Ontology Matching with Wikipedia

Bachelor-Thesis von Sven Helmuth Hertling aus Worms
November 2012

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Ontology Matching with Wikipedia

Vorgelegte Bachelor-Thesis von Sven Helmuth Hertling aus Worms

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Heiko Paulheim

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den November 7, 2012

(Sven Hertling)

Zusammenfassung

Das derzeitige Internet verfügt über eine große Anzahl Informationen, die allerdings nur für den Menschen lesbar und verwertbar sind. Suchmaschinen wie Google oder Yahoo leisten gute Dienste, um Informationen zu finden. Dies wird allerdings mit statistischen Verfahren bewerkstelligt, die über keine Bedeutung (auch Semantik genannt) verfügt. Am Beispiel von Wörtern, die mehrere Bedeutungen haben (Homonyme), wird dies sehr schnell klar.

Das Semantik Web (Web 3.0) versucht daher Informationen mit ihrer Semantik zu hinterlegen, um diese Informationen auch für Computer nutzen zu machen. Diese Semantik wird in sogenannten Ontologien abgespeichert, die generelle Konzepte der realen Welt abbilden. Ein Beispiel für ein solches Konzept ist eine Person mit ihrem Vornamen, Nachnamen und der Adresse.

Wenn mehrere Benutzer die gleiche Ontologie verwenden, kann man die Informationen maschinell und korrekt zusammenführen und austauschen. Auch wenn sich die Ontologien, zum Beispiel durch eine andere Sprache, unterscheiden ist es möglich sogenannte *Mappings* zu definieren. Diese *Mappings* sagen aus, dass zwei Konzepte dieselbe Semantik haben. Ein Mapping könnte zum Beispiel folgendermaßen aussehen: Person <-> Human, sowie Adresse <-> address. Das Automatische Auffinden solcher Relationen wird *Ontologie Matching* genannt.

Diese Arbeit beschäftigt sich damit, ähnliche Konzepte von zwei Ontologien durch Wikipedia zu finden. Die grundlegende Idee ist, dass zwei Beschreibungen von Konzepten semantisch gleich sind, wenn die Suche von Wikipedia möglichst viele äquivalente Artikel zurückliefert.

Um auch Ontologien in zwei verschiedenen Sprachen vergleichen zu können, werden alle sogenannten *Language Links* der Wikipedia benutzt. Diese verlinken alle Artikel einer Sprache mit den passenden Artikeln anderer Sprachen.

Abstract

Today's world wide web consists of a huge amount of human readable information. Quickly finding the right information becomes a difficult job over time. Search engines like Google or Yahoo doing their job well to find similar documents related to the search term. This is done by the help of statistical approaches, that do not contain any meaning (also called semantics) at all. This becomes clear quickly when using words with more than one meaning (homonyms).

The idea of the *semantic web* sometimes also called *Web 3.0*, is to create content with the meaning (semantic). This semantic information is stored in so called ontologies. They map general concepts of the real world such as a person with their first name,last name and address.

If multiple users utilize the same ontology, information can be merged correctly and exchanged in an automatic way. Even if the ontologies differ by a different natural language, it is possible to define mappins between ontologies. If two concepts from two different ontologies are mapped together, this means that they have a similar semantic. A mapping might look like this: Person <-> Human as well as Adresse <-> address. The automatic discovery of such relations is called *Ontology Matching*.

This thesis deals with finding similar concepts with the help of Wikipedia. The basic idea is that two descriptions of concepts are semantically equivalent, if the search engine of Wikipedia returns many similar articles.

In order to compare ontologies in different natural languages, all so-called *language links* are used. These links contain all article titles in every language where the article has a correspondence.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problems of current world wide web

Today, the internet offers much information which sometimes can be more confusing than helpful. It relies on *HTML* (Hypertext Markup Language) which can be interpreted twice. From a human point of view, it contains information between some tags. These tags don't contain any data for humans but rather for machines. They describe how the data is displayed. This is the only static info in a document that a computer can process. The main problem here is that the current web is only human oriented.

The amount of data is growing every day, but finding the required piece of information is often difficult , even if it is present in the web. Search engines like Google, Bing or Yahoo are doing their job very well with the help of statistic approaches and algorithms like *PageRank*(Page et al., 1999). Essentially the web is based on string comparisons with no *semantic* (meaning) at all. It is only possible to search for single keywords instead of full questions or sentences. A more content based search would be desirable.

Another problem for full text search engines are homonyms and synonyms (Paulheim, 2011a). Homonyms are syntactically equals, but have other semantics. A context makes this clear and should be also processed by a search engine. In case of synonyms two words are syntactically quite different, but have the same meaning. To find more results in a query, keywords can thus be replaced by their synonym.

The structure and organization of the web is highly decentralized (Hitzler et al., 2008). This goes along with heterogeneous data. Users can publish their sites with different encodings, file formats and natural languages. These pages can contain videos, pictures with text, music and so on. Often in videos, a playback is not supported on all platforms and browsers, because of missing decoders or proprietary software.

It is also possible that information is distributed over the internet and is not localized on one page. Today's search engines aren't able to connect multiple information from different pages together to a whole. Such facts are not available altogether, but it results as a consequence of multiple facts. Neither data nor information are in relation to each other.

Historically, there exist two ways to overcome all these issues. The first one is to transform and enrich the current web with the help of *machine learning*. It is better called *web mining* (Chakrabarti, 2002) and belongs to the domain of *data mining*. Basically the approach tries to enrich the content of a website automatically with pattern recognition, information retrieval and other technics of *data mining*.

It is subdivided into *Web Content Mining* (which is extracting information directly from the web page), *Web Structure Mining* (which belongs to the structure of links between websites) and *Web Usage Mining* (which discovers user access patterns) (Cooley et al., 1997) (Kosala and Blockeel, 2000). With *Web Content Mining* it is possible to receive information with annotated semantics. In case of heterogeneous data like the current web pages, it becomes more difficult to extract the meaning of paragraphs.

As an counterpart *semantic web* has been established. It is described in more detail in the next section 1.2.

1.2 Semantic Web

To solve the problems of the current web, a new technology has evolved. The approach of the *semantic web*, also called *web 3.0* (Hendler, 2009), is to enrich the current content manually in advance with formal languages. The basic idea is the ability to reason and query complex requests. With new (meta-)information computers can interact with them in a way that humans could think of a meaningful behavior (Hitzler et al., 2008). To achieve this goal, standards have been established. Most languages in use like *RDF* (Resource Description Framework) and *OWL* (Web Ontology Language) are endorsed and published by the *World Wide Web Consortium* (W3C) [1]. Nevertheless, there are other languages for the

[1] http://www.w3.org/TR/owl2-overview/

semantic web. In a survey with 627 participants at the beginning of 2007 it is shown that 71.9 % are neither OWL nor RDF(S) (see figure 1) .

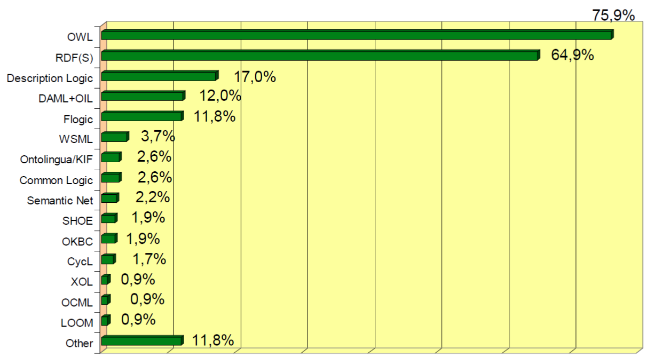


Figure 1: Ontology languages used by 627 participants in a survey in 2007 (Cardoso, 2007)

With RDF, for example, it is possible to describe a simple statement with subject, predicate and object. An object can also be a subject in another statement and so on. Finally, there is a whole graph of statements. In this graph all vertices are subjects or objects of a statement. They are connected with directed edges (also called properties) representing a predicate. Each of these statements are called *RDF-Triples*. All subjects and predicates are *resources* and are identified through an *uniform resource identifier* (URI). An URI is a more general concept than an *uniform resource locator* (URL), which is only a locator for sites instead of a real identifier. Every object in the real world is represented by different URIs. But different URIs do not indicate that they mean different things. Perhaps two users create other URIs for the same real thing. This happens in direct conclusion of the *non-unique name assumption* (Hustadt, 1994). The main reason for allowing this, is another important principle in the semantic web:

Anyone can say Anything about Any topic. (AAA principle) (Allemang and Hendler, 2011)

Anyone should be able to create his own identifiers and statements and should not be restricted by other users. To overcome the issue that two different URIs mean the same thing, it is possible to create a relation between them called *owl:sameAs*.

