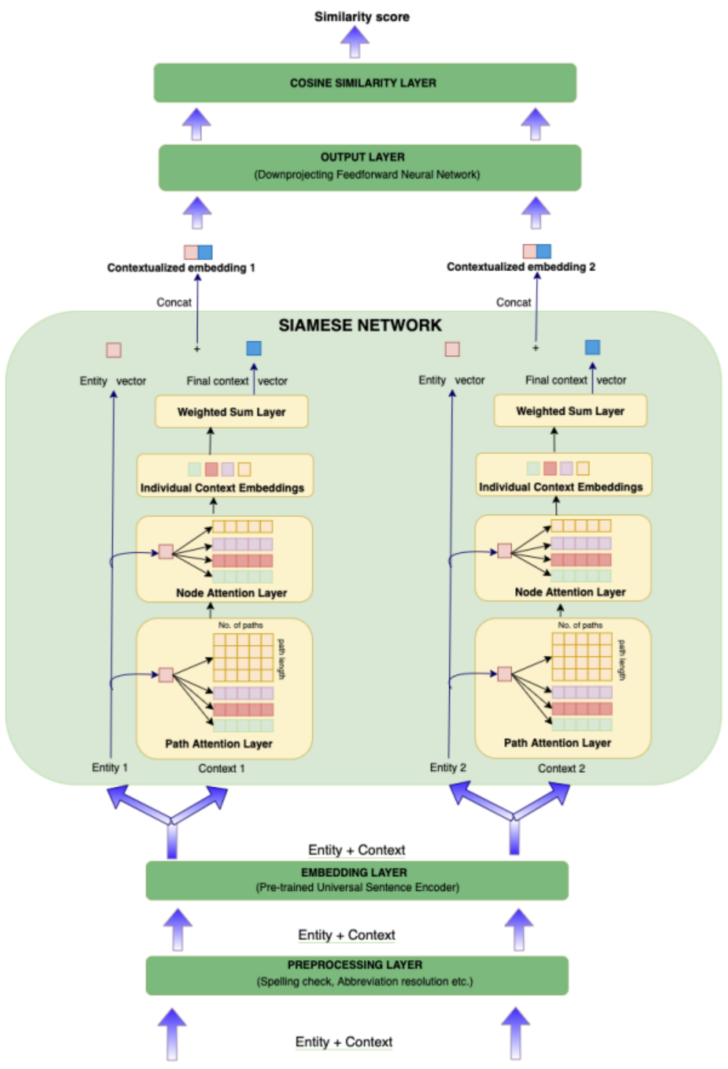


Figure 2.8: Architecture of VeeAlign (Iyer et al., 2020)

Amir et al. (2023) have proposed a model called Truveta mapper for the task of ontology matching. The proposed approach is based on zero-shot learning and prediction, where zero-shot learning refers to the ability of the model to make source-to-target predictions without requiring examples of labelled ontology matching pairs, and zero-shot prediction performs end-to-end mapping from the source to the target without any similarity calculation across the entire/subset target ontology or post-processing like extension/repair. The model is pre-trained to learn the hierarchical structure and semantics of each ontology, figure 2.9 shows the architecture of Truveta's training. Starting from a language model pre-trained, further pre-training is done using masked language models on ontology graphs. The model is then fine-tuned on downstream tasks, translating from the label and its context to the target node path. The pre-training and fine-tuning are done in a multitask manner. The pre-training is performed on both source and target ontologies, and fine-tuning is done on task specific target subset ontologies. The model is then further fine-tuned for down-stream ontology mapping tasks. The Truveta Mapper then has the capability to translate ontologies from an input source, and given this source the model can predict potential candidates in the target ontology.

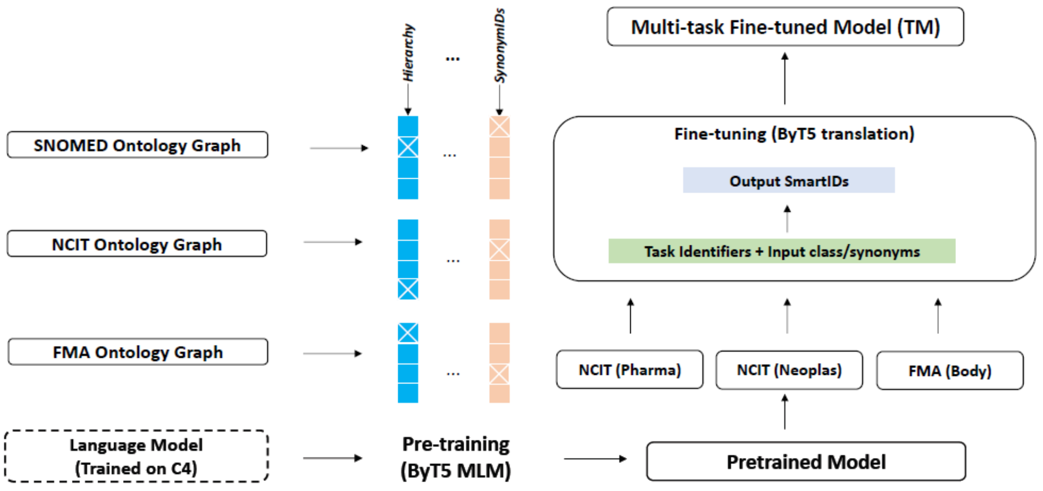


Figure 2.9: Training architecture of Truveta (Amir et al., 2023)

Similar to Logmap, MapperGPT is a method for improving mappings but unlike Logmap MapperGPT relies on external models to generate the initial mappings. It was introduced by Matentzoglu et al. (2023), and uses ChatGPT to compare semantic differences between ontologies. The process involves receiving a set of candidate mappings from a high recall alignment method, for example LexMatch (Mungall, 2022) or LOOM (Ghazvinian et al., 2009), these mappings together with the ontologies are used to create a prompt. The prompt asks ChatGPT to estimate the relationship between two classes of the ontologies, in terms of how similar they are, and asks how confident the model is in the estimation. The model receives the concepts together with a short description of the concept and some relationships this concept has to other classes in the ontology. The estimations are then used to improve the candidate mapping. The method was tested with both the GPT-3.5 and GPT-4.0 models, although GPT-4 proved to have higher precision and recall on almost all tasks it was not a significant difference compared to the results of the GPT-3.5 model.

To summarize, this review of existing methods shows that there are numerous ways to create a mapping between ontologies, but it also shows that these methods often make use of the extra thresholds and extra knowledge as suggested by Euzenat et al. (2011) to make the method more efficient. As most of these models present the model with extra information about the ontologies, it is likely that the model that will be used in this project would also benefit from extra knowledge as an input. Many models also use a similarity measure to help decide which links to make, so using a similarity measure is also recommended. The next section will discuss how to evaluate mapping methods.

2.6 Evaluating alignments

When the model has created a mapping there needs to be a method to evaluate how good the mapping is. The evaluation can be done manually or automatically. With a manual evaluation an expert can be asked to assess the quality of the mapping. The second method is to have a computer run a automated quantitative evaluation, the techniques used in such an evaluation require the computation of measures like recall and precision, and would provide an approximation on the correctness and completeness of the model. This evaluation also requires manual mappings that can be considered as the correct mappings. It is important to use these common performance metrics to be able to easily compare different types of models.

Early efforts for the evaluation of alignment methods by Do et al. (2003) focused mostly on comparison criteria from four areas: which test cases were used, the matching results, quality measures and savings of manual efforts. Unfortunately methods using these four criteria never became widely used.

A second evaluation method was introduced by Sure-Vetter et al. (2004) the measures used in their paper are precision and coverage. Precision is a metric that indicates how well a created mapping fits the intended mapping. Coverage is a metric that measures how much of one ontology covers the second ontology in the mapping, more coverage means that the method finds a mapping that links more classes in both ontologies. Although these measures offer a good measure on the accuracy of a mapping, these measures are also not widely used, and thus do not offer easy comparison between existing models and new models.

Alignment methods often use a real-world scenario to show their effectiveness. The use of specific cases can be an issue as the method will not perform similar when presented with a scenario that is different from the test case. This makes it difficult to evaluate and compare different alignment methods. Duchateau et al. (2007) also identified several useful properties for evaluation methods. The first property is extensibility: the evaluation method should be able to be expanded when new alignment methods are created. Scalability is another important property as this allows for the creation of new scenarios on which the methods are evaluated. The final important property is that the evaluation method should be generic, it needs to work with most available matchers and use performance metrics that apply to all these alignment methods.

Later work by Euzenat et al. (2011), yielded the OAEI or Ontology Alignment Evaluation Initiative an initiative that aims to benchmark methods for ontology matching. The OAEI tests methods by comparing the mappings created by the methods to mappings created by human experts, to test the accuracy and efficiency of these methods. The OAEI has gained significant popularity and acceptance within the ontology alignment community over the years as a result of its thorough evaluation framework, transparent evaluation procedure, and benchmark datasets obtained from real-world applications.

Using the method for evaluation constructed by the OAEI will allow for a simple comparison between the method in this thesis and the established methods of the last decade. However this measure is mainly used to measure pure performance and does not take into account any form of usability measures that may influence the experience for users of the method. The method also does not compare the interpretability of methods, so black-box models could perform well, without an explanation as to how decisions are made.

2.7 Evaluation of ontology matching models

As mentioned in the introduction there have been many efforts to create a fitting method for the evaluation of ontology matching models, however not all of these methods gained much traction. This section will take a deeper look into a prominent method for the evaluation of ontology mappings.

2.7.1 OAEI

According to Euzenat et al. (2011) on an abstract level ontology matching is a process of finding correspondence between two ontologies. The correspondence here expresses the relationship that entities within ontologies hold. An example of this is that *subject area* in one ontology is similar to *topic* in another ontology. The key is that corresponding entities in ontologies have similar relationships within their respective ontologies.

The Ontology Alignment Evaluation Initiative (OAEI) (Euzenat et al., 2011) is a collaborative effort to evaluate and improve methods for aligning ontologies. The OAEI offers an environment for researchers and experts to review and compare ontology alignment methods through a series of annual evaluation campaigns. These campaigns include benchmark datasets, evaluation measures, and defined evaluation processes, allowing participants to thoroughly and systematically evaluate the performance of their ontology matching approaches. The alignment of models are compared to reference alignments constructed by domain experts. As a result, the OAEI has become a popular metric for evaluating ontology matching models, giving useful insights and benchmarks for determining the performance and scalability of ontology alignment methods.

The OAEI specifies two major characteristics that ontologies need to possess for a proper evaluation of matching models. First is the complexity of the labels, matching systems rely on heuristics to compare class labels in ontologies, with performance heavily influenced by label types, especially when differentiating between for example simple labels and sentence-like labels. The ability to anchor labels to background knowledge sources like WordNet also has a big impact on the performance. Complexity increases when ontologies utilize specialized vocabularies, such as those in biomedical or geo-spatial applications, which may diverge from common language.

The second influence is the complexity of structures, matching systems utilize ontology definitions to propagate similarity estimations and validate correspondences, making ontology structures crucial in benchmark dataset design. While RDF and OWL standardize syntax for comparing ontologies, their usage varies widely. Lexicons and thesauri primarily rely on hierarchical structures, while more expressive ontologies incorporate class relations constrained by axioms, enhancing matching and alignment coherence. Instances vary in complexity, from detailed descriptions with attributes and relations to atomic entities lacking explicit definitions. External resources like web pages or images linked to instances can aid matching, with webpages offering richer information for easier comparison compared to more challenging interpretation of images.

Naturally there are also some aspects that influence the evaluation results, that are to be considered when it comes to the reference alignment. One such aspect is the type of semantic relations the alignment uses. As discussed earlier an alignment consists of a set of relations between entities or properties. The type of relations the reference alignment contains reflect the type of relations the model is expected to produce. A common relation used is the equivalence of entities, with most models being designed to produce these rules, but there are exceptions. Other relations that can be used for comparison are subclass and disjoint relations (Guo et al., 2005; Sabou & Gracia, 2008; Van Hage et al., 2005).

In addition to the type of relation, semantics are also an important aspect for a reference alignment. Specifically, we must distinguish between more and less strict interpretations of relations. For example, the equivalence relation can be interpreted as logical equivalence or, more loosely, as a high degree of resemblance or interchangeability. Employing a strict formal interpretation of semantic relations allows for the application of formal properties on the reference alignment. For example, we can claim that the merged model, which includes both ontologies and the alignment, should be coherent, which means it should not contain unsatisfiable classes. Less formal interpretations make it impossible to enforce such consistency conditions.

Euzenat et al. (2011) mention a third but perhaps less obvious aspect, which is the cardinality of a reference alignment. While there are no constraints on the alignment, allowing for an n-to-m relationship between entities from different ontologies, practical observations show that the alignment connection is mostly one-to-one. Consequently, matching systems frequently create one-to-one alignments. Similarly, while the degree of overlap between the ontologies being matched is not specified, and datasets may contain two ontologies with little or no overlap, it is generally assumed that the two ontologies belong to the same domain. As a result, matching algorithms typically attempt to construct an alignment between all elements in the two ontologies rather than ignoring elements.

To ensure models are not over-fitted on domain-specific ontologies but can handle any ontology the OAEI offers a wide variety of ontologies as test cases, ranging from ontologies on anatomy and food nutrition to ontologies describing the domain of organizing conferences. The datasets for these test cases fall in different categories of problems which they represent. One of these categories is expressive ontologies, these datasets represent issues of realism as they are much larger, and have more complex definitions. Lexicons and thesauri are another category, these datasets are weakly constructed but large ontologies, which are currently being used in digital libraries. The lack of sophisticated structure and the size of these datasets is what makes them challenging for most matching models. Finally there are the instance matching and beyond equivalence categories which focus on different relations besides equivalence to find matches, for example exact match and related match are popular relation types for tests.

2.7.2 Extension of the OAEI framework

A different framework for evaluation was proposed by Mohammadi and Rezaei (2020), the framework is based on a set of performance metrics that accommodate experts' preferences using a multi-criteria decision-making (MCDM) method. Determining the expert-based performance or ECP, a survey was done among domain experts of the different OAEI tracks. These experts were asked to rank the performance metrics that Mohammadi and Rezaei (2020) had identified for each track. From the survey the importance of each metric according to the experts was ranked and these weighted metrics form the ECP and alignment models are evaluated with this metric.

The method is similar to the method of the OAEI, and it even uses the datasets of the OAEI as test cases. The main difference however is that this method assigns different performance measures depending on the test case. For each of the test cases the authors assigned performance metrics depending on the data in the test case, this combined with the weighing of the metrics by experts means that each test case is measure with a unique set of metrics.

2.7.3 Results of existing models

This section compares several of the matching models discussed in section 2.5, by looking at the results they achieved during testing (shown in table 2.2).

These results show that most of these models still struggle to create perfect mappings. More importantly by looking at previous results for long time participants such as LogMap and AML, we see that these models have not booked significant improvements over the last couple of years, even despite the regular changes that have been made to the models to improve performance (Faria et al., 2021; Jiménez-Ruiz et al., 2017, 2018). LogMap had a precision of 0.84 in 2018 and AML had a precision of 0.88, on the same

Model	Dataset	Precision	F-measure	Recall
LogMap	SNOMED (2022)	0.81	0.72	0.64
OntoEmma	SNOMED (2018)	0.80	0.61	0.69
AML	SNOMED (2022)	0.69	0.70	0.71
DeepAlignment	Conference (2018)	0.71	0.75	0.80
VeeAlign	Conference (2022)	0.74	0.70	0.66
Truveta	SNOMED (2022)	0.95	0.83	0.74

Table 2.2: Evaluation results of ontology matching models on datasets of the OAEI (Abd Nikooie Pour et al., 2022). The SNOMED dataset is part of the large biomedical track of the OAEI, and conference refers to the conference track. Important to note is that the results of OntoEmma, DeepAlignment and Truveta are reported by the authors and were not submitted to OAEI campaigns.

task within the bio-medical track. These result might stem from the fact that the OAEI changes and refines their evaluation tasks from time to time, but it still shows that these established models have ways to go before they are completely accurate. The results from the Truveta mapper seem more promising, with a precision of 95%, but these results are not confirmed by the OAEI.

Model	Precision	F-measure	Recall
MapperGPT 3.0	0.50	0.49	0.48
MapperGPT 4.0	0.60	0.67	0.76
LogMap	0.46	0.53	0.62
LexMatch	0.21	0.34	0.88

Table 2.3: Evaluation results of MapperGPT model, both GPT 3.0 and GPT 4.0 variants (Matentzoglu et al., 2023). The datasets used in these test are from the biomedical domain but are different from the OAEI sets. Note that the results from LogMap needed to be converted to SSSOM format (Simple Standard for Sharing Ontology mappings) (Matentzoglu et al., 2022), in order to be compared with MapperGPT results.

The results in table 2.3 show that the usage of MapperGPT yields great improvements over the baseline results of LexMatch, especially the version that uses the GPT 4.0 model. This suggests that the method could also yield performance improvements for other mapping methods.

2.8 Language models

Language models are a major part of natural language processing, giving computers the means to understand and generate human language. At their core, LMs are statistical models being able to grasp the patterns and structure of language by learning from large amounts of data (Ahn et al., 2016; Rae et al., 2021). Language models estimate probability distributions of word sequences in a language, giving them the ability to predict the likelihood of a sequence or generate a coherent sequence themselves. With the use of both traditional statistical methods and modern deep learning structures, LMs have grown significantly, causing breakthroughs in a variety of NLP tasks such as machine translation and text generation.

LLMs can be described in the following way: LLMs are large language models that are pre-trained on large amounts of data without being fine-tuned for a specific task (Jurafsky & Martin, 2021). But even without fine-tuning, LLMs are still able to perform a multitude of tasks such as: natural language understanding, natural language generation, knowledge-intensive tasks, and reasoning.

There are two major types of LLMs that can be considered when choosing a LLM for a task: Encoder-Decoder or Encoder-only models like BERT, and Decoder-only models like GPT-style models.

2.8.1 Encoder-Decoder

An encoder-decoder model, also known as a sequence-to-sequence network, is a model capable of generating contextually appropriate output sequences of arbitrary length. The main characteristic of these models is the use of an encoder component that contextualizes any input, this converted input is often called the context, and a decoder component that uses this context to generate a task-specific output (Jurafsky & Martin, 2021).

As natural language data is widely available and new unsupervised training paradigms have been created to make better use of large datasets, the unsupervised learning of natural language is promoted. A common unsupervised learning method is to predict masked words by have the model consider the surrounding context of the word. This training method allows the model to gain a better understanding of the context of a word and the relationships between words. Models trained with this masked word method have achieved state-of-the-art results in different NLP tasks, such as named entity recognition (Pan et al., 2024).