Right now, there are only resources in these RDF statements. Another type are *literals*. They contains basic types and values like numbers or strings. They can only be an object of a statement and as a result, literals have only incoming edges.

A RDF statement like

$$\text{John plays football.} \tag{1}$$

can be interpreted as a labeled, directed multi-graph (see figure 2) or a simple statement like (1).

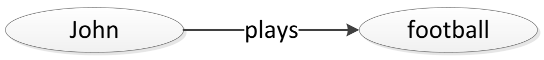


Figure 2: A Graph that displays a simple RDF statement. In this case the fact that John plays football.

It can be serialized developer friendly in N3 (Notation3) and XML. N3 is not a standard but often used, because it is more human-readable. For better machine processing, XML is preferred. With Unicode as encoding in mind, the bottom of the so called *semantic web stack* is established (shown in figure 3). The first appearance of the stack was in 2000 [2] made by Tim Berners-Lee and was concretized in 2005 [3] and 2007 [4].

[2] http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html
[3] http://www.w3c.it/talks/2005/openCulture/slide7-0.html
[4] http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb

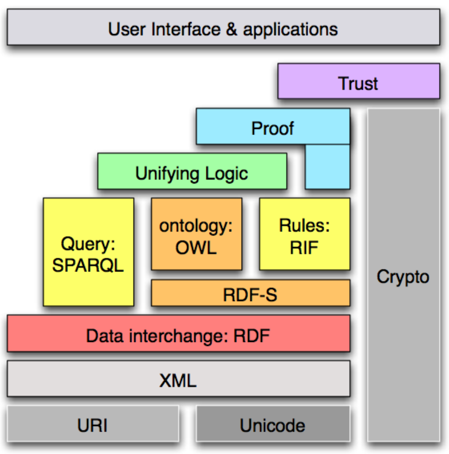


Figure 3: Semantic Web Stack (Berners-Lee, 2009)

On top of RDF there is a query language called *SPARQL* (**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage) and a rule language *RIF* (**R**ule **I**nterchange **F**ormat). Another block are ontology languages like OWL or RDF-S described in more detail in section 1.3. All other blocks like *unifying logic*, *proof* and *trust* are still objects of research.

1.3 Ontologies in a nutshell

With RDF it is possible to write simple statements with subject, predicate and object in a formal way. However, one important thing is missing: there are still no real semantics at all. To get an behavior described in the first published idea of the semantic web by Tim Berners-Lee (Berners-Lee et al., 2001) more meaningful information have to be added .

So problems denoted in section 1.1 can not be solved. One big step towards a meaningful tagging of data are Ontologies. They describe concepts and relations between data for a special purpose. In a more formal way, there are classes and properties (Noy and McGuinness, 2001).

Classes represents every abstract type. For example, a class could be *book*. A concrete book is an *instance* of the class. An instance can have more than one type. It allows also multiple inheritance of classes. Every class can be structured hierarchically in subclasses and superclasses. If x is a subclass of y it means that x is more specific than y. The other way around, this also applies for superclasses.

Properties connect classes and have a domain and a range, describing between which classes this property can exist. They can also be arranged in a taxonomic hierarchy, which means they can also have subproperties and superproperties. In contrast to *object-oriented programming*, properties have not be defined with classes and can exist without explicit domain and range. They describe only relations between independent classes.

Moreover, there are two types of properties - object properties and data properties. Object properties are defined between two classes. On the other side, data properties can only have literals as range. Literal can only have incoming edges and not outgoing ones. This implies that they are the leaves of the graph and can only contain primitive types like numbers or strings.

The two most used languages are RDFS and OWL. OWL offers more language constructs than RDFS like intersections and unions between classes and so called restrictions for properties. OWL has three sublanguages, namely OWL Lite,OWL DL and OWL Full. OWL Lite is a subset of OWL DL which is itself a subset of OWL Full. Thus every ontology in OWL Lite is also in OWL Full. All queries for OWL Lite / DL are decidable whereas OWL Full allows more features like that a class can also be used as a instance.

1.3.1 Example ontologies

A simple example for an ontology is *dublin core* [5]. It has some properties for general use. This includes properties like creator, publisher, subject and title.

In *RDFS* there are some predefined properties that are often used. An ontology is useless when it is not widely used. The buzzword here is interoperability. Building ontologies only makes sense when other users or programs are also able to understand the data. To find out the meaning, it is necessary to use the same ontology or some mapping between them. These mappings can be found in a (semi-) automatic way, which is more or less efficient. More about this topic is found in section 1.4.

An often used property of *RDFS* is *label*. A label describes the class, respectively the resource with meaningful strings. These strings can be annotated with language marks like *de* or *en*. One resource can also be tagged with multiple labels.

To describe a resource in more detail a *comment* should be taken. These comments contain many strings and thus can also have language tags for different natural languages. To refer to other pages or resources a tag called *seeAlso* is usable.

It is also possible to create an URI with a meaningful string in it (like `http://www.example.com#germany`), but it doesn't have to be. Labels and comments are especially designed to include such strings. Therefore a resource can also have an URI like `http://www.example.com/123456`. With an label like "germany" the semantic of it should be clear. Classes and properties can also be provided with these *RDFS* tags.

Another important point of an ontology is that every simple and easy statement has to be made explicit. These logical restrictions (often implicit) should also be formalized in an ontology to give them more semantics.

1.3.2 Inference

So far we have ontologies describing concepts of specific domain with classes and properties. This is called *T-Box* (terminological component) (Hitzler et al., 2008, p.167). On the other side we have some facts, respectively statements in the *A-Box* (assertion component). It is also known as *knowledge base* (kb).

Now it time to bring these two components together. A simple fact enriched with ontologies should result in new facts. The more explicit knowledge an ontology has, the more facts can be inferred.

There are three types of inference. The first one is *deductive reasoning*. It links premises (preconditions) with conclusions (logical consequences). In case of reasoning on ontologies, this is the most applicable method. A famous example for deductive reasoning is the following:

$$\text{All men are mortal.(T-Box)} \tag{2}$$
$$\text{Socrates is a man.(A-Box)} \tag{3}$$
$$\text{Socrates is mortal.} \tag{4}$$

In this case, the premises are on the one hand the *T-Box* (2) and on the other hand one fact of the *A-Box* (3). The conclusion (4) that Socrates is mortal is true if - and only if - all premises (2) and (3) are true. This applies especially for the ontology. If something is false, the whole conclusion is not true. In this example the conclusion (4) would not be true, if not all men were mortal. In this case it is easy to see that all premises are true, but in a large and unimaginable ontology it is not that simple. Deductive reasoning is the most used type of inference, because it always yields in true results. Tools doing this job are called *reasoner*.

The second type is *inductive reasoning*. With this method there are new rules instead of new facts produced. With the help of the knowledge base, new rules can be learned. It is not guaranteed that all rules are true for all facts. It can only be as good as the quantity and quality of the A-Box. It makes

[5] `http://dublincore.org/documents/dces/`

only sense to learn some rules, if enough facts about a specific concept are available. *Ontology learning* can be realized with an association rule learning approach from *data mining* research (Maedche and Staab, 2001). This can be implemented with the *APRIORI* algorithm. An association rule is extracted of a transaction which contains multiple items. The form of such a rule is

$$A \rightarrow B. \tag{5}$$

A and B are sets of items (Kavšek et al., 2003). If many items from A are in a new transaction C, than the set B is also often included in C. The algorithm is initially developed for *market basket analysis*. In this scenario a transaction is a shopping basket and the items are household items. The association rule (5) can then be substituted with $A = \{bike\}$ and $B = \{bikelock\}$. If someone buys a bike they usually acquire a bike lock, too. Association rules are also used in *recommender systems*. They suggest to buy another book in a online shop, that other users also bought. The whole approach can be transferred to learning hierarchies of classes. The transaction is replaced with an instance. The items are all types (classes) of this specific instance. For example, Julia (an instance) is a type of *woman* and *person*. Stephen instead is a *man* and a *person*. If there many of these examples, rules like

$$woman \rightarrow person \tag{6}$$
$$man \rightarrow person \tag{7}$$

can be learned (Paulheim, 2011b). These rules are then interpreted as subclasses. If an instance has the type woman (6) or man (7), it has also the type person. As a result, woman or man are subclasses of person. Domain and range can also be learned. The transaction is still one instance but the relations are also added with a tag if it is a domain or a range.

The third variation of inference is *abduction*. Premises are on the one hand that all men are mortal (2) and on the other hand that Socrates is mortal (4). The diagnosis is, that Socrates is a man (3). This type is less in use, because it is very prone to error.

1.3.3 Jena - A programming framework

To read ontologies from files, a programming framework is required. It is also possible to read the files manually with an XML parser, but a full framework offers more features like an *API* for reasoning. There are two really different approaches for representing and editing an ontology.

The first one is a direct programming framework. All classes in the ontology exists also as classes in the programming language. So the first step has to be generating code for a specified schema. *RDFReactor* can, for example, generate java classes from an ontology (Völkel and Sure, 2005). The resulting code of an application is much more readable with this type of framework. All classes have spelling names and can be manipulated through java methods with java primitives.

One problem of this approach is that it is only useful when creating new software including the generated classes. A solution for existing classes can be annotations (see otmj in (Quasthoff et al., 2010)) or specifying a mapping between the programming logic and the ontology in a separate file (Paulheim, 2011c). Through differences in expressiveness and basic ideas of object-oriented programming and ontologies, it is often not possible to find a simple and working correspondence between the two languages (Paulheim et al., 2011). These inaccurate mappings have the disadvantage that an import or export to RDF can yield in strange behavior, if it is possible at all. Using this type of programming framework is great when the application is domain specific and can use generated classes from an ontology. The resulting source code is more readable and maintainable.

For domain independent applications another framework should be taken into consideration. These tools support only general concepts like classes, properties and resources. This yields in more and incomprehensible code, but it is applicable for different domains and purposes. An often used framework is *Jena* (Carroll et al., 2004). It is an open-source project developed in java since 2000 by HP Labs. In

November 2010 it was adopted by the Apache Software Foundation [6]. With *Jena 2*, released in August 2003, the framework can handle RDFS and all three versions of OWL.

The core is the *RDF Graph*, which is a set of RDF triples (Klyne and Carroll, 2004). Through the interface of the graph multiple components are build. One of them is a storage component that allows to save models on hard disk with the help of triple stores respectively databases. As a query language, Jena provides SPARQL. It is also used to query SPARQL endpoints in the semantic web. The framework includes also a RDFS and a OWL reasoner to infer more facts. For more performance or different approaches, external reasoners can be connected with an API. Pellet is one example for such an external resource with an additional feature called entailment support (Sirin and Parsia, 2004). The last feature mentioned here is the opportunity to process rules. So, most of the semantic web stack technologies can be handled by this framework.

1.4 Ontology Matching

The vision of *semantic web* is to reason on large data and query complex requests. Every website should contain information with annotated semantics. Moreover it shouldn't be restricted, that anyone can use his own ontology. When creating an ontology it is may be not clear that a similar ontology already exists. The imagination of a formalized specific domain can also vary.

Another reason for using his own ontology is to create new classes with other properties and structure. By connecting the ontologies a feature called *interoperability* can be realized. Many users can interact with each other although the ontologies themselves may differ. The process of finding such connections (mappings) is called *ontology matching*.

1.4.1 Types of heterogeneity

Ontologies can differ on a structure and/or semantic level. This problem also occurs in databases and is already discussed by (Sheth and Kashyap, 1992) and (Kim and Seo, 1991). A more detailed look at ontologies is given by (Klein, 2001).

The differences are split up into *language level mismatches* (syntactic) and *ontology level mismatches* (semantic). Figure 4 shows this overview.

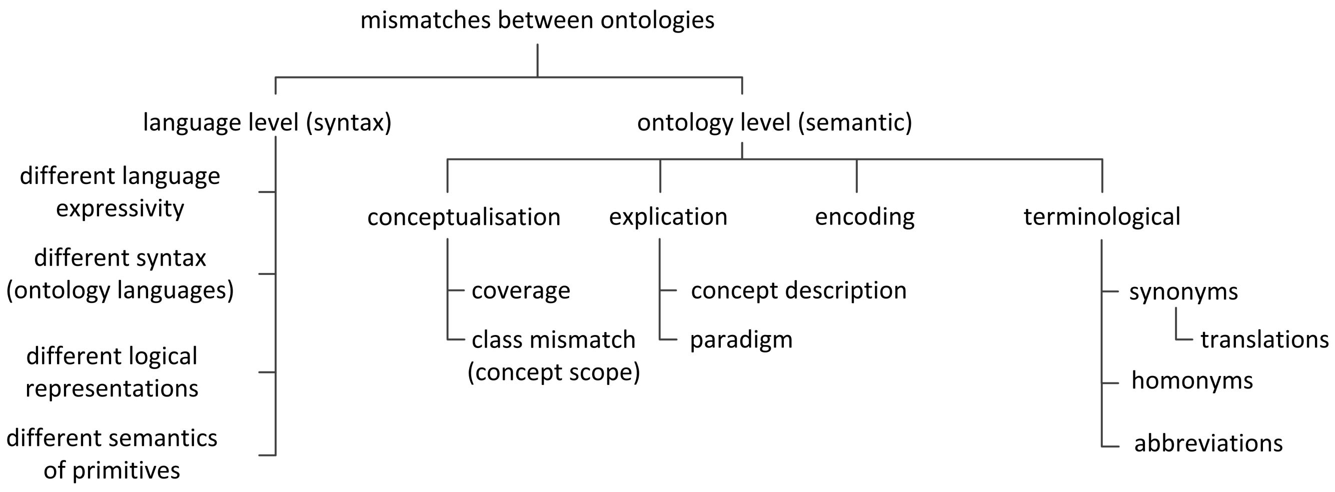

mismatches between ontologies
language level (syntax)
ontology level (semantic)
different language expressivity
different syntax (ontology languages)
different logical representations
different semantics of primitives
conceptualisation
explication
encoding
terminological
coverage
class mismatch (concept scope)
concept description
paradigm
synonyms
translations
homonyms
abbreviations

Figure 4: Mismatches between ontologies. It is split up into language level (syntax) on the one hand and ontology level (semantic) on the other hand (based on (Klein, 2001)).

A language level mismatches can be for example a different ontology language like *OWL* or *RDFS*. They differ in their syntax. Furthermore they don't have the same expressivity. RDFS, for example,

[6] `http://jena.apache.org/about_jena/about.html`

is only a subset of OWL. Another problem are different logical representations even if they are logical equivalent. The last point are varying semantics of primitives. The atomic statements of the language can be interpreted differently (Klein, 2001). These types of heterogeneous data is not the focus of this work. However, the tool described in section 3 can also process different ontology languages (OWL and RDFS) with the help of a framework like Jena (see section 1.3.3).

The other important mismatch of ontologies is on the ontology, respectively semantic level. It is divided into conceptualization, explication and terminological mismatches as well as encoding issues illustrated in figure 4.