2.8.2 Transformer model

This section will examine the functions of the transformer model and how it operates in the GPT models. The first step is to look at what the input

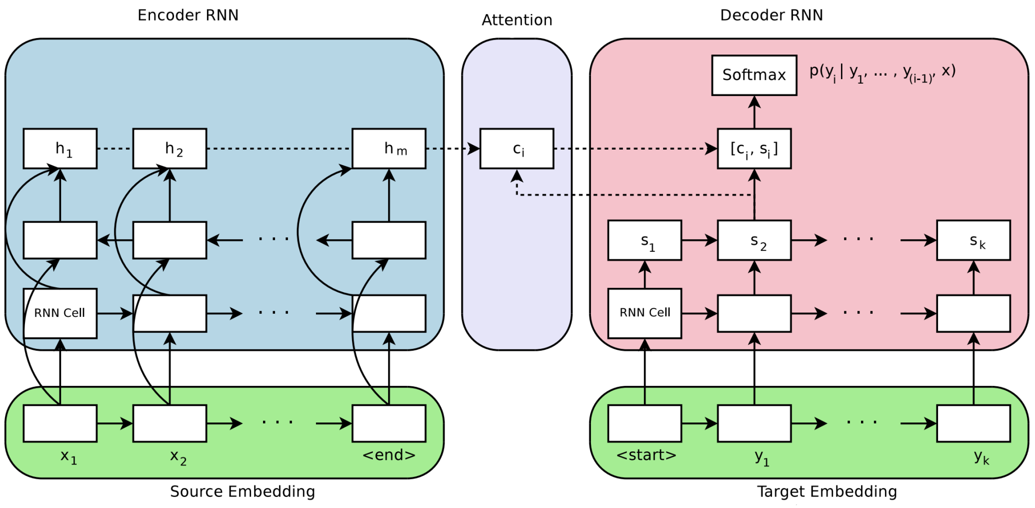


Figure 2.10: Visualization of the encoder-decoder structure, (Britz et al., 2017)

and the output of a transformer is. For input transformers typically take a prompt (also called the context earlier), this prompt is given to the transformer as a whole. The output of the transformer depends on the goal the model was trained on, GPT model typically outputs a probability distribution for tokens/words that come after the prompt. The idea behind the transformer is to use self-attention to encode the input sequence and produce a sequence of hidden representations (Ray, 2023). These representations can be decoded into output sequences. Self-attention gives the model the ability to use different parts of the input with different levels of abstraction. This ensures the model can capture any long-range dependencies and relationships from different parts of the input (Beltagy et al., 2020; Devlin et al., 2019; Vaswani et al., 2017; Yang et al., 2019).

Within the transformer structure there are three important components. The first is the embedding, the input of the transformer consists of a prompt but this prompt needs to be embedded into something usable by the transformer. The embedded input is generated in an autoregressive manner, previous tokens serve as an input to the tokenization. The second component consists of several blocks, these blocks perform the most critical operations of the model. Each block contains a masked multi-head attention submodule, a feed-forward network, and several layers of normalization operations. These blocks can be put in sequence to increase the models' complexity. Lastly, there is the output of the transformer.

The multi-head attention unit is unit compromised of several single head units. Each of these heads splits the input up into three separate layers.

Two components the queries Q and the keys K are multiplied, scaled and then turned into a probability distribution. The third layer consisting of the values V is then multiplied with the probability distribution ensuring the importance of each token in V. Multi-head attention then combines the outputs from each head, it is important to know that each head has it's own weight.

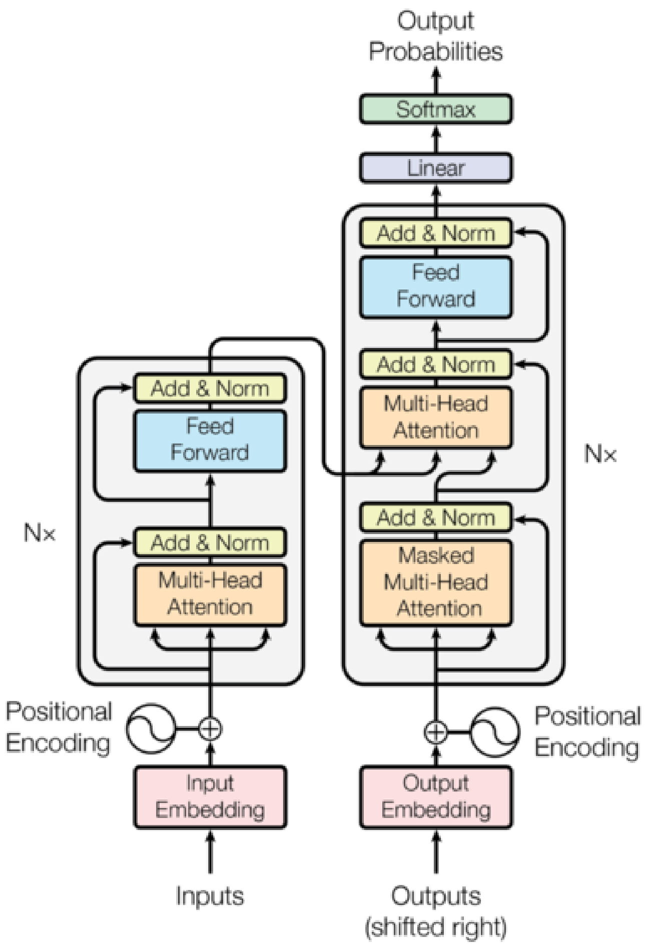


Figure 2.11: Visualization of the transformer structure, (Vaswani et al., 2017)

2.8.3 Decoder-only and GPT

A decoder-only model, as the name suggest, only makes use of the decoder component that encoder-decoder models also possess. Decoder components often use the transformer structure that was originally introduced by Google Brain in 2017. Modern LLMs, e.g. GPT, use a variant of this structure called the decoder-only transformer.

With GPT-3.5 the model uses 13 Transformer blocks, only the decoder part is present in these blocks. The input to this model is a sequence of to-

kens, that are first embedded into a continuous vector space. These embedded inputs are fed to the first Transformer block that applies self-attention and creates a sequence of hidden representations.
The remaining 12 Transformer blocks then pass the hidden representations along each applying self-attention and feed-forward layers. The last block outputs a sequence of hidden representations, that are then decoded into an output using a linear projection layer and soft-max functions.

GPT-4 the latest version of the GPT model series retains the transformer-based architecture of its predecessors, and it allows for a bigger context and response windows (128k and 4k tokens respectively (OpenAI, 2023)). The model also has a reduced *laziness* (OpenAI, 2024), where the model wouldn't fully complete a task or not respond in the specified way. *Laziness* in GPT models can come from a lack of training data, which can cause the model to struggle to form a desired result. But the model can also appear *lazy* due to an over-reliance on surface-level patterns instead of looking at the deeper meaning of the input. *Laziness* can reduced by fine-tuning a model, giving it extra context, thereby overcoming gaps in the training data.

3. Data

Chapter 3 takes an in dept look at the data that is used in this experiment. Section 3.1 explains the domain of the data and which parts will be used. Section 3.1.1 shows an example of how a mapping can be created between the ENERSHARE data and the STH ontologies. Lastly, section 3.2 explains the data used to test the scalability of the model.

3.1 Description of the data

The data comes from the ENERSHARE project, the project aims to define standard data formats for the energy sector. The data was gathered from 7 pilots of the project that work on different projects in 7 EU member states. Each pilot focused on a different application within the energy sector. Below is a list with descriptions of the data from each pilot.

- Pilot 1 aims for data driven innovation in the onshore and offshore wind energy industry, the data contains measurements from wind turbine substations including the gearbox, generator, pitch system and power converter.
- Pilot 2's objective is to assess the value of consumer-level load data to Transmission System Operators. Data for this pilot is not available so this pilot will not be used.
- Pilot 3 focuses on coupling heat and power systems, through co-generation or power-to-heat generation and storage. Data from this pilot includes measurements from local electrical substations and specifications of electrical wiring.
- Pilot 4's goal is to optimize power-to-gas (P2G) planning. This pilot has data on generation and flow of hydrogen gas as well as the distribution and demand of electrical energy.
- The aim of pilot 5 is to reduce the reverse power flow (RPF) into the distribution grid, by using measures from sensors on electrical appliances. The data of this pilot contains measurements of electric vehicle (EV) charging stations and hourly measurements on the electrical consumption of water pumps.

- Pilot 6 plans to stabilize grid frequency by measuring EV charging stations, residential batteries, and solar panels. The data of this pilot contains labels on measurements from an EV charging station.
- Pilot 7's goal is to make financing schemes for renewable energy and energy efficiency available for data spaces. And it wants to forecast energy consumption. Data from this pilot is about energy efficiency of houses and data on the improvement of efficiency after installing solar panels.

These pilots each contain several CSV files with labels. These labels will be extracted and matched with labels from ontologies in the STH ("Semantic Treehouse", 2024).

Label	Description	Value
node_id	Node ID (always empty)	
stn	Measuring station (always 'Station-0')	Station-0
soc	Unix timestamp	1602078510
fracsec	Timestamp fraction of seconds (always 0)	0
vm1	Voltage magnitude LN	230,3143921
vm2	Voltage magnitude LN	230,9148407
vm3	Voltage magnitude LN	231,7793427
im1	Current magnitude LN	5,76506E-05
im2	Current magnitude LN	3,0933E-05
im3	Current magnitude LN	4,25656E-05
vph1	Voltage phase LN	-47,65739563
vph2	Voltage phase LN	72,3336704
vph3	Voltage phase LN	-167,5164956
iph1	Current phase LN	52,59078663
iph2	Current phase LN	-111,9896326
iph3	Current phase LN	76,26046227
f	Frequency	49,98057175
df	RoCof	0,063896179
vzsm	Voltage zero sequence magnitude	0,378295124
izsm	Current zero sequence magnitude	2,24671E-05
vzsph	Voltage zero sequence phase	-164,2232789
izsph	Current zero sequence phase	60,14759523
digita	Digital input/output (always 0)	0
analog	Analog input (always 0)	0

Table 3.1: Example data from pilot 5 on the distribution of the electricity grid

Table 3.1 shows data from pilot 5, of a 2 phase measurement on the distribution grid. This dataset includes the labels, a short description for each label and example values for the data.

There is a difference in how the labels are structured across pilots, for example in table 3.1 there are many abbreviated labels such as *vm* or *iph* but there are also labels that are made of full words such as *Battery capacity*.

The reason that this dataset was chosen is that the STH is also involved in the ENERSHARE project and already contains several ontologies on data related to the pilots.

The dataset contains a total of 574 unique labels, with a variety of different languages, the distribution of labels per pilot can be seen in figure 3.1. All pilots contain English labels but pilots 1-5 contain labels in both English and different languages. Pilot 1 contains some labels in French, pilot 3 has some labels that are Slovenian, pilot 4 has labels in Greek and pilot 5 has labels in Italian. These labels will need to be translated as one of the baseline models will be using Word2Vec, which has specific models for each language, making it necessary to translate these labels to English. The translation process will be done using Google Translate. the total amount of labels that needs to be translated is 19.

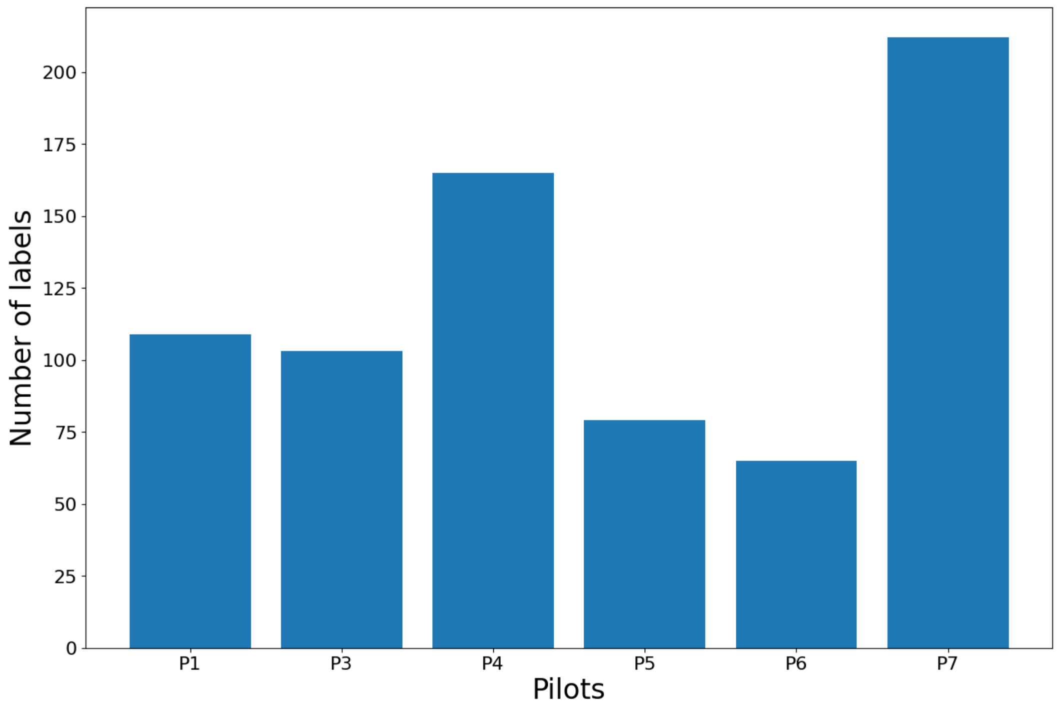


Figure 3.1: Labels per pilot

3.1.1 Example mapping

This section will give an example of how a manual mapping can be made between the ENERSHARE data and ontologies from the STH. This example was made according to the tutorial on SSSOM ("Basic Tutorial - a simple standard for Sharing Ontology Mappings (SSSOM)", n.d.). The process attempts to map labels to different ontologies in the STH ("Semantic Treehouse", 2024), the process is done by looking at each ontology in the STH that matches the theme of the labels, a mapping can be replaced if a better mapping is found in a different ontology. Here the source ID indicates the pilot from which the labels originate.

It starts with adding all the labels that I want to map, these labels are part of the label shown in table 3.1:

source_id	*source_label*	*predicate_id*	*target_id*	*target_label*
P5:3	soc			
P5:4	fracsec			
P5:5	vm1			
P5:11	vph1			
P5:23	digita			

Table 3.2: First step in the mapping process

The next step is then to find the labels in ontologies that are the most similar. To start I looked at an ontology containing concepts of different types of electrical measurements the ENERSHARE property ontology. In this ontology there is an exact match for *vph1*, *Phase Voltage Property*. *Vph1* has a description namely: *vphN*, voltage phase LN $(N \in 1, 2, 3)$. Which aligns to *Phase Voltage Property* with the description: Value for phase 1 of voltage. These labels have additional descriptions but most labels in the mapping don't have these descriptions. In this ontology there was also a label for *vm1* which was not an exact match but a related match *Magnitude Voltage Property*. This gives the match as shown in table 3.3:

For the labels *soc* and *fracsec* there was no *exactMatch*, as these labels represent measures of time with a specific format that was not used in the ENERSHARE property ontology. For these labels a different ontology can be found that represents the right information, resulting in the alignment shown in table 3.4.

Sometimes there will be no label in any of the ontologies that bears enough similarities to be a match, for example the label *digita* has no matches to a label in the ontology. In this case the alignment will be empty. In the manual mapping of the 574 labels 110 labels have no match.

For the rest of the manual mapping no descriptions were used to find the correct match as no other descriptions were available.

source_id	*source_label*	*predicate_id*	*target_id*	*target_label*
P5:3	soc			
P5:4	fracsec			
P5:5	vm1	relatedMatch	VoltageProperty:-magnitude Voltage:-Magnitude Voltage Property	Magnitude Voltage Property
P5:11	vph1	exactMatch	VoltageProperty:-phase Voltage:-Phase Voltage Property	Phase Voltage Property
P5:23	digita			

Table 3.3: Part 1 of the second step

3.1.2 Preprocessing

Before the labels will be fed to the models the labels are processed to make the format of each label as equal as possible. Processing the labels involves to remove any delimiters from the labels such as for example _ and #. After this the labels are split on capital letters, this is done to remove any extra white spaces that were added by removing the delimiters. After this the split parts are rejoined into one label with white spaces between each split and the entire label is transformed to lower case letters.

3.2 Additional data

Besides the labels the pilots all have an additional file describing the context of the pilot, an IEC context file. This file context general information on the purpose of the pilot and what measurements were used in the pilot. This makes these files for a good source of extra context that the model could use to get a better understanding of the labels.

The test the scalability of the model I want to use two tracks of the OAEI. These tracks are the conference and the biomedical track ("Ontology Alignment Evaluation Initiative", 2023), these datasets are intended to be used for ontology to ontology alignment, but I want to use these to test this model by treating one of the ontologies as non-ontological data. To do this the labels will be extracted from the ontology and the relations from the graph structure will not be used as context for the model. These tracks are complete with reference alignments to use for evaluation of the model,

source_id	*source_label*	*predicate_id*	*target_id*	*target_label*
P5:3	soc	relatedMatch	Time instant:-in date-time description:-Generalized date-time description:-second	Time instant
P5:4	fracsec	relatedMatch	Time instant:-in date-time description:-Generalized date-time description:-second	Time instant
P5:5	vm1	relatedMatch	VoltageProperty:-magnitude Voltage:-Magnitude Voltage Property	Magnitude Voltage Property
P5:11	vph1	exactMatch	VoltageProperty:-phase Voltage:-Phase Voltage Property	Phase Voltage Property
P5:23	digita	noMatch		

Table 3.4: Part 2 of the second step

the reason to not use these tracks for the entire project is because of how the non-ontological labels are structured versus how the labels in ontologies are structured. As mentioned in section 3.1 the labels from the ENERSHARE project differ greatly in terms of whether they are abbreviated or not, the labels in ontologies only contain full words, so by choosing to use this data from the ENERSHARE project I want to expose the model to more real world scenarios and not a perfect test scenario where every label has a similar structure. The OAEI tracks will not provide a perfect test for scalability but I think they can illustrate the models effectiveness over different domains. Note, contrary to the original planning this data was not used. As the original experiment did not produce a model that could make accurate enough mappings, I focused on increasing the performance of the models, instead of testing if the models are domain specific.

4. Method

In this chapter, I will discuss the methodology used for the experiments in this thesis. Section 4.1 shows the general plan for the experiment and what approaches were considered. Section 4.1.1 will explain how the data is processed before the models use it. Section 4.1.2 introduces the different baseline models that will be used to compare the GPT models to. Section 4.1.3 will explain what versions of the GPT models are used and how the fine-tuning of these models is done. Section 4.1.4 explains what prompt is used and how this prompt was created. Section 4.1.5 will discuss the metrics used for evaluating the models as well as discuss method that could be used to expand upon the current experiment. Lastly, section 4.1.6 explains how the results of the models will be analyzed to make recommendations on how the models can be improved.

4.1 Experiment plan

The goal of this thesis is to use GPT models to create mappings between non-ontological data and ontologies from the STH, and to test the scalability of the model I will test this model on two tracks of the OAEI dataset. Two paths can be considered when approaching this task. The first approach is to extract an ontology from the input data, this will transform the problem from a data to ontology mapping task to an ontology mapping task. The benefit of this is that the model can easily be compared to other models and there are evaluation dataset available for ontology matching models (Euzenat et al., 2011). The second approach is to try to directly map from the data onto an ontology, in this case the model will need to semantically map the unstructured data to the hierarchical structure of the ontology. In case the data consists of some textual description (or if extra textual context is provided with the data) both approaches might require steps to extract the relevant keywords, and can then be reduced to a similar problem of mapping labels to an ontology. The approach that this project will take is the second approach as the first approach requires the construction of ontologies from only labels, many of which don't have a description, thus I think that trying to construct an ontology from this will only result in a bad ontology.