Conceptualization mismatches contain different coverages and granularities between ontologies. Two ontology modelers will describe one concept in different detail levels. For example, ontology one describes only humans, whereas for ontology two the difference between male and female is important and necessary. Another problem is a class mismatch (Visser et al., 1997). Two classes seem to be semantic equals but have other instances. In (Klein, 2001), this is called scope.

Explication differences arise whe two classes are distinct with a qualified attribute or explicitly ordered with a sub-hierarchy. In (Chalupsky, 2000) it is called *modeling conventions* whereas (Klein, 2001) describe it as *concept description*. Two ontologies can use different *paradigms* for time, causality and so on. It is also possible to think of different representations of places. One describes it in a geographic coordinate system with latitude and longitude and another with the *Universal Transverse Mercator* system (UTM).

Terminological mismatches describe terms in two ontologies that differ in their syntax but have similar meaning. These synonym terms should be matched together although the strings have nothing in common. Homonym tags instead are syntactically equals, but can be used in other contexts and meanings. These tags often matched by a simple string comparison whereas they have another context in which they occur.

The last mentioned type of difference is *encoding*. Values in a label can expressed with different formats. Dates can be represented for example in various orders like *YYYY-MM-DD* or *DD.MM.YYYY*. Other values can have different physical quantities and other units of measurement.

Finally, one thing is not mentioned in (Klein, 2001). Names and tags can contain abbreviations and special types of it like acronyms (abbreviations made up of the initial components in a phrase). Misspelled words are also not syntactically equals and therefore can't be found by simple comparisons. Phrases and labels can also be used in an unexpected and jargon-specific way (Hughes and Ashpole, 2004).

Finally, two words can have the same semantic without being equals tags when the are translated in another natural language. This can also be interpreted as special synonyms. Therefore, all these issues of this paragraph can be summarized to the cluster of *terminological mismatches*.

1.4.2 The structure of ontology matching

Ontology matching is the (semi-) automatic process of connecting two ontologies which describe similar concepts and domains. In research area of *semantic web services* it is often called *ontology mediation* (deBruijn et al., 2006). Tools implementing this process are called *ontology matcher* and have typically two ontologies O_1 and O_2 as input and parameters or resources. The *parameter* can depend on input ontologies and change the behavior of the matcher, for example in case of multilingual ontologies. The abstract type *resources* can symbolize dictionaries, thesauri or other libraries that the matching algorithm want to use.

An optional input can be a mapping from previous matchers. The only output is a mapping between the two input ontologies. Figure 5 illustrates this abstraction.

1.4.3 Composition of matchers

There are many approaches for a matching algorithm.

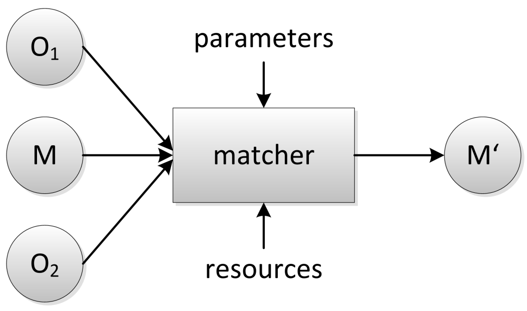


Figure 5: Matching process based on (Euzenat and Shvaiko, 2007, p.44). O_1 and O_2 are the two input ontologies. M is an alignment from previous matchers and M' is the resulting alignment.

To get better results and reduce the disadvantages of a single matcher, different strategies are applied. There are several ways to combine these matchers.

A sequential composition of matchers is depicted in figure 6.

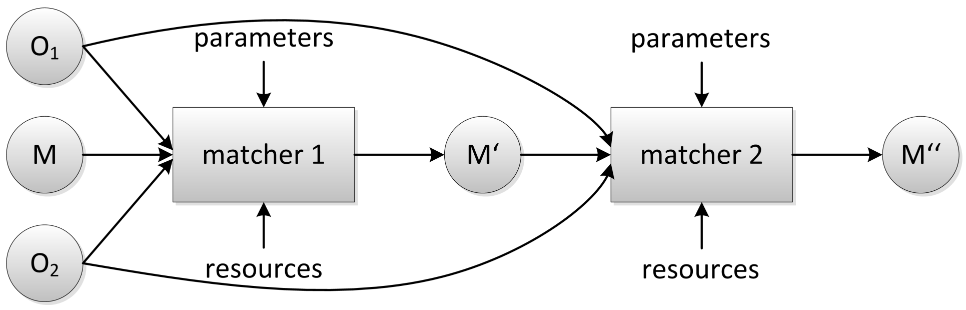


Figure 6: Sequential composition of matchers based on (Euzenat and Shvaiko, 2007, p.118). Matcher one and two have both the two ontologies has an input. The result of matcher one is the input of matcher two. The overall result is M''.

This composition is necessary when matcher two needs some initial mapping for his own approach. This is also useful for a specialized matcher, who only improves the mapping. This can be done by removing some mappings that are correct, but are not considered in a reference mapping. These *filters* can sometimes also work without the input ontologies.

Another way to join the matchers is a parallel composition. The matching systems run independently and as a post-processing step an aggregation of the results is necessary. The aggregation can highly vary by their merging result. This can be computed, for example, with an union or intersection over all results. Other algorithms are conceivable. The parallel composition can be subdivided into *heterogeneous* and *homogeneous parallel composition*. If the input is fragmented into different kinds, then this is a heterogeneous composition. Otherwise, if the inputs are forwarded to multiple matchers as a whole, then a homogeneous composition is made. Figure 7 shows the parallel composition.

1.4.4 Types of mappings

A mapping or alignment can be represented with different logical expressions and parameters as well as a complex mapping instead of a simple one.

Complex mappings can have a computation step between two concepts. Finding such mappings is a non-trivial task. For example, there are different speed units. Ontology A defines miles per hour and ontology B kilometers per hour. The approximate conversion between these two are x miles per hour * 1.6 = y kilometers per hour. This can be found by string comparison and the conversion in mind. Another complex mapping is a string concatenation between *first name* and *last name* to map against *full_name*. To find such mappings, the best way is to define *correspondence patterns* and, *matching conditions* and

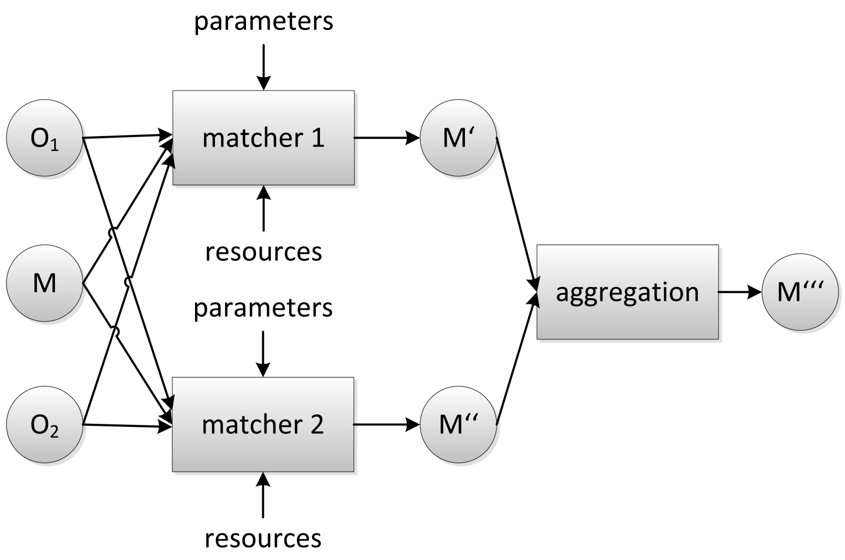


Figure 7: Parallel composition of matchers based on (Euzenat and Shvaiko, 2007, p.120). Matcher one and two run independently.The results M' and M'' are aggregated to the overall result M'''.

the resulting alignment (Ritze et al., 2010). The patterns have only placeholder in it, but substituting these dummy values with the correspondence values yields in the correct mapping.

(Ritze et al., 2010) specify four *correspondence patterns*. A simple one is called **Inverse Property (IP)**. With this pattern alignments like $writtenBy^{-1} \subseteq write\_paper$ can be detected. For this pattern there are three conditions. One of them is for example, that the passive and active voice changes between the verb phrases of the labels.

Another pattern is **Class by Attribute Type (CAT)** and the inversion **Class by Inverse Attribute Type (CAT^{-1})**. The CAT pattern found correspondences like $Accepted\_Paper \equiv \exists hasDecision.Acceptance$. The ontologies are shown in figure 8.

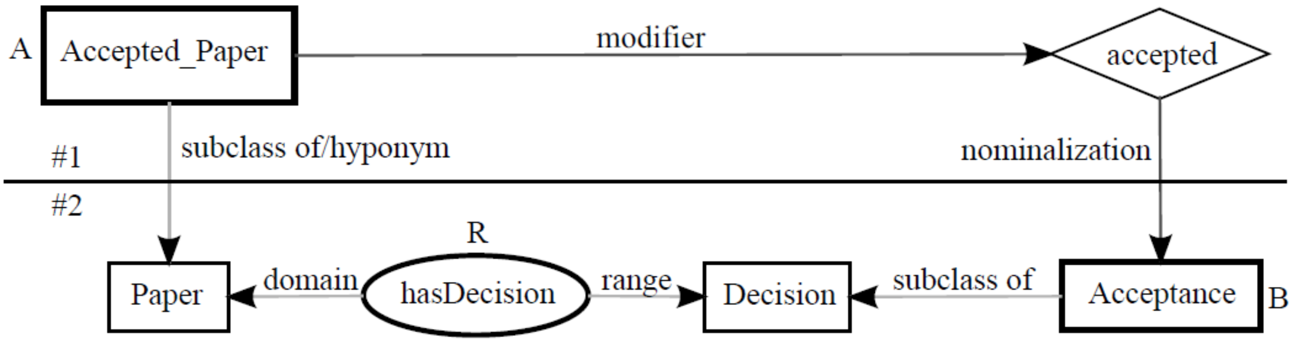


Figure 8: The pattern for Class by Attribute Type (CAT) by (Ritze et al., 2010).

The last pattern is **Class by Attribute Value (CAV)**. In conclusion, a class labeled with *Late-Registered_Participant* can be mapped with a property *earlyRegistration* and the range $r = false$. More detailed information is found in (Ritze et al., 2010).

All these complex mappings can be represented with *SWRL (Semantic Web Rule Language)* (Horrocks et al., 2004). It is not possible to express these mappings in RDF or OWL, because they have neither rules nor arithmetical expressions. An example of such a mapping in *SWRL* is shown in (Euzenat and Shvaiko, 2007, p.225).

Simple mappings instead have no computation step in between. They can be split up into *homogeneous* and *heterogeneous* mappings (Ghidini et al., 2007). Homogeneous mappings contain only the same types of resources like classes, object properties, data properties or instances. This type of mapping is often used, because it is sure that the union of the ontologies including the mapping is also in OWL Lite/DL. Of course the input ontologies have to be also described in OWL Lite/DL. Therefore a reasoning is still possible.

With heterogeneous mappings, a rapid valid union ontology is not possible. It is sufficient when an object property is mapped with a data property. Afterwards no reasoner can work with the union of the ontologies.

All simple mappings can be represented through different languages. This includes RDF, OWL and *contextualized OWL* (C-OWL) showed in (Euzenat and Shvaiko, 2007, p. 119 et sqq.). It can be contained directly in one ontology, making a mapping from the own ontology to another. This is called an asymmetric mapping. A symmetric mapping are two asymmetric ones resulting in a so called *bridge ontology*. This is an ontology containing only mappings from two imported ontologies.

From an abstract point of view an ontology mapping consists of 5 items. It can be represented as a 5-tuple (Euzenat and Shvaiko, 2007, p.46)

$$< id, e_1, e_2, r, c > \tag{8}$$

where

- id is an identifier for this mapping
- e_1 and e_2 are elements (entities) of ontology 1 respectively 2
- r is an relation that holds for the mapping
- c is a confidence value (also called measure)

E_1 and e_2 are only identifier for a specific resource. This is often an URI. Thus it is possible to express an homogeneous as well as an heterogeneous mapping. The confidence value c is an element of the interval $[0, 1]$. A value near 1 can be interpreted as a pretty sure mapping and value near 0 as a very weak one.

The relation r can be equivalence ($=$), subsumption ($\leq$ or $\leq$) and incompatibility ($\neq$) (Euzenat, 2004). It is possible to define more specific relations when matcher and reference alignment uses and interpret the same symbols equally.

To represent whole alignments in a programming language an API called *Alignment API* is often used. It is developed in java and contains at most interfaces for representing an alignment. Figure 9 shows the main classes of the API.

The alignment has an id,type and level. There are 3 different levels from 0 to 2. Level 0 has all items described in (8). Is is the basic variant which is language independent. Level 1 replaces e_1 and e_2 with sets of entities instead of a single one. Level 2 at least allows sets of expressions in a specific language. At this level complex mappings can be expressed through a rule language (Euzenat, 2004).

The type of alignment is usual the multiplicities like 1:1 (one-to-one), 1:m (one-to-many), n:1 (many-to-one) and n:m (many-to-many). In this case it is subdivided into 1 for total and injective properties, ? for injective, + for total and * for none. Figure 10 shows some configurations.

The *AlignmentProcess* is an interface for all matchers. The *align* method is called when the system should start the matching process. The *Evaluator* computes recall, precision and F-measure for a specific alignment. And each *Cell* represents a mapping with an id, confidence, relation and two entities.

Through an *AlignmentVisitor* an alignment can be rendered in different languages and formats (David et al., 2011). RDF is the one that is often used, for example in the Ontology Alignment Evaluation Initiative. A mapping in RDF for the *conference* track *cmt-confOf* (line 12 and 18) looks like

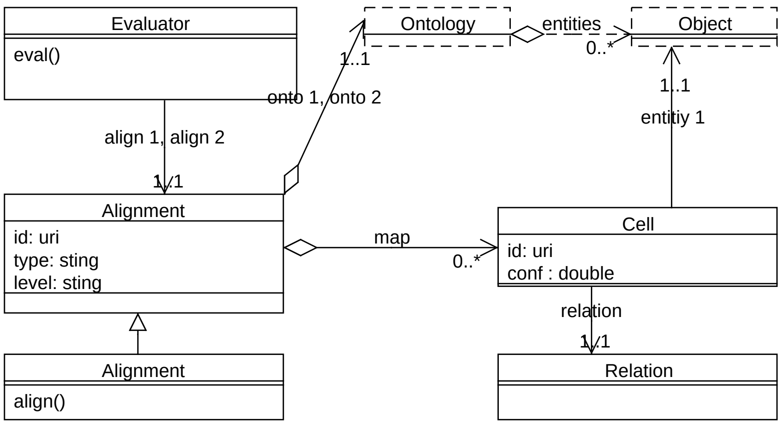


Figure 9: Main representational classes from the *Alignment* API based on (David et al., 2011).

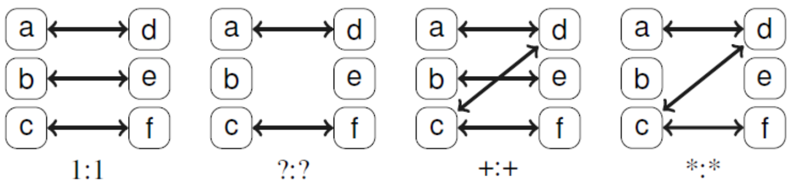


Figure 10: Possible configurations of alignments by (Euzenat and Shvaiko, 2007, p.49).