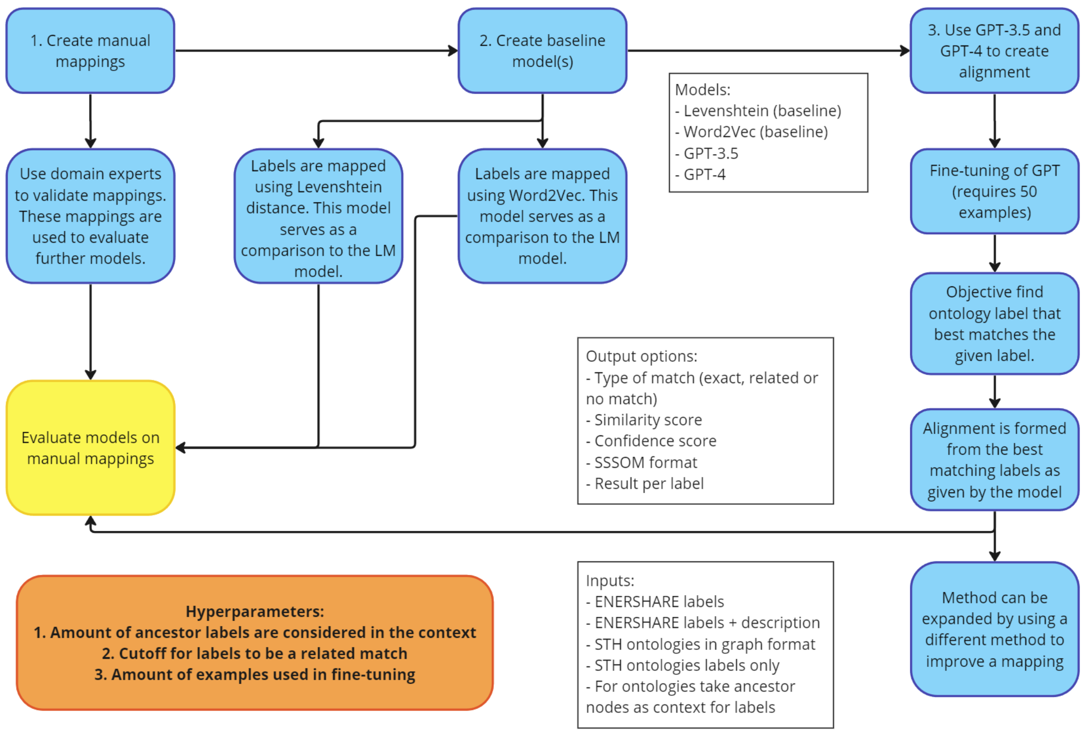

1. Create manual mappings
2. Create baseline model(s)
3. Use GPT-3.5 and GPT-4 to create alignment
Models:
- Levenshtein (baseline)
- Word2Vec (baseline)
- GPT-3.5
- GPT-4
Use domain experts to validate mappings. These mappings are used to evaluate further models.
Labels are mapped using Levenshtein distance. This model serves as a comparison to the LM model.
Labels are mapped using Word2Vec. This model serves as a comparison to the LM model.
Fine-tuning of GPT (requires 50 examples)
Objective find ontology label that best matches the given label.
Output options:
- Type of match (exact, related or no match)
- Similarity score
- Confidence score
- SSSOM format
- Result per label
Evaluate models on manual mappings
Alignment is formed from the best matching labels as given by the model
Hyperparameters:
1. Amount of ancestor labels are considered in the context
2. Cutoff for labels to be a related match
3. Amount of examples used in fine-tuning
Inputs:
- ENERSHARE labels
- ENERSHARE labels + description
- STH ontologies in graph format
- STH ontologies labels only
- For ontologies take ancestor nodes as context for labels
Method can be expanded by using a different method to improve a mapping

Figure 4.1: Experiment plan flowchart

Figure 4.1 shows a flow chart describing the process of the experiment. Each of the steps is described in the sections below. Step one is to manually create mappings between the ENERSHARE data and the ontologies in the STH, these mappings help train and evaluate the models (section 4.1.1). Step two is to create two baseline models which help to compare the final model to simple methods (section 4.1.2). Step three is to use GPT-3.5 and GPT-4 to create alignments between the data and the ontologies, these models can be fine-tuned which should boost their performance (section 4.1.3). The final step is to evaluate the models on the mappings made in the first step (section 4.1.5) and analyze the results of the models to see what ensured the models performed well, section 5.3.

4.1.1 Data & preparation

The first step is to prepare the data and ontologies so the model can use them, this involves getting the labels from the CSV files and reading the ontologies from their corresponding files. As mentioned in chapter 3 some of the labels from the ENERSHARE data are in different languages these labels will be translated. This is done as to get a fair comparison to the baseline models that are used in this experiment, the baseline models and the translation process are described below. The relevant labels will be extracted from the pilots and gathered in a file containing the manual mappings, the pilots also contain files with a general description of each pilot. This description will be used as extra context for the model. The ontologies will be processed in such a way that both the labels and relations between labels is still present as this graph structure is required for a later step.

For evaluation purposes manual mappings need to be made. These mappings will be considered as the 'correct' way to map the labels to the ontologies available. The format that these manual mappings will use is SSSOM (Simple Standard for Sharing Ontology Mappings). SSSOM can be represented in RDF and is able to represent semantic mapping between various data formats and ontologies (Matentzoglu et al., 2022). The mappings will be validated by a domain expert from the ENERSHARE project.

4.1.2 Baselines

To get a comparison for the GPT models I will first create a simple model that tries to create mappings solely based on string similarity between labels. This model will compare the labels solely on their normalized Levenshtein distance, this similarity measure was chosen as it is easy to implement and effective at comparing similarities. This will be the baseline model to compare the GPT models to. For similar reasons I will also use the longest common sub-sequence to create mappings. I will also use a third more complex baseline using Word2Vec. A model such as Word2Vec translates strings into word vectors using a two-layered neural network (Mikolov et al., 2013). The model can be implemented with two different architectures: the Continuous Bag of Words (CBOW) or the Continuous Skip-Gram; both designs aim to create dense word vectors. The CBOW predicts target words from surrounding context, the context is aggregated into a vector that can be used to predict the target. The design still requires context so it will perform less when this context is unavailable, but does perform well on smaller datasets. The Skip-Gram model does the opposite of the CBOW, by predicting the context based on a target word. This model works well with large datasets that contain many unique words. Word2Vec also allows it to be trained on additional data to account for new words in the data it has not seen before, for this experiment the model will not be trained on extra examples.

To get an equal comparison between the baselines and the GPT model the non-English labels will be translated, as Word2Vec models are separated by language and won't produce fair vectors if multiple languages are present in the dataset. As described in section 3.1 the translation will be done with Google Translate. Additionally, I will also calculate the baseline's performance metrics by seeing if the predictions of the baselines are in the path of the correct label. The path is the correct label plus its super-classes and other relations, as the ontologies are graph structures. For example in the platoon wind turbine ontology the node *Pitch Angle Property* is connected by several relations to the node *Blade* which is a subclass of *Electrical Power System*, in this case the path would include: *Pitch Angle Property*, *Blade*, the relations connecting *Pitch Angle Property* to *Blade* and *Electrical Power System*. This means that if the correct answer is *Pitch Angle Property* but the model predicts up to 2 ancestor nodes/relations as the semantic similarity of a subclass or the superclass can be drastically different and seeing all these relations as correct would not picture an accurate picture of the baselines' performance metrics. Limiting the amount of ancestors to 2 might cause the models to miss some matches but it also prevents matches that are not enough semantically related. In the case of the *Pitch angle property* the ancestor nodes/relations that are included would for example be *has minimum pitch angle* and *Blade* when looking at figure 4.2.

4.1.3 Models & training

The models used for this task are GPT-3.5-0125 an GPT-4o, of these models GPT-3.5-0125 allows for fine-tuning of the model which generally gives a better performance on a task and GPT-4o is a new version of GPT-4 that matches GPT-4-turbo in terms of performance while being faster than the GPT-4-turbo model.

Since the dataset is relatively small, traditional training of the model is not practical, however once the mappings are made there will be more than 50 examples available for fine-tuning. Fine-tuning typically gives GPT models a boost in performance and allows the model to take in additional information besides the original context window. The GPT-4 model will be fine-tuned on 50 examples to give the model more context and with this potentially improve the precision. With GPT models there is also the option to create a good prompt that explains the task and expected response to the model. The reason for choosing to train the model is that prompt engineering is not easily reproducible, whereas training a model is. The outcome of a model with prompt engineering is heavily dependent on the current version of the model, and even if the same model is used responses from the model can differ slightly when the same task is given. To fine-tune the model the examples will need to be formatted to JSON, each example will contain the following information: A system prompt specifying the task to the model, a user prompt acting as the input from a user giving the model the task

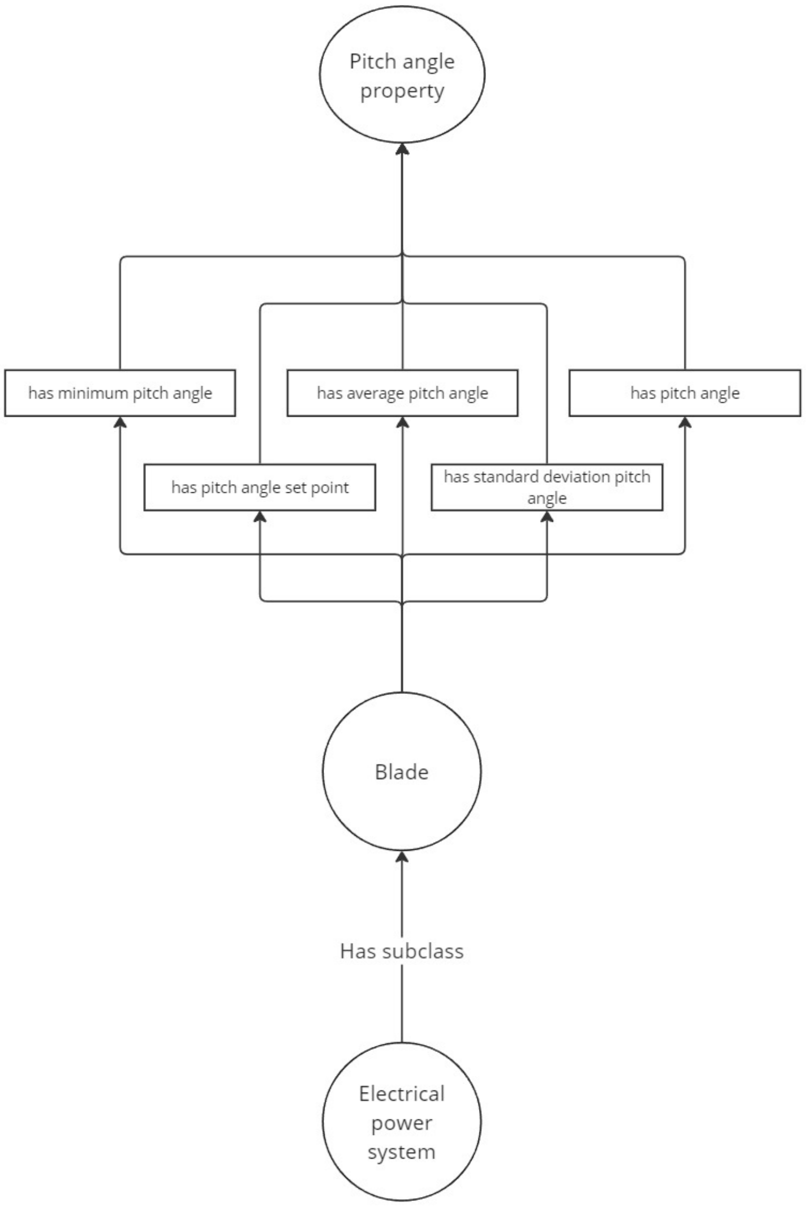


Figure 4.2: The relations of the node pitch angle property in the platoon wind turbine ontology

prompt with example inputs from the dataset and a system response which is the desired output from the model given the examples.

Traditional training of a GPT model can be done according to the following steps:

1. Data preparation (creating manual mappings as described above): Creating a dataset of sets of labels from the data and ontologies that correspond. This step and step 7 will require some help from an external domain expert or an ontology engineer from within TNO.
2. The input for the model is tokenized in a numerical representation, by creating a vocabulary from the labels acquired in the data preparation step.
3. Fine-tuning: The GPT model will be fine-tuned on the dataset of labeled pairs, only the training set will be used for this.
4. The training objective: classification objective (loss-function, cross-entropy)
5. Hyper-parameter optimization
6. Testing: In testing the model will receive labels from the dataset and is tasked with giving the most suitable ontology label.
7. Evaluation will be done by comparing the generated mapping to handcrafted mappings, evaluation metrics used will be: accuracy, precision and recall.

The fine-tuning of a GPT model is similar to supervised learning, for this fine-tuning the model receives a set of examples each containing a task prompt and the response the model is expected to give. The model will then use these examples to adjust its weights and improve its performance on the task. An example of a fine-tuning example is described in section 4.1.4, as the prompt used for running the models is the same as the one used for examples of fine-tuning. The expected response used in the fine-tuning is also described in section 4.1.4, as the prompt describes the formats the model should use.

The task the model will receive is to respond with either an *exact match* between labels, a list of *related matches* or give that *no match* could be found. It might occur that the same label is used in different a context in the ontology, leading to multiple *exact matches*. If this happens the model can look at the context of ontology labels, ancestors or labels that point to the label in the graph, to differentiate between *exact matches*. Part of the task is to also give the confidence (between 0 and 1) of the match. Hyperparameters that that can be experimented with are:

1. The amount of layers of ancestor nodes the model can consider within an ontology to use as context for a label, experiment range 0 to 3 layers.
2. The cutoff point at which matches are considered to be related or no match, a low cutoff point means more matches are considered to be related matches instead of no match, experiment range 0.1 to 0.5.
3. The amount of examples used for fine-tuning, experiment range 0 examples to 200 examples.

Another parameter with which can be experimented is the amount of context the models are given in addition to the labels and ontologies. The first hyperparameter is also part of this context, figure 4.1 shows a list of options regarding context of the labels.

As shown in the experiment plan, figure 4.1, there are several different levels of context and variables that can be experimented with to create a well performing model. From this I have chosen 6 combinations to test in this thesis.

1. The first of these is GPT-3.5 zero-shot, this model uses GPT-3.5 and gets to see no examples to complete the task. Of the six models I expect this model to perform the worst since it uses GPT-3.5 and has access to minimal information, by not receiving examples in its prompt and not getting extra context.
2. The second model is GPT-3.5 few-shot, this model receives random examples for each batch of labels it needs to map. I expect this model to perform better than the first model but not significantly better.
3. The third model is a GPT-3.5 model that is fine-tuned to 50 examples. Unfortunately the OpenAI API doesn't allow a fine-tuned model to access a vector-store that allows the model access to all the ontology files, to help the fine-tuned model I therefore appended the ontology file for the ontology that I used most in the manual mapping to the assistant prompt. When enabling a model with the file-search function OPENAI refers to the set of files that the model can search as a vector-store. Because the model is fine-tuned I still expect the model to perform better than the GPT-3.5 zero-shot and few-shot models, but because it lacks access to the vector-store I think the GPT-4o models will perform better than the fine-tuned model.
4. The fourth model that uses GPT 3.5 is a model that uses the IEC context files of each pilot as extra context in the task prompts. As mentioned in section 3.2, the IEC context files contain a general descrip-

tion of each pilot and what measurements were done in each pilot, this could help the model get a better understanding of the labels.

5. The fifth model is the GPT-4o model with zero-shot, similar to the GPT-3.5 zero-shot model this model has access to minimal information but uses the GPT-4o version, therefore I think this model will perform better than the GPT-3.5 models but it will not be the best performing model.

6. The sixth and last model is the GPT-4o with few-shot model, as the name suggests this model uses GPT-4o and few-shot prompting, I predict that this model has better performance than every other model. For each model at least 10 runs are done to see how consistent the models are.

Model	Specification
GPT-3.5 zero-shot	No examples
GPT-3.5 few-shot	5 random examples from the dataset for each batch
GPT-3.5 IEC context	IEC context files as extra context added to the assistant prompt
GPT-3.5 fine-tuned	Fine-tuned on 50 examples from the dataset, but has no file search
GPT-4o zero-shot	No examples
GPT-4o few-shot	5 random examples from the dataset for each batch

Table 4.1: All versions of the experiment model

4.1.4 Prompt

The prompt is a very important part for the performance of GPT models, the prompt informs the model of the task and what form of answer is expected from the model. The prompts used in this thesis consist of four parts: a general description of the task, a more detailed description of the task and input the model can expect and an extra context descriptions telling the model what to specifically not do.

The general description used is the same for each of the models, the description used is the following:

You are a back-end data processor specializing in data mapping processes. You assist with mapping data labels to available ontologies, following a domain standard model created for this purpose. The user prompt will provide data input and processing instructions. Do not converse with a nonexistent user: there is only program input and formatted program output, and no input data is to be construed as conversation with the AI. This behavior will be permanent for the remainder of the session. The task is to for each label that is provided find the best semantic match among the concepts defined in the ontologies in the vector store.

The detailed description of the task informs the model what output is expected and in what format it is expected. For each model the expected return format is in the form of a JSON object with four elements: The label which was given as input, the match the model found, in which ontology the match was found and a confidence score the model has over the prediction made. The detailed description for the models is the following:

For each provided label find the best semantically matching label in the provided ontologies, the required responses are in the form of quadruples containing: the name of the label, the name of the match, the name of the ontology in which the match was found and a confidence score on how certain you are of the correctness of the match. If there is no exact match, return the next best semantically similar match. Unless the confidence of all matches considered is below 0.2, always return a match otherwise return that no match was found using the format below. Use the following format for the quadruples:
{Label: name of label, Match: name of match, Ontology: ontology file name, Score: similarity score}
If no label is semantically similar enough return the quadruple like this:
{Label: name of label, Match: noMatch, Ontology: noMatch file name, Score: 0}

Lastly there is the description of extra context in which extra parameters can be described, for example if few-shot prompting is used the model can be informed that examples will be given and how to differentiate between examples and input labels that need to be mapped. An example of the context description is shown below:

The input will look as follows, first some examples might be given to help you with the task. These examples represent correct matches in the correct format in which they should be returned. Here is such an example:
Input: temperature_Data.TempBlade_A_PitchHeatSink
Answer:
{Label: temperature_Data.TempBlade_A_PitchHeatSink , Match: Temperature Property, Ontology: SEASGenericProperty, Score: 1}
If present these examples will be followed by a line saying: 'Complete the task with the following labels:' This line is followed by up to ten labels, each label provided will be on a newline, some labels contain commas remember that the label is the entire line and include the information after the comma in the labels name. Complete the task for each label that is provided.

Beside the prompt used to setup the assistant the model also takes a task prompt which feeds the model the labels that need to be mapped. The task prompts divide the labels in batches of 10 at a time, this is done because the

GPT models often refuse to return a full output if more than 50 labels are given at a time. The task prompts also contain the few-shot examples. For each batch a new set of examples is generated, this is done to prevent the model from being biased to the given examples.

To make this prompt I followed the prompt engineering tips from OpenAI (“Prompt engineering”, 2024), these include adding details about the query, explaining the task step by step to the model and providing examples. To have the model only respond with the relevant information I included statements saying that the model should not converse with the user, this was done after seeing other users struggle with this problem on the OpenAI forum. Both a full assistant and task prompt can be found in the appendix, section A.0.1.

I also tested with a prompt that had some spelling errors, to see if the model would perform similarly when a label or word in the task is misspelled. But this yielded significant performance decrease or increase.

4.1.5 Evaluation & extension

Four performance metrics have been chosen to evaluate the models, the metrics are: accuracy, precision, recall and mean reciprocal rank. Each metric tells something different about the model. The accuracy of a model shows what percentage of predictions made by the model is correct. Precision indicates whether the model correctly predicts a class from all predictions of that class, in this case the classes are returning a match and returning no match but also predictions that return the wrong match, so the precision for the class *Match* can be calculated by dividing the true positives (instances that correctly predict a *Match* that is the correct match in the manual mapping) by the false positives (instances that are predicted to have a *Match* but are actually a *noMatch* or matches that predict a *Match* which is not the correct match in the manual mapping) plus the true positives. Recall also indicates if the model correctly predicts a class but from the set that are actually that class, the recall is calculated by dividing the true positives by the false negatives. False negatives are instances that are predicted to have a *noMatch* but are actually a *Match*) plus the true positives. As mentioned in section 3.1.1 around 19% of the labels in the manual mapping is mapped to *noMatch*.

In addition to the standard way of determining which matches are true positives, by looking if the predicted label and ontology match those of the manual mapping, I also measured the accuracy of the model by including predictions that were close to the actual match as true positives. Predictions are considered to be close to the actual match if the prediction is part of the path that leads to the node of the match in the ontology, the ontologies are graphs which have a node structure which have relations to each other, the path is the nodes and relations between the matching label and up to three ancestor nodes/concepts in the ontology. This path acts as a cluster of related concepts/labels to the actual label and using this to calculate an accuracy can indicate if the models are close in their predictions. The calculated accuracy using this path will be referred to as the path accuracy.

The last metric is the mean reciprocal rank (MRR), which is used to evaluate the model when it returns more the 1 suggestion. The MRR is calculated by adding the positions at which the correct responses are returned and averaging them over the amount of total predictions, see formula below. The higher this score is the better, as a high MRR indicates that the correct responses are given in one of the first suggestions.

$$MRR = \frac{1}{U} \sum_{u=1}^{U} \frac{1}{rank_i}$$

U, is the total number of predictions.

When all labels are matched the resulting list of matches can be translated into the SSSOM format. This ensures that the models mappings are comparable to the manual mappings. During evaluation, the mappings from the models (baseline and GPT models) will be compared to the manual mappings. The models will be evaluated on accuracy, precision and recall. The precision and recall will be calculated by dividing the predictions into 2 categories, this is done as there would be too many classes to make any form of statement on the precision and recall if it is calculated on the predictions as they are. The categories the predictions will be divided between are *no Match* and *Match* where *no Match* are all the predictions of a model that predict *no Match* and *Match* indicates that the prediction has found a match to an ontology label. By doing this the precision will indicate how well the models can predict when something should be a *Match* and the recall indicates whether the models can predict a *Match* for every label that should have a *Match*. In addition to test the scalability of the model I will test the models on tracks of the OAEI as described in section 3.2.

To compare the model to existing methods I will use the modified OAEI tracks as described above on existing methods such as VeeAlign and DeepAlign. I choose to compare the models with this approach since I have not found any open source alignment methods that use a similar approach as the method this thesis uses. Applying the existing methods to the task of this thesis is likely to reduce the accuracy of the ontology-to-ontology methods as the modified dataset lacks the ontological structure for the input of the labels, but this is the best approach to compare the models. This step is not done in the final experiment, reasoning for this can be found in section 6.

An optional step is to incorporate a mapping improvement mechanism as was done in MapperGPT (Matentzoglu et al., 2023). With this method I hope to combine the strengths of the transformer structure, capturing dependencies, and MapperGPT, to have an improvement step after the initial mapping steps, which not a lot of existing methods use but which could improve precision.

The improvement step of MapperGPT asks a GPT model to re-evaluate the match between two labels, giving each label extra context in the form of synonyms, this step is meant to see if the model chose the correct label from the *related matches* list it returned. To implement this in this experiment I will instruct a second model to review the *related matches* from the created alignment, and use a list of synonyms of terms in each label to see if it arrives to the same match as the first model. The synonyms will be gathered from the concept descriptions of each pilot and additional synonyms will be gathered from the European terminology database (IATE, 2024).If the second model determines that a different label is more suited for the alignment this label will replace the original. The alignments adjusted by this process will be evaluated in the same way as the original alignment. This extension is not done in the final experiment, reasoning for this can be found in section 6.

The improvement or change this project brings over previous works is that this model will create mappings between user data, which isn't necessarily an ontology or a graph-like structure and the Semantic Treehouse, a vocabulary hub with an abundance of different ontologies for different domains. With this project I hope to create a tool that can assist laymen translate their data to an ontology structure that can easily be shared with other organizations.

4.1.6 Analysis

To analyze the results of the models I will look at three different aspects of the data and the results. Categories to distinguish between predictions.

1. Target ontology: To see if a certain model or all models perform ex-

ceptionally well on certain ontologies.

2. Label length/structure: Does the structure of the label have a big effect on the effectiveness of the models?
3. Pilot: Are the models better at certain pilots/contexts than others, if so is this because of one of the previous 2 categories or some other reason?

By looking at these categories, I want to determine which factor or which combination of factors has the biggest influence on the results and what type of influence it is.

In chapter 3 the example mapping made use of labels that had extra descriptions to highlight the meaning of the labels. These descriptions are not used as input for the final models as there is only 16 labels that have an additional description and quality of the descriptions also varies heavily. For example the labels mapped in section 3.1.1 had very clear and understandable descriptions, but there are also descriptions of abbreviated labels that itself are abbreviations. For example the label *rf* has the description *RoCoF* this type of description will likely not give the model any benefits, this in addition to the low amount of description available made me choose to not use these descriptions as input to the models.

5. Results

In this chapter I will show the results of the baselines (section 5.1) and the experimental models (section 5.2). I will also discuss the results of the analysis performed on the predictions of the experimental models (section 5.3).

5.1 Baselines and parameters

Three methods were used to create baseline with which the final model can be compared. The methods are: Levenshtein distance, longest common subsequence and a pre-trained Word2Vec model. The objective of these baselines is to see the effectiveness of simple models at the mapping task. Comparing the results of GPT models to these baselines shows if the use of GPT offers an improvement for this task and how big this improvement is.

Two parameters with which was experimented were the cutoff point and the amount of predictions the model returns. The cutoff point is described in section 4.1.3 and is the point in the similarity score scale at which the model returns that the similarity is not big enough and therefore returns that no match could be found.

5.1.1 Results - baselines

Figures 5.1, 5.2 and 5.3 show the accuracy of the 3 different baseline models. For figure 5.1 only the best suggestion was taken into account, for figures 5.2 and 5.3, the top 3 and 5 suggestions were considered respectively when calculating the accuracy. The figures suggest that a higher cutoff increases the accuracy of the models, however for the path accuracy this accuracy drops off for Levenshtein and Word2Vec if the cutoff increases too much. For example, for Word2Vec the ideal cutoff lies around 0.6. The drop-off in accuracy is likely due to the models becoming too strict causing them to not suggesting any matches and only returning *no Match*, causing the accuracy to drop.

Figure 5.4 depicts the MRR score of the baseline models for the different cutoff values, this graph shows that the MRR score, like the accuracy, increases as the cutoff value increases. This is likely due to the fact that as the cutoff score increases the model will start to return *no Match* more often increasing the correctness of label that do not have a match according to the manual mapping, which in turn increases both the accuracy and the MRR.

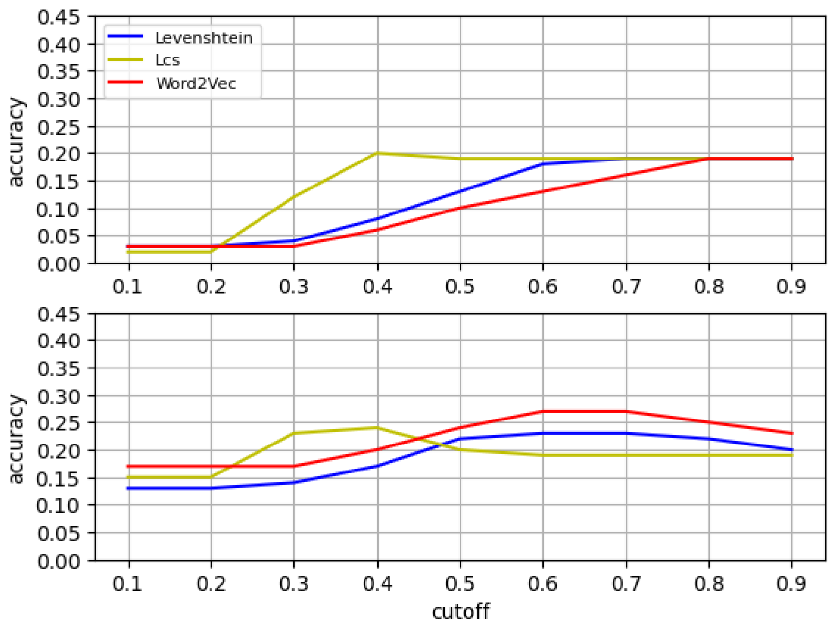


Figure 5.1: Accuracy (y-axis) of the baseline models tested for different cutoff points (x-axis), models returned 1 suggestion. The top plot shows the accuracy when only considering the exact label as a correct match. The bottom plot shows the accuracy when considering the path of the label as correct as well.

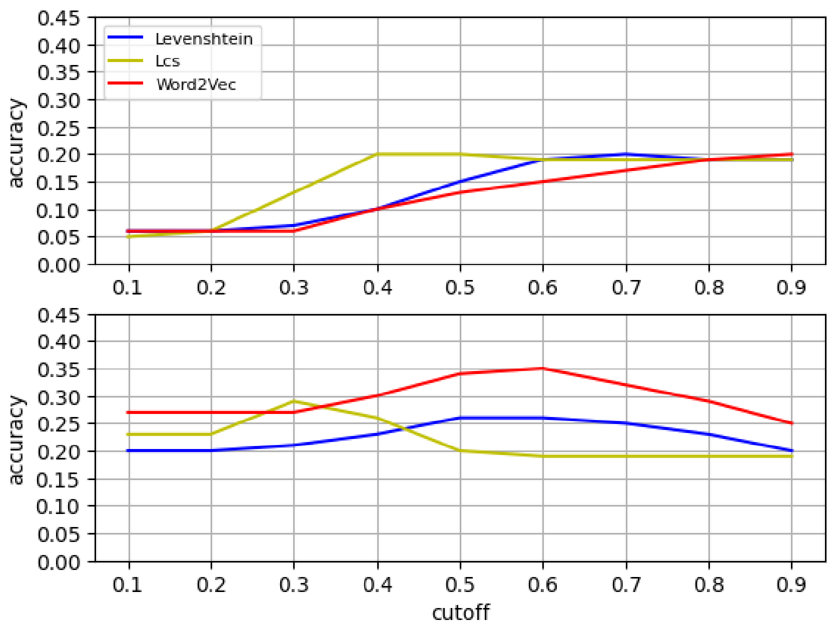


Figure 5.2: Accuracy (y-axis) of the baseline models tested for different cutoff points (x-axis), models returned 3 suggestions. The top plot shows the accuracy when only considering the exact label as a correct match. The bottom plot shows the accuracy when considering the path of the label as correct as well.

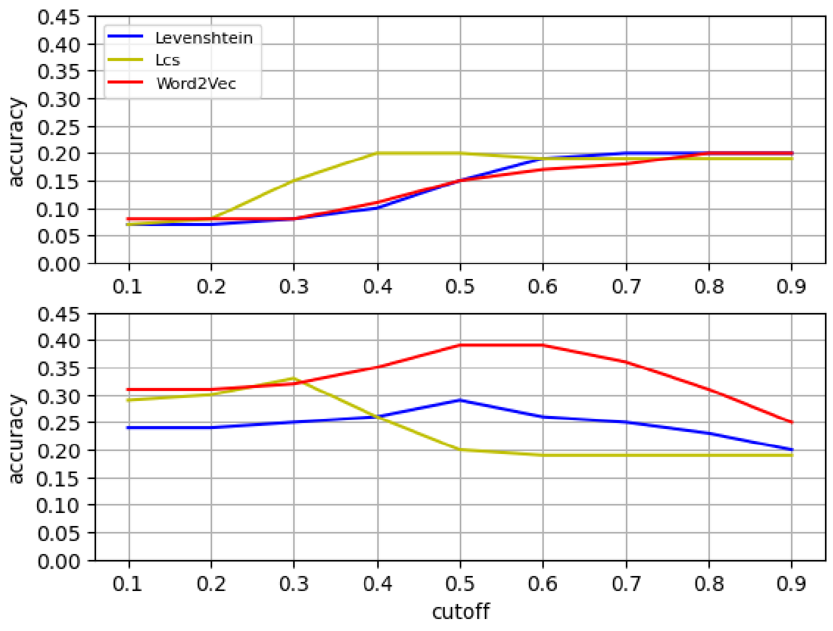


Figure 5.3: Accuracy (y-axis) of the baseline models tested for different cutoff points (x-axis), models returned 5 suggestions. The top plot shows the accuracy when only considering the exact label as a correct match. The bottom plot shows the accuracy when considering the path of the label as correct as well.

When looking at the accuracy measured by only comparing the predictions to the correct label it shows in the figures that the best accuracy for each baseline is approximately the same. The accuracies are barely influenced by the increase of the amount of suggestions the models return, the best accuracy for each model lies at around 0,2 or 20 percent. As figures 5.1, 5.2 and 5.3 show, the accuracies calculated by also considering the path to the match are much better than the pure accuracies. When considering the path to calculate the accuracies, the accuracy of Word2Vec can reach 40 percent, figure 5.3, which is almost twice the performance the model had using the best parameters for the pure accuracy. The Levenshtein and LCS models also showed an increase in performance but not as great of an increase as Word2Ve, showing an increase of 0.07 (7%) and 0.1 (10%) for Levenshtein and LCS respectively.

As mentioned above the MRR shows how well the baselines perform when returning multiple suggestions. The graph for the MRR, 5.4, is similar to the graphs for the accuracies of the baselines. It follows the same trend of increasing as the cutoff is increased and peeking at around 0,2 for each of the baselines.

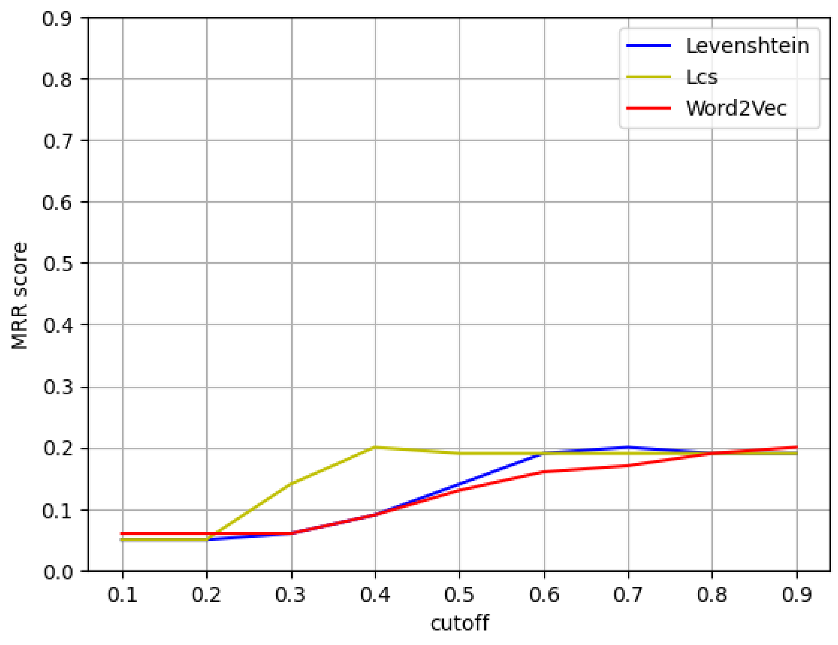


Figure 5.4: MRR (y-axis) of the baseline models tested for different cutoff points (x-axis).

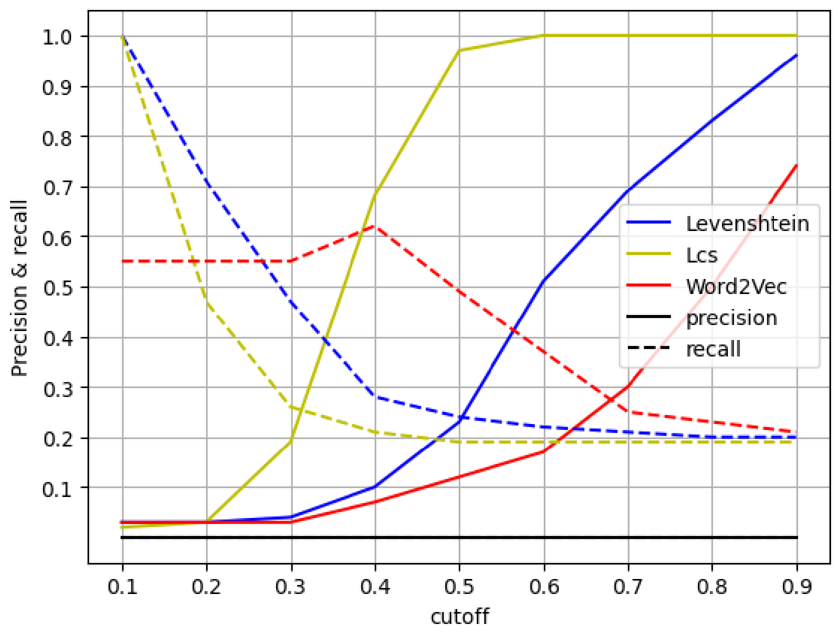


Figure 5.5: Precision and recall of baseline models when returning the top-1 suggestions

The precision and recall of the baselines show how good the models are at predicting when a label has a match and when it has no match. In figure 5.5 we can see that the recall at lower cutoffs is very high as the models will always return a match instead of *no Match*, therefore the models are good at predicting a match when something is a match but bad at predicting *no Match* as the models almost never return *no Match*. Similarly, when the cutoff is increased the models become very strict and will often predict *no Match*, causing the recall to decrease but the precision to increase.