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns="http://knowledgeweb.semanticweb.org/heterogeneity/alignment"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<Alignment>
<xml>yes</xml>
<level>0</level>
<type>??</type>

<onto1>
<Ontology rdf:about="http://cmt">
  <location>http://nb.vse.cz/~svabo/oaei2010/cmt.owl</location>
</Ontology>
</onto1>

<onto2>
<Ontology rdf:about="http://confOf">
  <location>http://nb.vse.cz/~svabo/oaei2010/confOf.owl</location>
</Ontology>
</onto2>

<map>
  <Cell>
    <entity1 rdf:resource="http://cmt#ProgramCommitteeChair"/>
    <entity2 rdf:resource="http://confOf#Chair_PC"/>
    <measure rdf:datatype="xsd:float">1.0</measure>
    <relation>=</relation>
  </Cell>
</map>
```

```
30 <map>
31   <Cell>
32     <entity1 rdf:resource="http://cmt#writePaper"/>
33     <entity2 rdf:resource="http://confOf#writes"/>
34     <measure rdf:datatype="xsd:float">1.0</measure>
35     <relation>=</relation>
36   </Cell>
37 </map>
38
39 </Alignment>
40 </rdf:RDF>
```

Listing 1: Extract of conference reference alignment in XML

The level is 0 (line 7), which means that each map (line 22-29 and 30-37) contains only one cell (line 23-28 and 31-36). Each cell again has two entities (line 24-25 and 32-33), a relation (line 27 and 35) in this case an equal relation and a confidence (measure) in line 26 respectively 34 with the value of 1.0. The type (line 8) is ?:?. So in both directions the alignment is injective (Euzenat and Shvaiko, 2007, p.45)

1.4.5 Evaluation measures

To evaluate the quality of a matcher, the output mapping is compared to a *reference alignment*. This alignment is created by human experts to guarantee the correctness of the reference mapping. To test the system for their stability the ontologies are varied slightly. The reference alignment has still its validity, even if parts of the ontology are removed or replaced by random strings.

Evaluating a matcher can be regarded as a sub-problem of *binary classification*. The most used example for an *classifier* is a test for a disease. In case of ontology matching the test is not for a disease but for alignments that might be included in the reference mapping or not. The evaluation of this test can yield in 4 different results. The returned mapping from a matcher is now alignment A and the reference mapping is abbreviated with R.

The first result is called **true positive** (tp) and means that an alignment is contained in A as well as in R. So the matcher has computed the right mapping. Another right behavior of a matcher is not to find the mappings missing in alignment R. This is called **true negative** (tn).

The next two results are errors that a developer of a matching system wants to reduce. In statistics the following error is called *type 1 error* or *error of the first kind*. It means that the matcher returns too many alignments that are not contained in the reference alignment. To improve the system the threshold should be higher so that the matcher returns mappings where the confidence is higher.

If the threshold is too high another type of error comes into play. The *type 2 error* or *error of the second kind* is a **false negative** (fn) result. The matcher doesn't return a mapping although the mapping is contained in the reference alignment. To reduce this type of error the system should return more mappings. The threshold of a system can be lower to reduce this error. It is also called **false positive** (fp).

True and *false* reflects the expectation in this case the reference alignment. Whereas the terms *positive* and *negative* correspond to the prediction respectively observation of an matcher. To conclude, the first term refers to the expectation (reference alignment) and the second one to the observation (returned alignment from a matcher). Table 1 shows the results in an overview. Green ones are correct results and red ones are errors that the system made. One is anxious to reduce the errors but keep the correct results.

It is also possible to illustrate this with mathematical sets. The intersection between the reference alignment R and alignment A are the *true positives*. The two errors are mathematically spoken $R \setminus A$ (*false negatives*) and $A \setminus R$ (*false positives*). Figure 11 displays these sets.

		Reference alignment R (expectation)	
		contained	missing
Alignment A (observation)	contained	true positive (tp)	false positive (fp) - Type 1 error
	missing	false negative (fn) - Type 2 error	true negative (tn)

Table 1: Classification of results a matcher can make. The green cells represents desirable alignments whereas the red ones should be avoided to get good results.

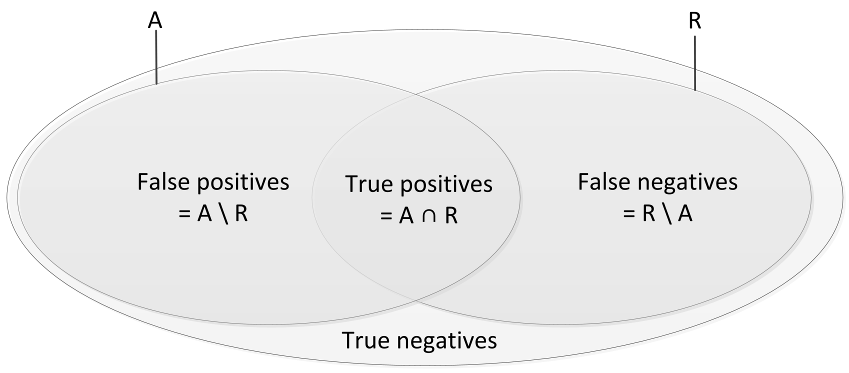


Figure 11: Illustration of the errors and correct results a matcher can make. A is an alignment returning from a matching system and R is the reference alignment. This figure is based on (Euzenat and Shvaiko, 2007, p.206)

Precision

In *information retrieval* three measurements have been developed (Manning et al., 2008, p.155). The first one is *precision*. With the abbreviations tp for *true positive* and so on it is defined as

$$Precision = \frac{tp}{tp + fp}. \tag{9}$$

It can also be described with the alignment A and reference alignment R:

$$Precision(A,R) = \frac{|R \cap A|}{|A|}. \tag{10}$$

The precision (also known as ***p****ositive* ***p****redictive* ***v****alue*) describes the quality of the alignments that a matcher found. All alignments that are found and correct are divided by all returned alignments. A value near 0 means that only a few alignments are correct. If the value equals 1, all alignments are contained in the reference alignment. It is very easy to result in a high precision value. The system should only return alignments where it is fully confident, even if it is only 1 alignment. If 1 correct alignment is assumed, the precision of the system is also $prec = 1$.

Recall

The counterpart is recall (a synonym is ***t****rue* ***p****ositive* ***r****ate* and *sensitivity*). It can be calculated with the formula

$$Recall = \frac{tp}{tp + f\mathbf{n}}. \tag{11}$$