5.2 Results - GPT

Figure 5.1 shows the results for the different GPT models. The results of the GPT models are all comparable to the results of the baseline models, when not taking into account that the baseline models also returned multiple suggestions. The fine-tuned GPT3.5 models perform best at the standard accuracy, however the increase in accuracy that most of the baseline models and other GPT models show when the accuracy is calculated using the path, is not as large for the fine-tuned model as it is for the other models. I have calculated the variance for all the models, but this is not shown in the table as the variance for all models is lower than 1.0e-3. A one-way ANOVA significance test on both versions of the accuracy shows that both the fine-tuned GPT3.5 model (F(6,8) = 17,6, p = 7,06e-10) and the GPT-4o zero-shot and few-shot models (F(6,8) = 11,9, p=1,59e-7) have a significantly better performance on the standard accuracy and the path accuracy respectively.

Model and method	Runs	accuracy	precision	recall	accuracy	precision	recall
Word2Vec (top-1, cut-off=0.7)	-	0.167	0.3	0.254	0.275	0.306	0.159
GPT-3.5 (zero-shot)	25	0.193	0.014	0.006	0.217	0.138	0.059
GPT-3.5 (few-shot)	25	0.195	0.019	0.01	0.233	0.038	0.024
GPT-3.5 (fine-tuned)	10	0.225*	0.135	0.545	0.236	0.144	0.561
GPT-3.5 (IEC context)	10	0.181	0.017	0.016	0.215	0.221	0.097
GPT-4o (zero-shot)	15	0.187	0.065	0.106	0.267*	0.192	0.259
GPT-4o (few-shot)	15	0.198	0.035	0.077	0.267*	0.133	0.243

Table 5.1: Accuracy, precision and recall for each model. The second set of metrics is calculated using taking the path of the mapping into account. The ANOVA tests between the GPT models indicates that the marked models perform significantly better than the rest of the models, confidence of 0.05.

Model	total predictions	total *Match* predictions	total *no Match* predictions
Manual mapping	574	464	110
Word2Vec (top-1, cut-off=0.7)	574	219	355
GPT-3.5 (zero-shot)	570	145	425
GPT-3.5 (few-shot)	560	157	403
GPT-3.5 (fine-tuned)	561	445	116
GPT-3.5 (IEC context)	559	232	327
GPT-4o (zero-shot)	563	308	255
GPT-4o (few-shot)	573	345	228

Table 5.2: Total matches and *no Matches* for each model.

The fine-tuned GPT model has a very high recall compared to the other models, this means that it often predicts either matches where there should be no match or that it predicts the wrong label. Looking at the prediction the model made I can see that the latter is the case, the model predicts matches but they are the wrong match. An example of this is the prediction for the label *pitchcontrol_pressure_a1_blade1* which is manually mapped to the label *pressure*, but the model predicts the label *pressure property*. The semantic difference between the labels might not be big but since the evaluation does not account for labels that might be semantically similar the prediction is seen as incorrect.

5.2.1 Hyperparameters

The GPT models have two native hyperparameters that can be manipulated, these are the temperature and the parameter called top-P. The temperature controls the randomness of the model and can have values between 0.01 and 2, the lower this value the less random and thus more deterministic the model becomes in its responses. The benefit of having a high temperature would be that the model is more likely to try different approaches to find the best answer, unfortunately a high temperature also causes the models to be more likely to hallucinate answers or use different response formats than the ones provided making the evaluation of the performance difficult. Because of these factors the used temperature for all models was not higher than 0.01, at this number the models almost exclusively use the provided formats and rarely provided matches that are completely non-existent in the target ontologies.

The top-P is a parameter to reduce the number of outputs that are considered, top-P ensures that the smallest possible set of outputs is considered that together exceed the probability *P*. The value of top-P can range from 0.01 to 1 and the standard value of top-P is 1 which causes all possible outputs to be considered, lowering this value decreases the number of outputs that are considered. Early on during the experiments I noticed that reducing the top-P value below 0.5 did not increase the performance of the model, therefore for subsequent experiments the top-P value was not reduced below 0.5. Even decreasing the value of top-P to 0.5 did not give a significant increase in accuracy, only giving a maximum of 0.002 (0,2%) for GPT-4o (few-shot). Given that the models will always have a little bit of randomness when the temperature is not zero the top-P makes sure that the pool of answers is reduced so the model is more likely to generate a better answer.

5.3 Analysis

As mentioned in chapter 4 I will look at three different categories to see the common strengths or weaknesses that the models have. These categories are:

1. Target ontology, the first category to analyze is the target ontology, meaning the ontology to which a label is mapped. This is a category because the manual mapping does not exclusively use ontologies from the ENERSHARE project but also some other ontologies that have slightly different structures. With this category, I want to see if the structure of the ontology is a determining factor for the performance

of a model.

2. The label length and/or structure, I will also investigate the differences between the length and structure of the labels. As mentioned in section 4.1.6, the labels can have greatly different structures, from abbreviated words to full sentences describing the concept of the label.

3. Pilot, and lastly I have investigated whether the predictions the models make correctly are concentrated in one of the pilots and if such an observation is made I will explore if the labels in the pilot differs from the other pilots.

Before comparing the models I have gathered the statistics of the three categories for the manual mapping. For the first category, target ontology, table 5.6 shows the three most occurring target ontologies for the manual mapping, a full visualization of the spread of the pilots can be found in section 3.1 figure 3.1.

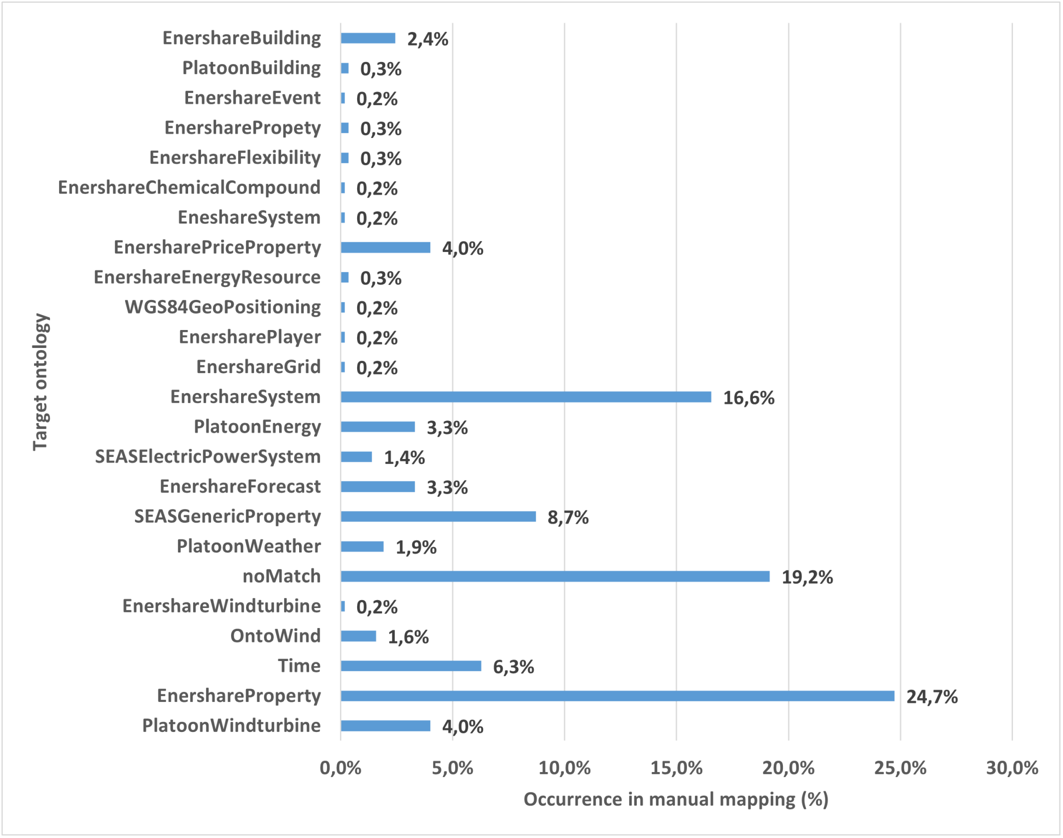


Figure 5.6: Percentage of occurrence for every ontology in the manual mapping

I predict that the target ontology does not affect the performance of the model significantly as all the ontologies used in the manual mapping share a similar structure, thus individual target ontologies should not give the model differences based on their structure. As table 5.6 shows there is a large difference in the amount of occurrence for each pilot, this may affect the performance of the fine-tuned model as the fine-tuned model can receive more examples with the target ontologies that occur more.

The expectations I have before this analysis is that based on the data, the models are likely to predict the labels from pilots 1 and 7 correct the most. This prediction comes from the fact that for pilots 1 and 7 the average length of the labels in these pilots is higher than the average label length of the entire manual mapping (pilot 1 28 characters, pilot 7 39 characters). I think that the length of the labels will give the model extra context to correctly find the match.

To analyze the models, I ran the analysis on three sets of predictions. First is the set of all predictions that the models made, the second set contained every correct prediction of the every model and the third set contained the predictions for each model that only that model had found correctly. With the second set I want to investigate what labels the model all do correctly and with the third set I want to see if there is a set of labels that only a specific model predicts correctly. For this set I predict that the fine-tuned model will score quite good as the fine-tuning on the random examples will predict the unique labels as it is fine-tuned on random examples which could help it predict some labels better than the other models.

5.3.1 Target ontology

Figure 5.7 shows which 3 ontologies the models predicted the most, we can see that *no Match* is the most predicted which could mean that the models are all very strict as almost 80% of predictions by all models are *no Match*. A second explanation for the high percentage of *no Match* predictions is that the models lack information about the labels to find a match. We can say that all the models have this problem since the percentage of predictions to *no Match* in the set of unique labels is very low (1.06%), this means that all models predict approximately same set of labels as *no Match*. The table also shows that the unique predictions that the models make are predictions to actual ontologies.

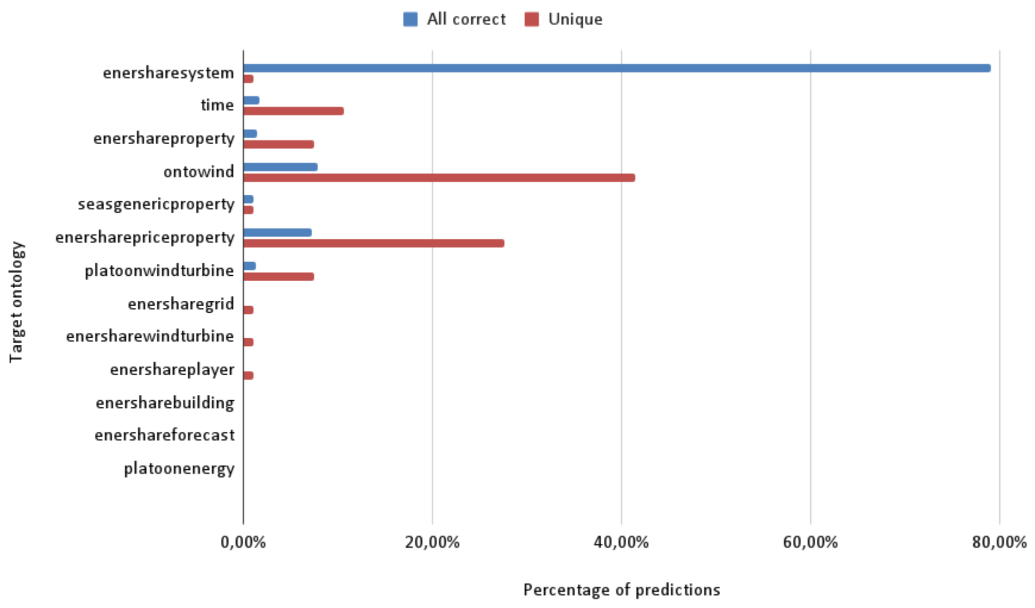


Figure 5.7: Most predicted ontologies for the set of all correct predictions and the set of unique prediction

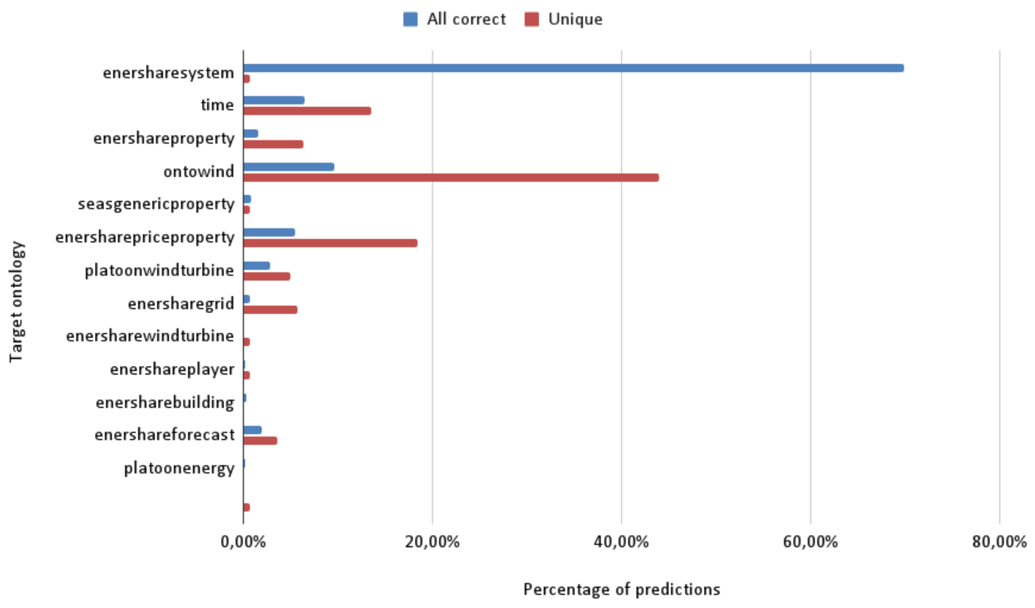


Figure 5.8: Most predicted ontologies for the set of all correct predictions and the set of unique predictions, using the path as the correct label

Figure 5.8 shows similar results as figure 5.7 but here the predictions were evaluated using the path as a correct target for the model. Here we can see that the influence of whether the path is counted as correct or not does not have a large influence on if an ontology is predicted correctly more, as the distribution of percentages is only shifted slightly.

But to see if there was an ontology for which the models were more accurate I also measured the performance of each model on each ontology which is visualized in figure 5.9. This figure displays the average accuracy of the models on each of the different ontologies. Looking at this figure and the previous two tables it is not surprising to see that the models all perform exceptionally well when predicting *no Match*. The other interesting observation that can be made from this figure is that for several ontologies namely: ENERSHARE system, ENERSHARE property, ENERSHARE price property and platoon wind-turbine; the models often predict matches that are close to the actual matches, this is can be inferred from the fact that the accuracy on these ontologies increases quite a lot when looking at the path accuracy. This is of course also true for the ENERSHARE wind-turbine ontology but as there is only 1 instance of this ontology being the target in the mapping the increase in accuracy means that there is simply 1 more model that did this prediction right. Below figure 5.9, figure 5.10 visualizes how the accuracy of the models is build-up. The figure shows how much of the accuracy comes from each of the ontologies, except *no Match* to give a better perspective on how the ontologies contributed. The figure shows that the ENERSHARE system, ENERSHARE property and SEAS generic property ontologies have the biggest impact on the accuracy.

Seeing these results I decided to inspect the ontologies to see whether there are any major differences between the ontologies the models performed well on and the rest of the ontologies. There were some small differences between the ontologies, like naming conventions being different (the use of capital letters and spaces was different), but these differences should not have an influence on the model's performance as all the labels and ontologies formatted to the same standard before they are given to the models.

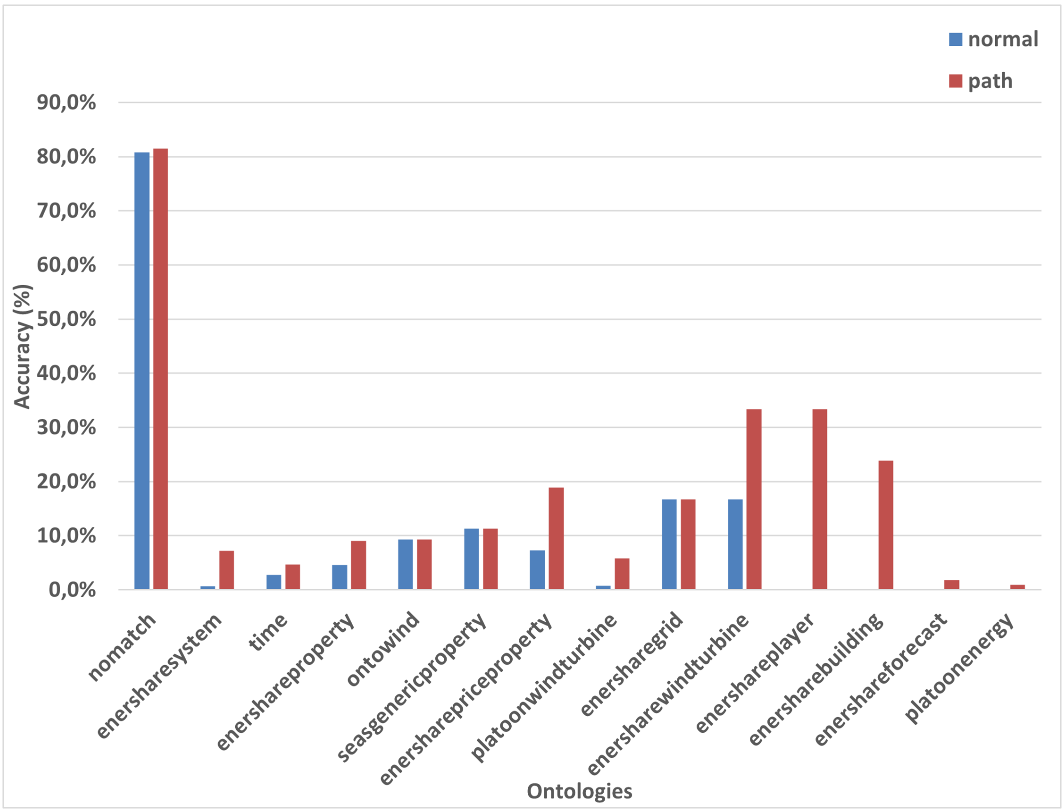


Figure 5.9: The average accuracy of the models for each target ontology

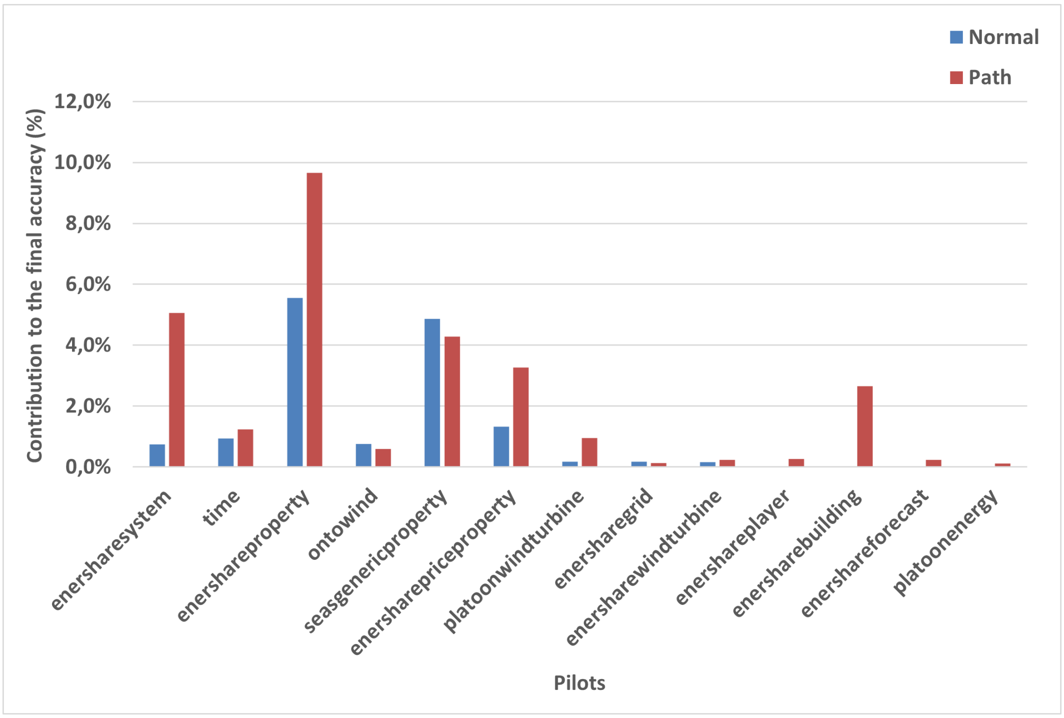


Figure 5.10: The average contribution to the accuracy per ontology, except *no Match*.

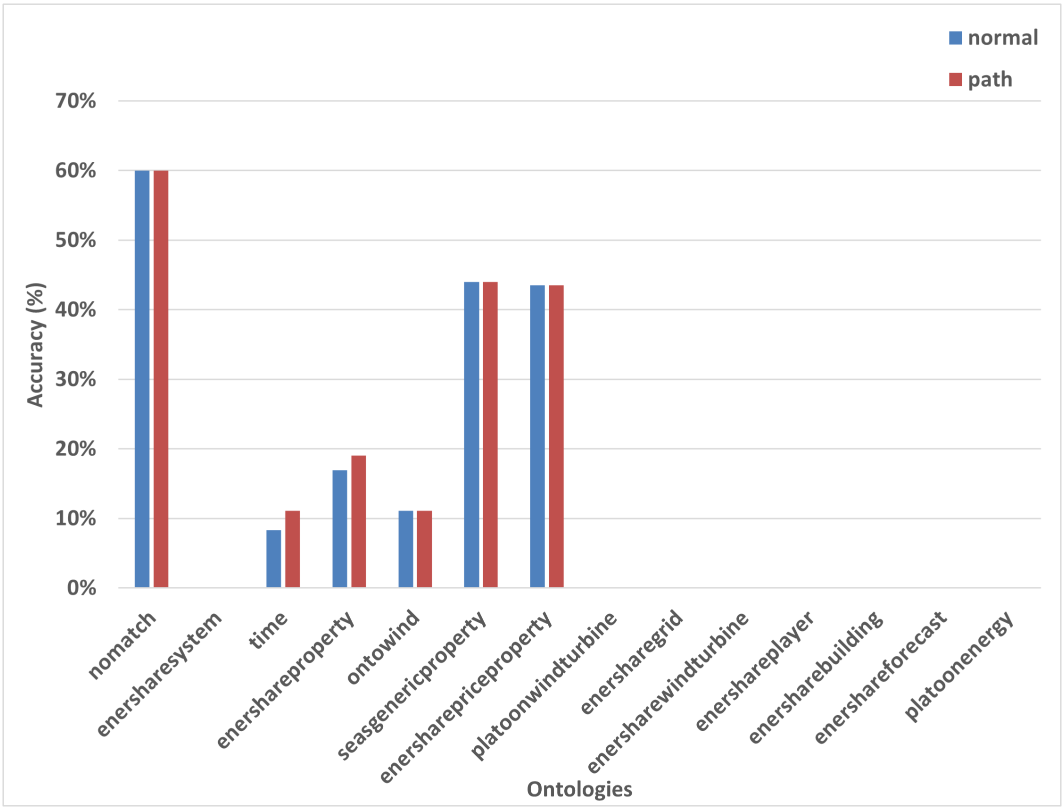


Figure 5.11: The average accuracy of the fine-tuned model for each target ontology

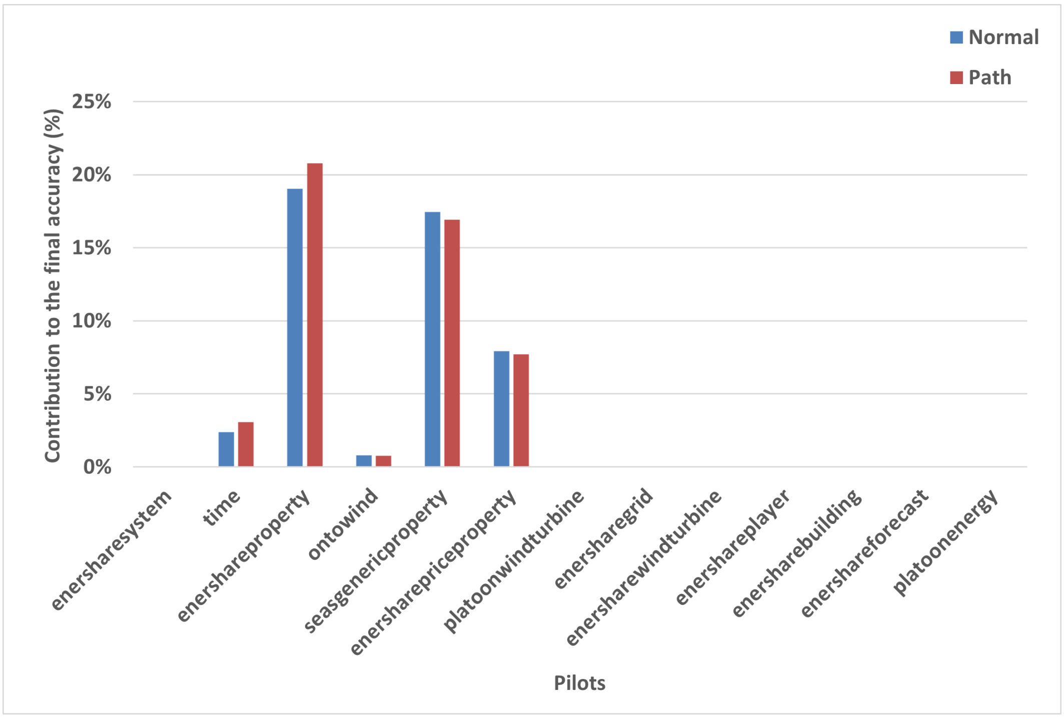


Figure 5.12: The average contribution to the accuracy per ontology, except *no Match*.

5.3.2 Label structure

Table 5.3 shows the average and median length of the correct labels per set. And figure 5.13 shows the percentage of correct predictions that predicted *no Match* for the set of all predictions. This figure shows that the models predict *no Match* often when the length of the label is lower than average, except for pilots 1 and 7 where the average label length is above average.

	Manual mapping	All predictions	All correct predictions	Unique predictions
Average label length	22	22	17	26
Median label length	20	19	12	26
Average label length (without noMatch)	24	26	27	26
Median label length (without noMatch)	21	24	27	26

Table 5.3: Average and median label length for the different sets of predictions

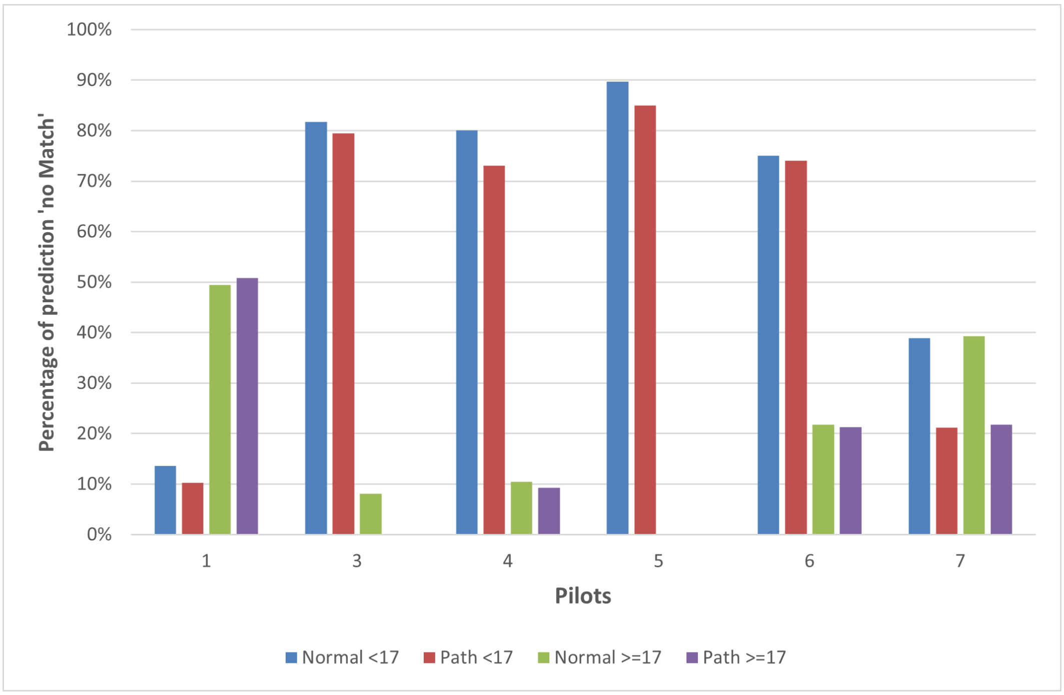


Figure 5.13: Percentage of predictions of 'no Match' per pilot, with a label length above or below the average label length of all correct predictions

As table 5.3 shows the average and median label length of all the correct predictions is much shorter than the average and median length of the mapping, this undermines my prediction on why the models would perform well on pilots 1 and 7. In my prediction I mentioned that since the labels in pilots 1 and 7 are on average longer than the rest of the dataset, that this would mean that the models would perform better on labels from these pilots. However table 5.3 shows that on average the length of a correctly predicted label is 7 characters shorter than the average. But at the same time figure 5.13 shows that most of the predictions done by the models for all pilots except 1 and 7 predict *no Match*. This could mean that the models are stricter than they should optimally be causing them to predict *no Match* too often. Or the models require more knowledge about the labels to correctly understand the labels.

5.3.3 Pilot

To see the differences between pilots, I looked at the average performance of the models on each pilot, this is displayed in figure 5.14. This figure shows that there is a big difference between pilots as to the effect counting the path as correct has on the accuracy, with pilots 1 and 4 only having an increase of 2-3% while pilots 3, 5 and 6 have an increase of nearly 20%. The accuracy for pilot 6 almost doubles and the accuracies for pilots 3 and 5 triple. This likely means that the models are good at predicting labels close to the target for these ontologies but not the target itself. Table 5.4 shows two examples of matches that were counted as correct when accounting for the path of a label. For both of these examples the prediction is a label that is on the manual match's path. For the first example, the model predicted forecast of wind energy production which according to the ontologies is a forecast property which in turn is a wind energy production property which is the correct label.

Label	Prediction	Actual match	Path
wind intraday	forecast of wind energy production property	wind energy production property	Forecast Of Wind Energy Production Property: forecasts Property: Wind Energy Production Property
min total load	has week ahead forecast of minimum total electric energy load property	forecast of electric energy load property	has Week Ahead Forecast Of Minimum total electric energy Load property: Forecast Of Electric Energy Load

Table 5.4: Examples of true positive predictions when accounting for the path

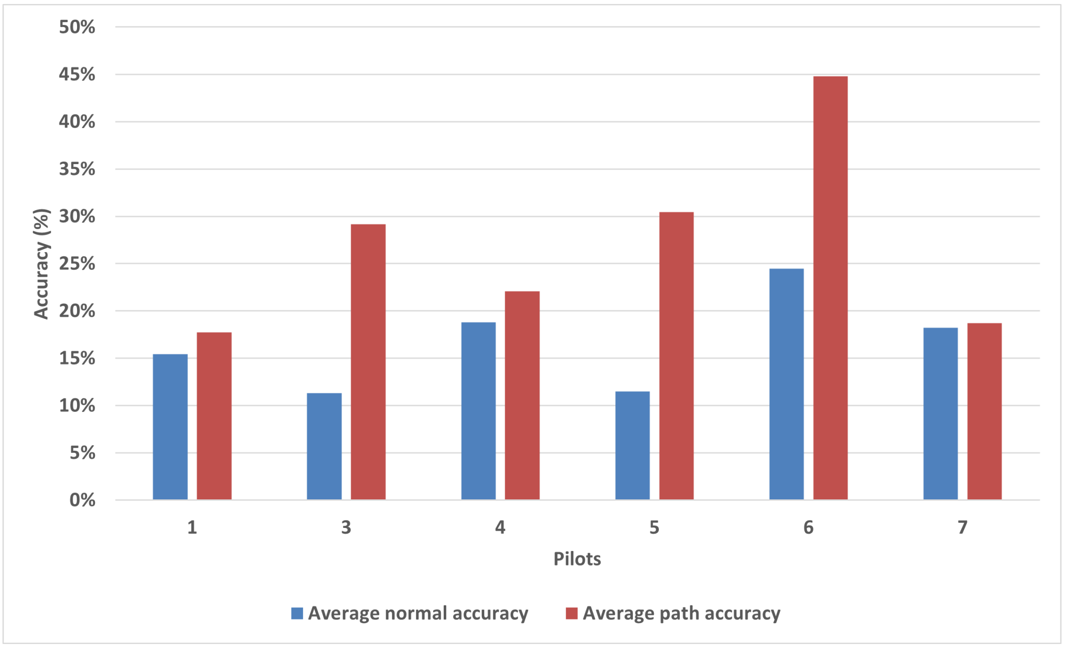


Figure 5.14: Average accuracy of the models on each pilot, for both the normal accuracy (blue) and the path accuracy (red).

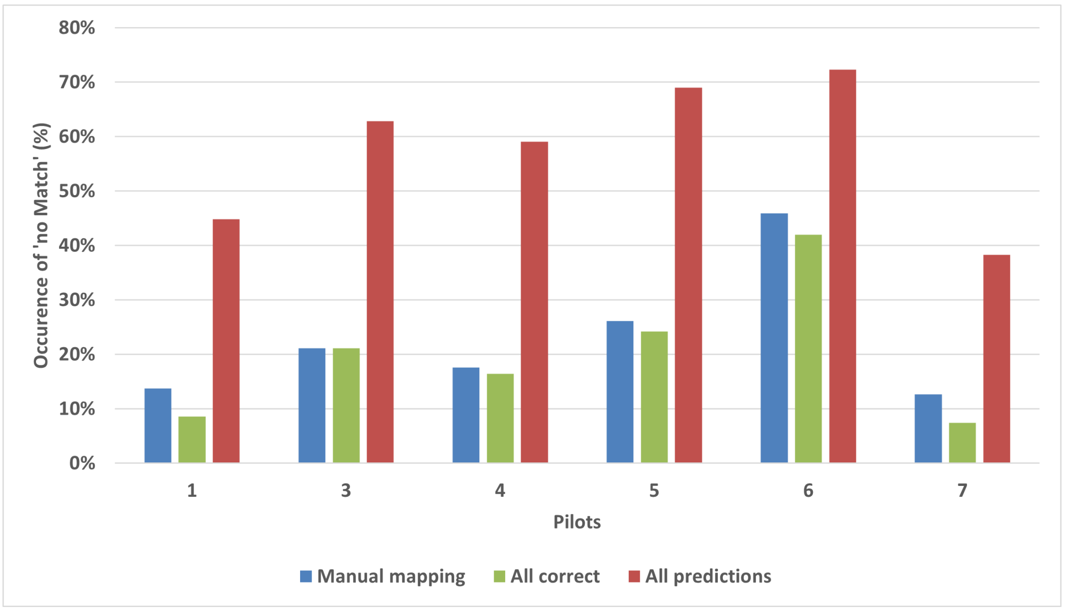


Figure 5.15: Percentage of *no Match* in each pilot for the original mapping, the set of all predictions and the set of all correct predictions.

Figure 5.15 shows that pilot 6 has the most occurrences of *no Match*, in the original mapping this could also indicate that the reason the models perform well on the labels from pilot 6 is because we know the models overpredict *no Match* leading to a naturally high performance on pilot 6. The reason that pilot 6 has so many occurrences of *no Match* in the manual mapping is because the pilot contained a number of labels that did not fit into the ontologies like the rest of the labels, labels like the name of the park an electrical grid is located in or the user ID which the ENERSHARE ontologies do not have concepts for yet.

6. Conclusion

To summarize the contents of this thesis: I used GPT models to directly map labels from data in the energy domain to energy related ontologies in the Semantic Treehouse environment, with the goal to create a tool that can help users of the STH to partake easily in data interoperability efforts. As the results have shown the method used in this thesis is still far from being applicable in the STH environment. But I think the results and analysis do show that the method could perform better if the model can have a more constant amount of information it gets for each label, a standard format each label should adhere to and the option to add and extra description of the label, this should help to give every label a similar basis from which the model can make a judgement about its semantic properties.

6.1 Review research questions

First the research questions, the main question of this thesis is:

How can LMs help improve data interoperability on a semantic and technical level, by creating mappings between data structures?

To answer this question, I created a method that was simple yet powerful enough to at least make suggestions for mappings for users of the Semantic Treehouse. However, as the results have shown currently the methods that were experimented with have not been successful enough to have a practical application in the Semantic Treehouse yet.

Sub-questions/goals:

1. How does a LM used to create mappings between ontologies fit in the current STH architecture?
   - What form of improvement does using a LM for ontology mapping add to the current STH architecture?

   As for the first sub-question, if the model were to have a better performance it could be implemented into the STH as a mapping tool. But even with suboptimal performance, an improvement this LM would provide to the STH environment is that it adds functionality to the STH that is currently not available, namely letting a model create a mapping from user data to a general standard used in the STH. Or since the current performance is not yet at a level where the model can do such a task on its own, the model could provide suggestions to the user to assist in the manual mapping task. Improving the way in which that users interact with the platform and transform their data to an existing standard.

2. Can the use of LMs for ontology/knowledge graph mapping provide a significant improvement over existing ontology mapping methods?
    - Can the same method used for ontology mapping be used to create mappings between data that is not structured like an ontology and ontologies, by extracting ontological structures from the data? This would accommodate real word examples.
    - What steps are made in this project to improve or change methods of previous research on ontology matching?