The difference to precision is that now the true positives are divided by $tp + fn$ and not by $tp + fp$. This value shows how many alignments of the reference mapping were found. It can also be described depending on the sets A and R:

$$Recall(A,R) = \frac{|R \cap A|}{|R|}. \tag{12}$$

To improve this value, a matching system can return all possible mappings. All reference alignments will be also contained for sure. Therefore the recall is 1. The smaller this value, the less reference alignments were found.

F-measure

Recall and precision can be improved separately very well. Thus it is essential to have a measure that considered recall as well as precision. This measure is called *F-measure* and is defined as

$$F_\beta = \frac{(1+\beta^2) * precision * recall}{(\beta^2 * precision) + recall} \tag{13}$$

This formula is valid for all positive real β. The value of β is often $\beta = 1$ which is the harmonic mean of precision and recall shown in equation 14.

$$F_1 = \frac{(1+1^2) * precision * recall}{(1^2 * precision) + recall} = \frac{2 * precision * recall}{(precision + recall} \tag{14}$$

Other widely used values for β are the F_2 measure that weights recall higher and $F_{0.5}$ measure that is more precision-oriented ($F_0 = precision$).

The origin definition of the F-measure was developed by van Rijsbergen (Rijsbergen, 1979, p.134) with his effectiveness function

$$E = 1 - \frac{1}{\alpha\frac{1}{precision} + (1-\alpha)\frac{1}{recall}}. \tag{15}$$

By substituting α with $\alpha = \frac{1}{\beta^2+1}$ the E function is

$$E = 1 - \frac{(\beta^2+1) * precision * recall}{\beta^2 * precision + recall} = 1 - F_\beta. \tag{16}$$

The conversion between E function and F-measure is found in (Sasaki, 2007).

For an analysis of multiple F-measure values with the corresponding recall and precision a triangle is a nice representation. It is used in the results of the *Ontology Alignment Evaluation Initiative* of 2011.5. Figure 12 shows this triangle. From the bottom right diagonal to the imaginary top left corner are increasing recall values. And from the bottom left to the top right are increasing precision values. The solid lines display specific F_1 values.

All these measures are compliance measures.For a more detailed evaluation of a system, performance measures are added. They depend on the implementation of a system as well as the hardware that runs the matching task. The most used measure in this category is the amount of time that a matcher use for the task (speed). Other not that important measures are memory usage and scalability (Euzenat and Shvaiko, 2007, p.212).

1.4.6 Types of matchers

There are multiple approaches to match ontologies. Each of it has its advantages as well as disadvantages. Therefore many matching systems combine different approaches to get better results. Figure 13 shows many approaches hierarchically subdivided.

Matchers can be classified by the level on which they operate. A matcher on the element level uses only the entities of a ontology like labels, comments and so on. This ignores every connection, respectively relation to other concepts. On an structural level the approaches analyze how the entities are related to each other. This includes inheritances of classes and also of properties. It is often the case that these

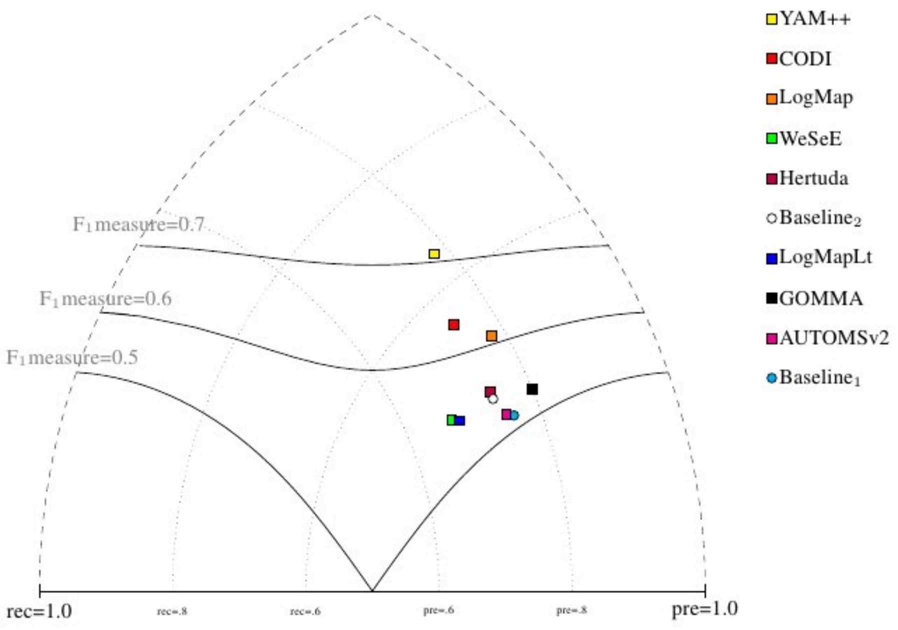


Figure 12: Triangle for displaying multiple F-measure values with corresponding recall and precision. See `http://oaei.ontologymatching.org/2011.5/results/conference/index.html`

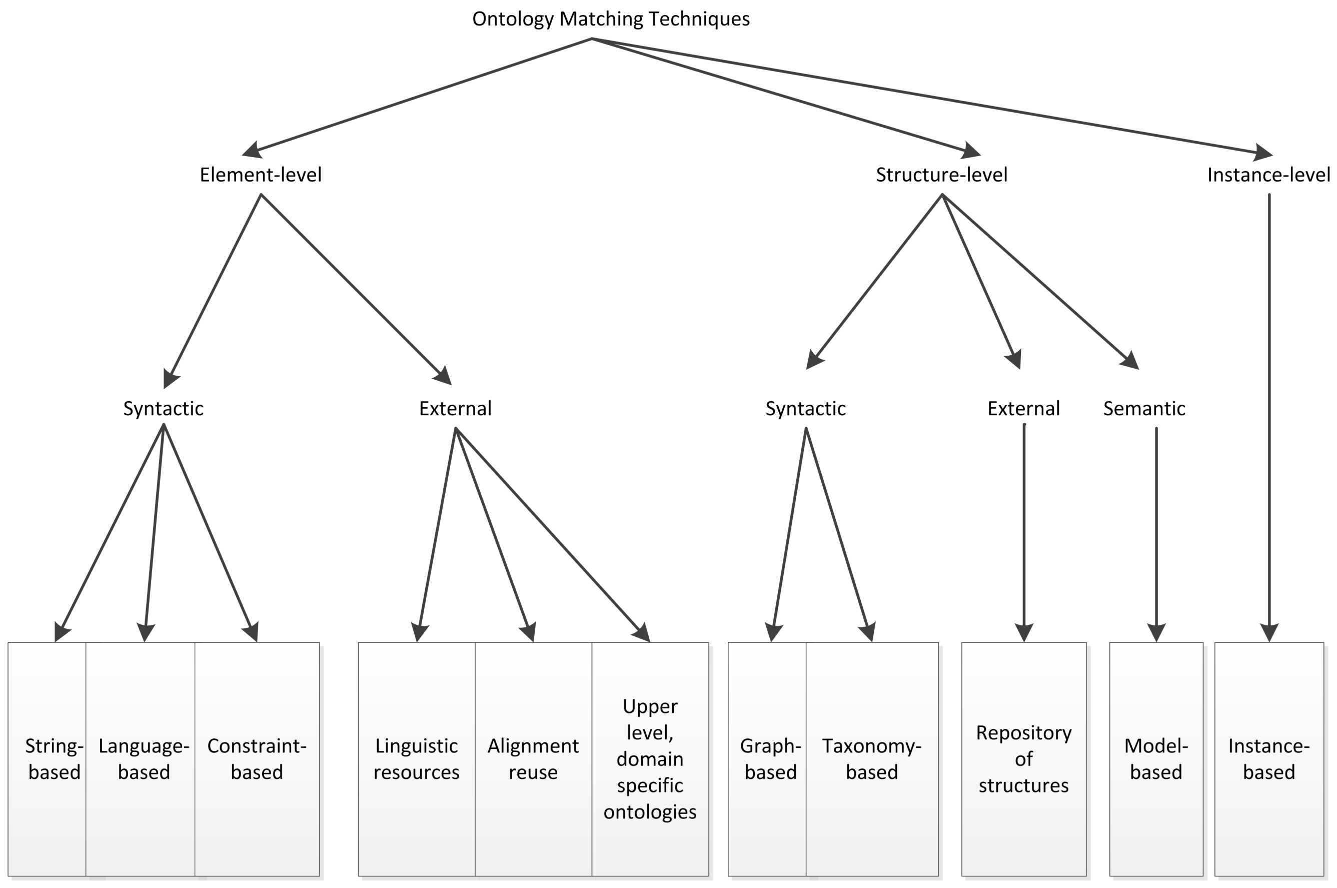


Figure 13: Classification of elementary matching approaches (based on (Shvaiko and Euzenat, 2005)).

matchers require an initial mapping as input. This can be produced by previous systems based on the element level.

The second division contains syntactic, semantic and external. Syntactic approaches uses only the ontology. Whereas the semantic approaches use formal or model-theoretic semantics. This can be applied with an reasoner. An external approach uses (external) resources like thesauri. It is also possible to query tools that contains common knowledge. The following paragraphs describe every approach in more detail.

String-based techniques

A string based approach compares given strings like comments, label etc. for equality (name=name), prefixes (mobile=mobilePhone) or postfixes (name=hasName). The idea behind all these comparisons is that two strings describe the same concept if they are very similar. This is true for a large set of words, but homonyms are mapped although the semantics differ.

Another comparison of strings can be achieved with string distance measures. One of them is the *edit-distance*. It divides the amount of steps need to transform string one into string two by the major length of the strings. Transform steps can be inserting, deleting or replacing a character. The Damerau-Levenshtein distance add one extra basic operation so that two adjacent characters can also be transposed (Damerau, 1964; Levenshtein, 1966).

A common string distance is the n-gram analysis. It computes how many substrings of length n are equals. This is divided by the amount of substrings that is greater.

Language-based techniques

Language-based techniques are based on natural language processing. This contains the removal of stop words that require knowledge about the used language. These techniques also contain algorithms for stemming like the Porter stemming algorithm (Porter, 1980). All language-based systems can be used as a preprocessing step for string-based matchers. They can also contain a tokenization of terms. For example an often used style for writing words without spaces is called camel case. A term *ontology-Matching* will then be split up into $ontology$ and $matching$. The tokens are then processed with other techniques.

Constraint-based techniques

Constraint-based techniques use the constraints defined in the ontology. This can be the cardinality of properties or attributes. Some restrictions are also applied to (data-) types and can be used to match entities. This approach can be also used for checking of already mapped elements. If a datatype property has the same range, this can increase the confidence value.