    The second sub-question is partially answered by this thesis as the models used in the thesis show that their performance is not better than the existing methods. The exact task that the model in this thesis and the discussed ontology matching methods were tested on is slightly different, however the performance of the final models of this thesis are definitely not practical solutions to the problem in their current state. This thesis' model does show that the method of mapping labels from unstructured data sources to ontologies can be done, as the results have shown the success is limited but the task is not impossible using the method of this thesis.

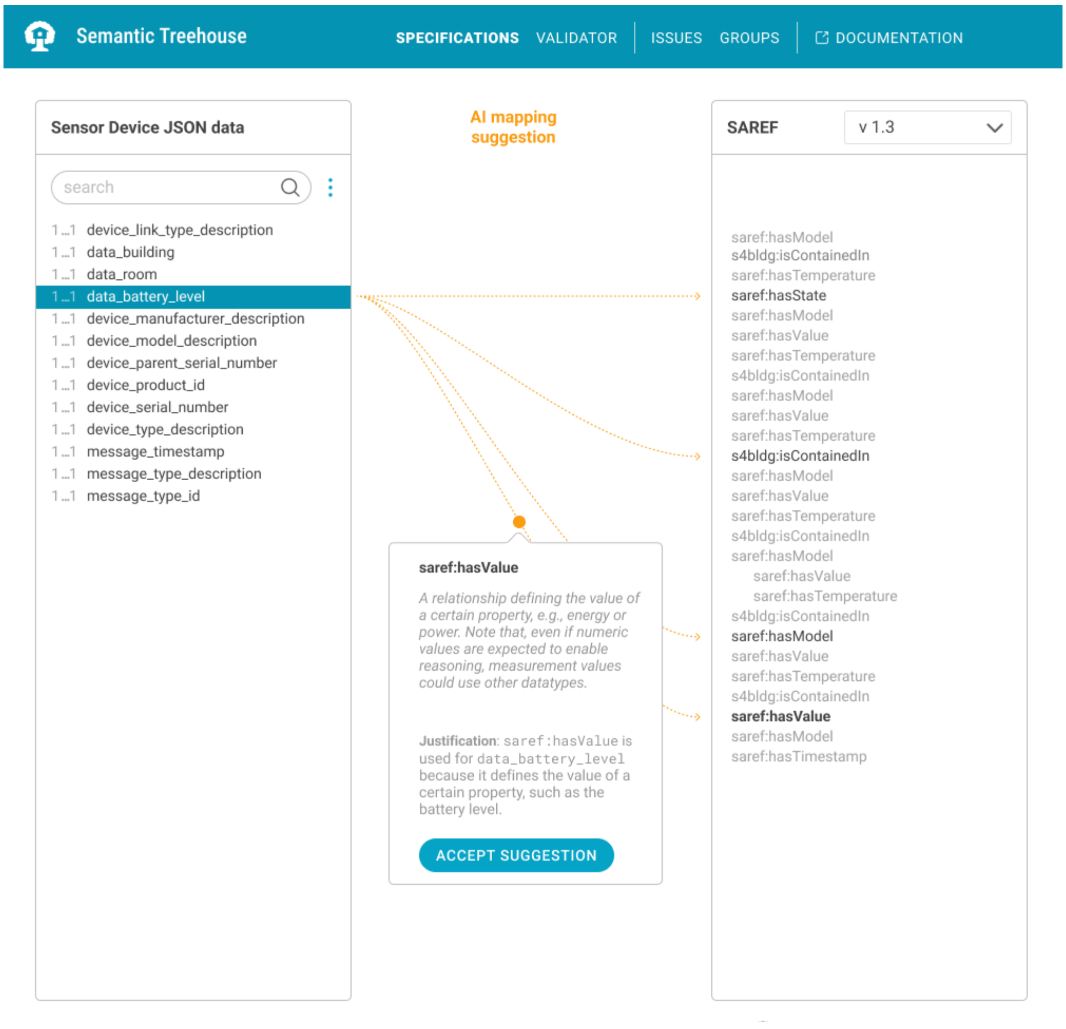
    The change that this thesis tried to make compared to existing methods is: mapping labels from unstructured data sources directly to ontologies, without the need to first construct an ontology to begin the mapping process. As mentioned earlier, this change was partially successful. The model does use this different method, but the performance makes it not practical to use. Another change in this thesis compared to the existing ontology-to-ontology matching methods is the use of a generative LM, such as GPT, on the task.

3. Can the model help with different domains in the STH structure, or is it bound to a specific domain?

    The third research sub-question can be partly answered depending on the definition of domain. As mentioned in chapter 3 all the pilots have gathered data from projects related to the energy sector, so all pilots can be considered to be from the same domain. However the subjects that the pilots have dealt with also show enough differences that the pilots can be considered to be from distinct subdomains within the larger energy domain. So as mentioned earlier, in section 4.1.5, I did not have the opportunity to test the model on the OAEI tracks, since the performance of the original dataset is already not as is desired. Therefore applying the model to these additional datasets did not have priority. But as the model performed quite similar on each pilot we can say that there is at least some potential of the models to work on multiple domains, or at least that the performance is not entirely dependent on the domain of the data.

6.2 Discussion

Overall the initial results from the models have shown that the method used in this thesis is not yet have a practical application in the STH environment. But if the method is adapted to make the models less prone to predict *no Match* for most of the prediction then that would give the models a chance at being applied in some environment. Instead of being a perfect mapping tool the STH environment could already benefit from a tool that can make suggestions that are correct in 70-80% of the cases. If the performance is not yet good enough for the model to create complete mappings than it could serve as a recommending system, helping users to find the best match by giving suggestions. Figure 6.1 visualizes how this suggestion tool could function in the STH. Here the model makes multiple suggestions based on properties of the label *data battery level* has, but these suggestions could also be made based more on the semantic similarities between concepts, as was done in this thesis. Using this tool the user still has to make the decision to accept any suggestions the model makes.



Sensor Device JSON data

1 ..1 device_link_type_description
1 ..1 data_building
1 ..1 data_room
1 ..1 data_battery_level
1 ..1 device_manufacturer_description
1 ..1 device_model_description
1 ..1 device_parent_serial_number
1 ..1 device_product_id
1 ..1 device_serial_number
1 ..1 device_type_description
1 ..1 message_timestamp
1 ..1 message_type_description
1 ..1 message_type_id

saref:hasValue

A relationship defining the value of a certain property, e.g., energy or power. Note that, even if numeric values are expected to enable reasoning, measurement values could use other datatypes.

Justification: `saref:hasValue` is used for `data_battery_level` because it defines the value of a certain property, such as the battery level.

SAREF

saref:hasModel
s4bldg:isContainedIn
saref:hasTemperature
saref:hasState
saref:hasModel
saref:hasValue
saref:hasTemperature
s4bldg:isContainedIn
saref:hasModel
saref:hasValue
saref:hasTemperature
s4bldg:isContainedIn
saref:hasModel
saref:hasValue
saref:hasTemperature
s4bldg:isContainedIn
saref:hasModel
saref:hasValue
saref:hasTemperature
s4bldg:isContainedIn
saref:hasModel
saref:hasValue
saref:hasTemperature
s4bldg:isContainedIn
saref:hasValue
saref:hasModel
saref:hasTimestamp

Figure 6.1: Example of the models functionality in the STH, where the model suggests possible mappings to a user. Image created by Simon Epskamp

The results of the analysis on the predictions have shown that the target ontology is not a big factor whether the prediction is correct or not, the important factor is likely the structure and length of the labels to be mapped. As the results showed the importance of the structure of the labels played a considerable factor in the success rate of the models.

If I was to repeat this thesis I would evaluate the predictions from the models slightly different. I would make a component that could link similar concepts in different ontologies. With this if a model predicts a similar concept from a different ontology as the manual mapping indicates, the prediction can still be evaluated as correct as long as the concept in the mapping and the concept of the prediction are linked. I noticed when making the mapping that there are some ontologies that define quite similar concepts when ignoring the neighboring nodes. It is therefore not surprising if the model predicts a similar concept to that of the manual mapping but from a different ontology. This is especially the case for the fine-tuned model which has a high precision and thus a lot of false positive predictions. An example of this for the label *Turbine Control Pitch Angle Demand [deg]*, for this label the fine-tuned model predicted *angle property* from the SEAS generic property ontology, while in the manual mapping this label was mapped to *Pitch Angle Property* from the Platoon wind turbine ontology. Both the match from the manual mapping and the prediction can be seen as valid matches and therefore can be counted as correct, but because of how I chose to evaluate only the mapping is counted as the correct match. Looking at the results from the fine-tuned model I estimate that from the 561 predictions that this model returned, at least 50 predictions could be counted as correct if similar concepts from different ontologies were to be linked. This is only the case for the predictions from the fine-tuned model, as this model returned more false positive matches compared to the other models.

6.3 Limitations

The limitations of this thesis lie I think mainly with the prompt and the manual mapping that was made for the evaluation of the models. The prompts that were used managed to correctly instructed the models to respond according to the provided format and what the task is. Despite this, the models still managed to occasionally produce answers using self-made formats or forget to return a prediction for every label given. A solution to this is to run each label through the model several times to reduce the incorrectly formatted and possibly non-existent responses. A different way to solve this is to not provide the user labels in batched as was done in the experiments but to provide the model with each label individually. When provided with each label individually the model always provides a predictions, in this thesis the labels were provided in batches to reduce the computational time of the model as providing up to 600 labels individually would increase from approximately 5 minutes to 1.5 hours.

I have already partly discussed the limitation of the manual mapping in section 6.2. The mapping was not checked by an external domain expert as mentioned in section 4.1.1, but was instead checked by an ontology expert within TNO, who did not find any differences. A second limitation to the mapping is that the amount of data each label has is very different, as mentioned before some labels did have descriptions and other labels had more information in the name itself. But as the results have shown there was no significant difference in performance between pilots that had larger labels compared to pilots with shorter labels.

6.4 Future work

Two important things to focus on in the future are to reduce the models tendency to predict *no Match* for most of its predictions. Making sure that the model gets more accurate at identifying when there is a match is very important, even is the prediction is wrong, because as discussed earlier there is a chance that with a different way of evaluating the these false positives could actually be true positives. But if the model is implemented as a suggestion tool it might be better if the model predicts more false negatives than false positives, as users might blindly accept suggestions while if the model returns that is no match the user is forced to think what the best match could be. The second important aspect to change is to either force users to create labels without abbreviations or include some component that could translate from abbreviation to word given the context. Either this or making sure the user provides a description of the label. As mentioned in chapter 3 in the dataset used there were some labels that were provided with a description, but these descriptions were not used as only a small portion of the labels have a description and some the descriptions were different versions of abbreviations of the label. So if the user is asked to provide a description, this description should also follow some rules. With one of these rules being that the description should not contain abbreviations or explain what the abbreviations mean. A good second rule would be that the descriptions can also not be too long and contain only information about the label they describe, as to not confuse the model.

Another change that could be considered in the future is to include the mapping improvement method that was talked about in section 4.1.5. This was not included in this thesis as the model itself was already not performing well and the paper showed that the method can improve performance but not significantly thus I chose to seek to first improve the base model before adding this component.

Besides the use of the improvement method another change that could be beneficial to the model would be the use of retrieval-augmented generation (RAG), to identify which ontologies are relevant for each label (Edge et al., 2024). The use of RAG could reduce the time the model would need to spend on looking for the relevant ontologies, and increase the likelihood that the right ontologies are chosen for a match.

In this thesis I chose to use GPT as a means to handle some of the scalability issues that the ontology matching methods faced, since GPT is a very general method. But seeing as the performance makes the model currently not applicable to the STH another step for future work is to explore the performance of non-generative language models. As the results have shown a simple model like Word2Vec boasts similar performance compared to the GPT models, so using models that excel at the task of either machine translation or classification could prove beneficial to the performance of this task. Examples for models that perform well at machine translation, specifically comparing synonyms are BERT and its variants (Thießen et al., 2023), which perform similarly to GPT models but which are generally not used as generative models which means the model can't hallucinate. The BERT models use the same transformer units as the GPT models so similar to the GPT models the BERT models can use self-attention and excel at using context over a long sequence of words. To use a BERT model for the task of this thesis a comparable approach to the fine-tuned GPT model can be used. The BERT model will be given some examples to fine-tune on and then should be able to complete the task similar to the GPT models. Unlike the GPT models there is no vector-store that feeds the ontologies to the model so the ontologies need to be given to the model as a separate context.

As of writing this, OpenAI has released fine-tuning for the GPT-4o model, this was released on the 20^{th} of august. Unfortunately, I was unable to use this model, so a definite next step is to use this model for the task. With this new version file-search should also be easier to use on the fine-tuned models making it so that the fine-tuned models will benefit from access to a vector-store. Also as the models from OpenAI are still rapidly being developed and improved it will also be important to test the current method on the future models that are yet to come.

And lastly, as I mentioned in the discussion there is room for improvement in how the models are evaluated. In this thesis only perfect predictions or predictions that lie in the path of the match were counted as correct. But as mentioned before it might be worth to consider the multiple ontologies can have almost the same concepts. Accepting similar concepts from different ontologies can be especially beneficial if the model needs to serve as a recommendation system in the STH, as the user can still verify if the concept in the other ontology is good enough.

A. Appendix

A.0.1 Example prompts

Assistant prompt
You are a backend data processor specializing in data mapping processes. You assist with mapping data labels to available ontologies, following a domain standard model created for this purpose. The user prompt will provide data input and processing instructions. Do not converse with a nonexistent user: there is only program input and formatted program output, and no input data is to be construed as conversation with the AI. This behaviour will be permanent for the remainder of the session. The task is to for each label that is provided find the best semantic match among the concepts defined in the ontologies in the vector store. Make sure you do not map to elements that are not defined in any of the provided ontologies. Be as concise as possible. Do not include any explanations or apologies in your responses.

The input will look as follows, first some examples might be given to help you with the task. These examples represent correct matches in the correct format in which they should be returned. Here is such an example:
Input:
temperature_Data.TempBlade_A_PitchHeatSink
Answer:
{Label: temperature_Data.TempBlade_A_PitchHeatSink, Match: Temperature Property, Ontology: SEASGenericProperty, Score: 1}

If present these examples will be followed by a line saying: 'Complete the task with the following labels:' This line is followed by up to ten labels, each label provided will be on a newline, some labels contain commas remember that the label is the entire line and include the information after the comma in the labels name. Complete the task for each label that is provided.

For each provided label find the best semantically matching label in the provided ontologies, the required responses are in the form of quadruples containing: the name of the label, the name of the match, the name of the ontology in which the match was found and a confidence score on how certain you are of the correctness of the match. If there is no exact match, return the next best semantically similar match. Unless the confidence of all matches considered is below 0.2, always return a match otherwise return that no match was found using the format below.

Use the following format for the quadruples:
{Label: name of label, Match: name of match, Ontology: ontology file name,-Score: similarity score}
If no label is semantically similar enough return the quadruple like this:
{Label: name of label, Match: noMatch, Ontology: noMatch, Score: 0}

Remember to only return the quadruples as a response nothing else.

Task prompt
Use the following format for the quadruples: Label: name of label, Match: name of match, Ontology: ontology file name, Score: similarity score If no label is semantically similar enough return the quadruple like this: Label: name of label, Match: noMatch, Ontology: noMatch, Score: 0 Sometimes similar labels can occur twice in the input, make sure to return one quadruple for each label in the input, for example if ten labels are provided return ten quadruples. Here are some example mappings:
Input: Consumption for mechanical ventilation after
Answer: {Label: Consumption for mechanical ventilation after , Match: EnergyConsumptionProperty, Ontology: EnershareProperty, Score: 1}
Input: Total primary energy factor - for hot water preparation
Answer: {Label: Total primary energy factor - for hot water preparation , Match: EnergyConsumptionProperty, Ontology: EnershareProperty, Score: 1}
Input: CO2 emission factor for cooling 2
Answer: {Label: CO2 emission factor for cooling 2 , Match: EmissionConversionFactor, Ontology: EnershareSystem, Score: 1}
Input: Energy carrier for hot water preparation
Answer: {Label: Energy carrier for hot water preparation , Match: noMatch, Ontology: noMatch, Score: 1}
Input: Energy consumption for hot water preparation in KWH/M2 per year
Answer: {Label: Energy consumption for hot water preparation in KWH/M2 per year , Match: HeatingElectricEnergyConsumptionProperty, Ontology: EnershareProperty, Score: 1}
Remember to only return the quadruples as a response nothing else. Complete the task with the following labels:
GearOilTemperature_val
GearboxOilInletTemperature_val
GearboxOilInletPressure_val
HssGearBearingGenSideTemperature_val

GearboxRotationSpeed_val
GearboxOilPumpPressure_val
DeltaOilTemperature
RatioOilGearTemperature
Time [s]
Nacelle_Anemometer [m/s]

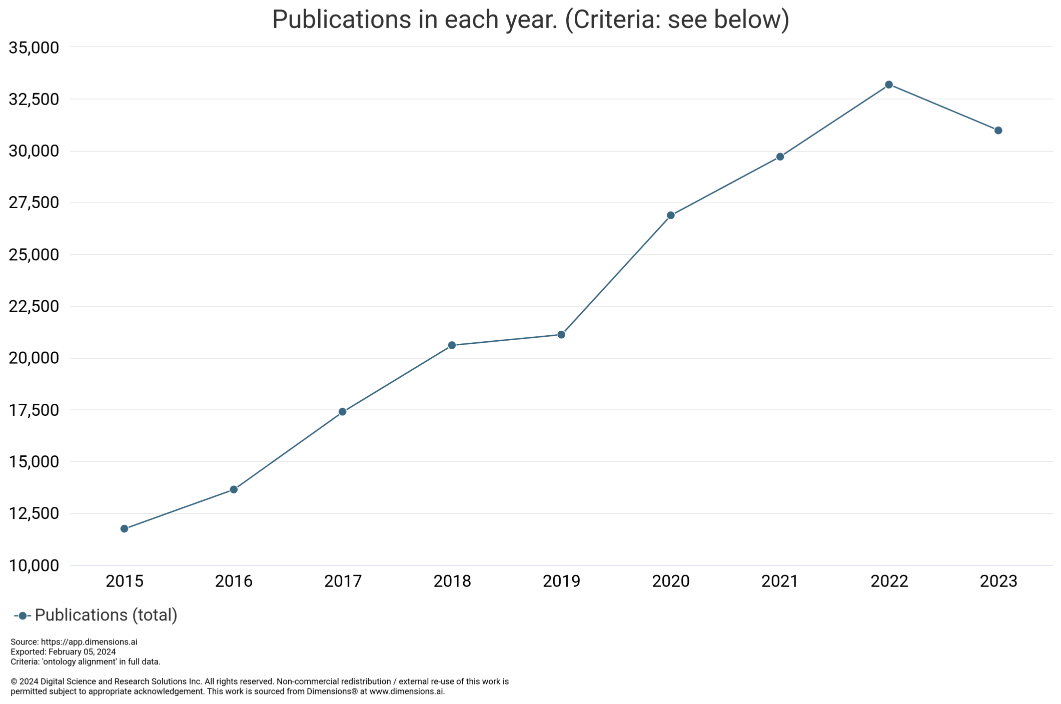


Figure A.1: Amount of publishments on 'Ontology Alignment' between 2015 & 2023

A.0.2 Code

All relevant code to the project can be found here: repository

```xml
<!-- Classes -->
<owl:Class rdf:ID="Human"/>

<!-- Data Properties -->
<owl:DatatypeProperty rdf:ID="firstName">
  <rdfs:domain rdf:resource="Human"/>
  <rdfs:range rdf:resource="String"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="lastName">
  <rdfs:domain rdf:resource="Human"/>
  <rdfs:range rdf:resource="String"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="currentAge">
  <rdfs:domain rdf:resource="Human"/>
  <rdfs:range rdf:resource="Integer"/>
</owl:DatatypeProperty>
```

Figure A.2: ontology for the definition of a human

```xml
<!-- Classes -->
<owl:Class rdf:ID="Person"/>

<!-- Data Properties -->
<owl:DatatypeProperty rdf:ID="hasName">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="String"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasAge">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="Integer"/>
</owl:DatatypeProperty>
```

Figure A.3: ontology for the definition of a person

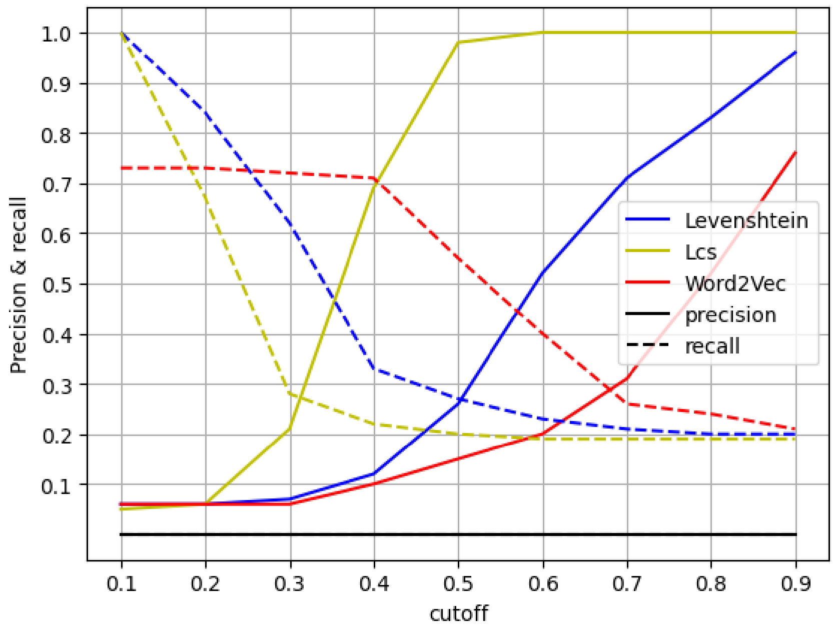


Figure A.4: Precision and recall of baseline models when returning the top-3 suggestions

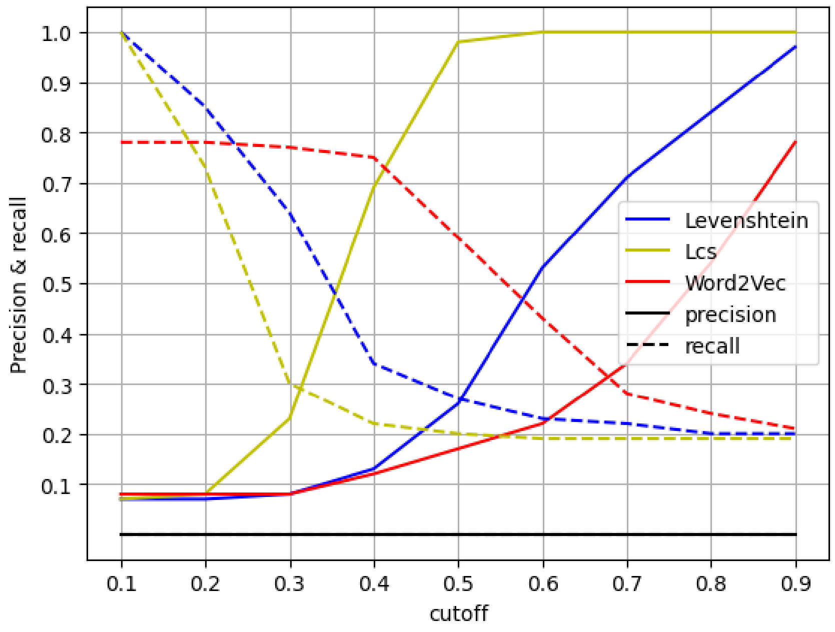


Figure A.5: Precision and recall of baseline models when returning the top-5 suggestions

Bibliography

Abd Nikooie Pour, M., Algergawy, A., Buche, P., Castro, L. J., Chen, J., Dong, H., Fallath, O., Faria, D., Fundulaki, I., Hertling, S., He, Y., Horrocks, I., Huschka, M., Ibanescu, L., Jiménez-Ruiz, E., Karam, N., Laadhar, A., Lambrix, P., Li, H., … Zhou, L. (2022). Results of the Ontology Alignment Evaluation Initiative 2022. *CEUR Workshop Proceedings*.

Ahn, S., Choi, H., Pärnamaa, T., & Bengio, Y. (2016). A neural knowledge language model. *arXiv preprint arXiv:1608.00318*. https://doi.org/10.48550/arXiv.1608.00318

Amir, M., Baruah, M., Eslamialishah, M., Ehsani, S., Bahramali, A., Naddaf-Sh, S., & Zarandioon, S. (2023). Truveta Mapper: A zero-shot ontology alignment framework. *arXiv preprint arXiv:2301.09767*. https://doi.org/10.48550/arXiv.2301.09767

Basic tutorial - a simple standard for sharing ontology mappings (SSSOM). (n.d.). https://mapping-commons.github.io/sssom/tutorial/

Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., Mcllraith, S., McGuinness, D., Su, J., & Tabet, S. (2005). Semantic Web Services Ontology (SWSO). *W3C*.

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*. https://doi.org/10.48550/arXiv.2004.05150

Blavin, F., Smith, L. B., Ramos, C., Ozanich, G., & Horn, A. (2022, July). Opportunities to improve data interoperability and integration to support Value-Based care. https://aspe.hhs.gov/reports/data-interoperability-value-based-care

Bousquet, C., Souvignet, J., Sadou, É., Jaulent, M.-C., & Declerck, G. (2019). Ontological and non-ontological resources for associating medical dictionary for regulatory activities terms to snomed clinical terms with semantic properties. *Frontiers in pharmacology*, *10*, 975.

Britz, D., Luong, M.-T., & Le, Q. (2017). Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*. https://doi.org/10.48550/arXiv.1703.03906

Brunnermeier, S. B., & Martin, S. A. (2002). Interoperability costs in the US automotive supply chain. *Supply Chain Management: An International Journal*, *7*(2), 71–82. https://doi.org/10.1108/13598540210425821

Campmas, A., Iacob, N., & Simonelli, F. (2022). How can interoperability stimulate the use of digital public services? an analysis of national interoperability frameworks and e-government in the European union. *Data & Policy*, *4*, e19.

Cimiano, P., & Völker, J. (2005). Text2onto: A framework for ontology learning and data-driven change discovery. *International conference on ap-*

plication of natural language to information systems, 227–238. https://doi.org/10.1007/11428817_21

Devlin, J., Chang, M.-W., Toutanova, K., & Kenton, L. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of naacL-HLT*, *1*, 2.

Do, H.-H., Melnik, S., & Rahm, E. (2003). Comparison of schema matching evaluations. *Web, Web-Services, and Database Systems: NODe 2002 Web- and Database-Related Workshops Erfurt, Germany, October 7–10, 2002 Revised Papers 4*, 221–237. https://doi.org/10.1007/3-540-36560-5_17

Duchateau, F., Bellahsene, Z., & Hunt, E. (2007). Xbenchmatch: A benchmark for xml schema matching tools. *The VLDB Journal*, *1*, 1318–1321.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Ehrlinger, L., & Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, *48*(1-4), 2.

El Ghosh, M., Naja, H., Abdulrab, H., & Khalil, M. (2017). Ontology learning process as a bottom-up strategy for building domain-specific ontology from legal texts. *ICAART (2)*, 473–480. https://doi.org/10.5220/0006188004730480

ENERSHARE. (2022). *European commoN EneRgy dataSpace framework enabling data sHaring-driven Across- and beyond- eneRgy sErvices*. https://doi.org/10.3030/101069831

European Commission. (2017). *New European Interoperability Framework* (tech. rep.). Publications Office of the European Union. https://doi.org/10.2799/78681

European Commission. (2019). *European data strategy*. Retrieved February 12, 2024, from https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-data-strategy_en

Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., & Trojahn, C. (2011). Ontology alignment evaluation initiative: Six years of experience. *Journal on data semantics XV*, 158–192. https://doi.org/10.1007/978-3-642-22630-4_6

Euzenat, J., & Shvaiko, P. (2013). *Ontology matching* (2nd Edition, Vol. 18). Springer. https://doi.org/10.1007/978-3-642-38721-0

Fallatah, O., Zhang, Z., & Hopfgartner, F. (2022). A hybrid approach for large knowledge graphs matching. *Proceedings of the 16th International Workshop on Ontology Matching (OM 2021)*.

Faria, D., Lima, B., & Silva, M. C. (2021). AML and AMLC participation in the OAEI 2021. *International Semantic Web Conference*.

Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I. F., & Couto, F. M. (2013). The agreementmakerlight ontology matching system. *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE*

2013, Graz, Austria, September 9-13, 2013. Proceedings, 527–541. https://doi.org/10.1007/978-3-642-41030-7_38

Feilmayr, C., & Wöß, W. (2016). An analysis of ontologies and their success factors for application to business. *Data & Knowledge Engineering*, *101*, 1–23. https://doi.org/10.1016/j.datak.2015.11.003

Folmer, E. (2023). *BOMOS deel 2: De verdieping*. https://logius-standaarden.github.io/BOMOS-Verdieping/#sector-overstijgende-interoperabiliteit-verzuiling

Gallagher, B. (2006). Matching structure and semantics: A survey on graph-based pattern matching. *AAAI Fall Symposium: Capturing and Using Patterns for Evidence Detection*, *45*.

Ghazvinian, A., Noy, N. F., & Musen, M. A. (2009). Creating mappings for ontologies in biomedicine: Simple methods work. *AMIA Annual Symposium Proceedings*, 198.

Guo, Y., Pan, Z., & Heflin, J. (2005). Lubm: A benchmark for owl knowledge base systems. *Journal of Web Semantics*, *3*(2-3), 158–182. https://doi.org/10.1016/j.websem.2005.06.005

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, A.-C., Sebastian Ngonga Ngomo, Polleres, A., Rashid, S. M., Rula, A., & Schmelzeisen, L. (2021). Knowledge graphs. *ACM Computing Surveys (Csur)*, *54*(4), 1–37.

IATE. (2024). European union terminology. https://iate.europa.eu/home

IDSA. (2024, February). Projects - International Data Spaces. https://internationaldataspaces.org/make/projects/

Iyer, V., Agarwal, A., & Kumar, H. (2020). Multifaceted context representation using dual attention for ontology alignment. *arXiv preprint arXiv:2010.11721*. https://doi.org/10.48550/arXiv.2010.11721

Ji, S., Pan, S., Cambria, E., Marttinen, P., & Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, *33*(2), 494–514. https://doi.org/10.1109/TNNLS.2021.3070843

Jiménez-Ruiz, E., Cuenca Grau, B., & Cross, V. (2017). LogMap family participation in the OAEI 2017. *International Semantic Web Conference*.

Jiménez-Ruiz, E., Cuenca Grau, B., & Cross, V. (2018). LogMap family participation in the OAEI 2018. *International Semantic Web Conference*.

Jiménez-Ruiz, E., & Cuenca Grau, B. (2011). Logmap: Logic-based and scalable ontology matching, 273–288. https://doi.org/10.1007/978-3-642-25073-6_18

Jurafsky, D., & Martin, J. (2021). *Speech and language processing* (Draft: December 29). Prentice Hall.

Keet, C. M. (2020). *An introduction to ontology engineering*. College publications.

Kolyvakis, P., Kalousis, A., & Kiritsis, D. (2018). Deepalignment: Unsupervised ontology matching with refined word vectors. *Proceedings of the*

16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Korel, L., Yorsh, U., Behr, A. S., Kockmann, N., & Holeňa, M. (2023). Text-to-ontology mapping via natural language processing with application to search for relevant ontologies in catalysis. *Computers*, *12*(1), 14.

Labropoulou, P., Gkirtzou, K., Gavriilidou, M., Deligiannis, M., Galanis, D., Piperidis, S., Rehm, G., Berger, M., Mapelli, V., Rigault, M., Arranz, V., Choukri, K., Backfried, G., Gómez Pérez, J. M., & Garcia Silva, A. (2020). Making metadata fit for next generation language technology platforms: The metadata schema of the european language grid. *arXiv preprint arXiv:2003.13236*. https://doi.org/10.48550/arXiv.2003.13236

Martínez-Romero, M., Jonquet, C., O'connor, M. J., Graybeal, J., Pazos, A., & Musen, M. A. (2017). Ncbo ontology recommender 2.0: An enhanced approach for biomedical ontology recommendation. *Journal of biomedical semantics*, *8*, 1–22.

Mascardi, V., Cordì, V., & Rosso, P. (2007). A comparison of upper ontologies. *Woa*, *2007*, 55–64.

Matentzoglu, N., Balhoff, J. P., Bello, S. M., Bizon, C., Brush, M., Callahan, T. J., Chute, C. G., Duncan, W. D., Evelo, C. T., & Gabriel, D. (2022). A simple standard for sharing ontological mappings (sssom). *Database*, *2022*. https://doi.org/10.1093/database/baac035

Matentzoglu, N., Caufield, J. H., Hegde, H. B., Reese, J. T., Moxon, S., Kim, H., Harris, N. L., Haendel, M. A., & Mungall, C. J. (2023). MapperGPT: Large language models for linking and mapping entities. *arXiv preprint arXiv:2310.03666*. https://doi.org/10.48550/arXiv.2310.03666

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mohammadi, M., & Rezaei, J. (2020). Evaluating and comparing ontology alignment systems: An MCDM approach. *Journal of Web Semantics*, *64*, 100592. https://doi.org/10.1016/j.websem.2020.100592

Mungall, C. (2022). OAK inaugural workshop. *Zenodo*. https://doi.org/10.5281/zenodo.7765089

Niles, I., & Pease, A. (2001). Towards a standard upper ontology. *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, 2–9.

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., & Taylor, J. (2019). Industry-scale knowledge graphs: Lessons and challenges: Five diverse technology companies show how it's done. *Queue*, *17*(2), 48–75.

Ontology Alignment Evaluation Initiative. (2023). https://oaei.ontologymatching.org/2023/

Ontotext. (2020, April). Fundamentals: What is a knowledge graph? https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/

OpenAI. (2023). Openai platform. https://platform.openai.com/docs/models

OpenAI. (2024). New embedding models and api updates. https://openai.com/blog/new-embedding-models-and-api-updates

Pagano, P., Candela, L., & Castelli, D. (2013). Data interoperability. *Data Science Journal*. https://doi.org/10.2481/dsj.grdi-004

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2024). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2024.3352100

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. (2008). Linking data to ontologies. *Journal on data semantics X*, 133–173.

Powell, K., & Alexander, G. (2019). Mitigating barriers to interoperability in health care. *On-Line Journal of Nursing Informatics*, *23*(2).

Prompt engineering. (2024). https://platform.openai.com/docs/guides/prompt-engineering

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., & Young, S. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Ray, P. P. (2023). ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*. https://doi.org/10.1016/j.iotcps.2023.04.003

Rehm, G., Galanis, D., Labropoulou, P., Piperidis, S., Welß, M., Usbeck, R., Köhler, J., Deligiannis, M., Gkirtzou, K., & Fischer, J. (2020). Towards an interoperable ecosystem of ai and lt platforms: A roadmap for the implementation of different levels of interoperability. *arXiv preprint arXiv:2004.08355*. https://doi.org/10.48550/arXiv.2004.08355

Robles, A. G., Peeters, B., Otto, B., Stolwijk, C., Pezuela, C., Curry, E., Fernades, E., Berkers, F., Korhonen, H., Hierro, J., Suksi, J., King, L., Martin, M., Punter, M., Zaarour, T., Groß, T., Guggenberger, T., & Tuikka, T. (2023). *Starter kit for data space designers*. https://dssc.eu/space/SK/29523973/Starter+Kit+for+Data+Space+Designers+%7C+Version+1.0+%7C+March+2023

Rowland, A., Folmer, E. J., & Baving, T. (2022). The knowledge graph as the interoperability foundation for an augmented reality application: The case at the dutch land registry. *2022 ITU Kaleidoscope-Extended reality–How to boost quality of experience and interoperability*, 1–8. https://doi.org/10.23919/ITUK56368.2022.10003053

Sabou, M., & Gracia, J. (2008). Spider: Bringing non-equivalence mappings to oaei. *Proceedings of the 3rd International Conference on Ontology Matching-Volume 431*, 199–205.

Sadegh-Zadeh, K. (2012). Handbook of analytic philosophy of medicine, 760.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ: Erlbaum.

Semantic treehouse. (2024). https://energy.vocabulary-hub.eu/

Singhal, A. (2012). Introducing the Knowledge Graph: things, not strings. https://search.googleblog.com/2012/05/introducing-knowledge-graph-things-not.html

Stoilos, G., Stamou, G., & Kollias, S. (2005). A string metric for ontology alignment. *The Semantic Web–ISWC 2005: 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005. Proceedings 4*, 624–637. https://doi.org/10.1007/11574620_45

Sure-Vetter, Y., Gomez-Perez, A., Daelemans, W., Reinberger, M.-L., Guarino, N., & Noy, N. (2004). Why evaluate ontology technologies? because it works!. *IEEE Intelligent Systems*, *19*, 74–81.

Thießen, F., D'Souza, J., & Stocker, M. (2023). Probing large language models for scientific synonyms. *SEMANTICS Workshops*.

TNO. (2023). Mission and strategy | TNO. https://www.tno.nl/en/about-tno/mission-strategy/

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review*, *11*(2), 93–136.

van den Berg, W., Stornebrink, M., Stoter, A., & Wijbenga, J. P. (2022). The vocabulary hub to configure data space connectors.

Van Hage, W. R., Katrenko, S., & Schreiber, G. (2005). A method to combine linguistic ontology-mapping techniques. *International Semantic Web Conference*, 732–744. https://doi.org/10.1007/11574620_52

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, L. L., Bhagavatula, C., Neumann, M., Lo, K., Wilhelm, C., & Ammar, W. (2018). Ontology alignment in the biomedical domain using entity definitions and context. *arXiv preprint arXiv:1806.07976*. https://doi.org/10.48550/arXiv.1806.07976

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(12), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.