Linguistic resources

Linguistic resources can find mappings between terms that are syntactically different but have similar semantics. To match these synonyms, lexicons, thesauri or dictionaries are required. These often external resources can also be helpful to match ontologies in different natural languages. The lexicons can be domain specific to find special mappings.

A common thesauri for the English language is WordNet (Miller, 1995). This contain linguistic relations between words like synonyms and homonyms.

Alignment reuse

Alignment reuse tries to profit by already mapped elements from ontologies with a similar domain. If the task is to match ontology A with ontology B and there exists a mapping between A and a new ontology C as well as a mapping for A and C, this can be used for further approaches. It can be seen as a mathematical transitive closure depicted in equation (17), where M is a relation that has the meaning of a mapping.

$$\forall a \in A, b \in B, c \in C\, (aMc \wedge bMc) \Rightarrow aMb \tag{17}$$

It can also be used as a hint to map elements. If a mapping between a and c already exists, further approaches can also compare c to b instead of a to b.

Upper level ontologies

The idea of upper level ontologies is to describe very abstract concepts that can be used and mapped with most of the existing ontologies. This provides more semantics in case of missing structure or labels, comments and so on.

Upper level ontologies like Cyc, SUMO (Suggested Upper Merged Ontology) and DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering)describe general knowledge. The mapped upper level concepts can thus be a hint, that entities are related to each other.

Graph-based techniques

These techniques consider the input ontologies as a graph. The similarity of concepts are not based on their labels or comments, but instead of their position in the graph. The idea is that the neighborhoods of two matched concepts can also be very similar. It is also usable for finding a new property mapping, when some classes are already matched. An approach for this type is found in (Dang et al., 2012). It is called *GraphbasedUseClassMatcher* and match properties when the domain and range are already matched. The basic schema for it is showed in figure 14.

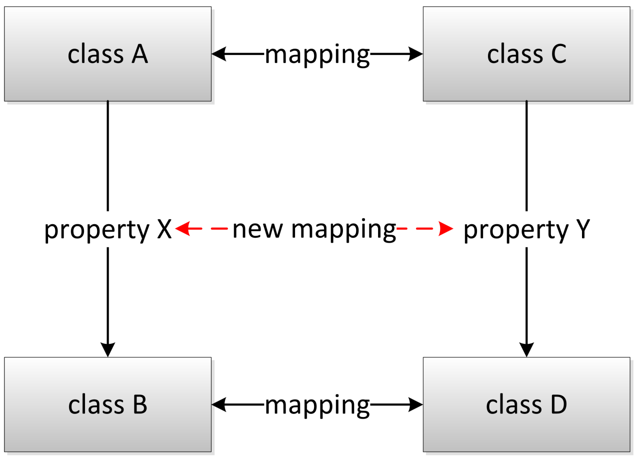


Figure 14: Graph-based matcher where the new mapping between property X and Y is in red. The classes have to be mapped in a previous step (see (Dang et al., 2012)).

Taxonomy-based techniques

Taxonomy-based techniques use the structure of inheritance to find new mappings. Sub- and super-classes are an often used instrument in ontologies and, therefore, these algorithms can also produce great results. The input ontologies are reduced to a graph, where the edges represents the inheritances.

Repository of structures

Some structures appear very often in ontologies - like the concept of a person with name, phone and so on. These structural equivalences are stored in so called repositories of structures. They include a subset of the ontologies to find these patterns in unknown schemata.

This does not result in new mappings but the effort may be lower, because only the matched subgraphs required a detail look to really find a new alignment. To increase the speed of the pattern matching it is possible to compute a similarity value between subgraphs, depending on number of nodes, maximal path length and other features (see (Euzenat and Shvaiko, 2007, p.69)).

Model-based techniques

Model-based techniques uses the semantics that is given implicit by the definition. The idea is that two equal concepts have similar interpretations.

It is also possible to use this technique to improve the confidence of alignments or removing them if they are indefensible. To find such mappings, both input ontologies along with the alignments are combined and a reasoner can find an inconsistency when a disjoint class is no longer disjoint after the combination (Paulheim, 2011d).

Instance-based techniques

This technique does not depend on the ontology itself. It finds mappings between classes by instances. With many instances available that have two different types from maybe different ontologies, these two classes can be mapped. It is showed in figure 15 that many instances has the type *city* as well as the type *metropolis*. Therefore a mapping between them can make sense.

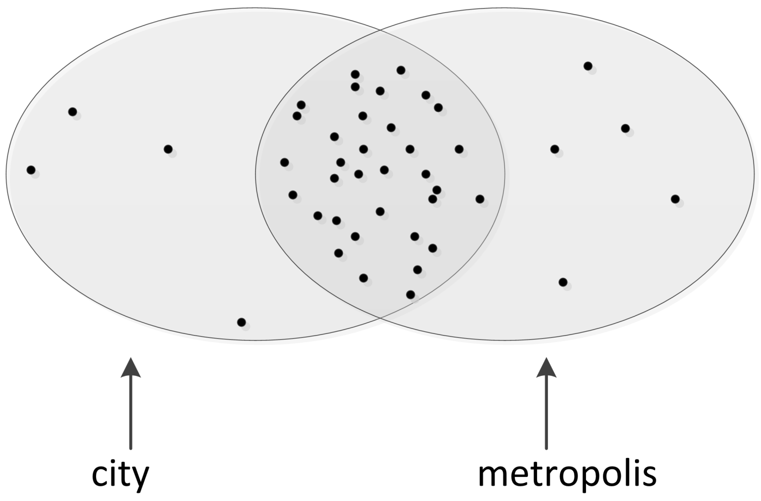


Figure 15: Many instances have both the type *city* as well as *metropolis*. Thus a mapping can be created out of the instances and without any ontology (based on (Paulheim, 2011d)).

1.5 Wikipedia as an external resource

In this work a new matching approach will be presented and evaluated. The basic idea is to use Wikipedia as huge background knowledge that is still growing. It is an element based approach and the basic idea is to map concepts when the terms describing this concept return similar articles from the search engine in Wikipedia. The results rely therefore on the detail level of Wikipedia for a specific domain.

There are different approaches to match ontologies with external resources. It is, for example, also possible to use a search engine to use the hole web for matching. This approach is restricted to Wikipedia, because the intention is that Wikipedia contains more assured knowledge. This can yield in better precision, but on the other hand the recall can be low. That is often the case when the domain of the input ontologies is covered in more detail. It will be better when the terms (labels and comments) have own articles, where everything is described in more detail.

When manually reading through an article, one can find language links on the left side. This will link to the corresponding pages in other languages, where the topic is also discussed. The proposed approach will retrieve all article titles in all available languages. To compare concepts, it is important how many articles overlap in a language.

The continuation of this thought will end up that it is possible to find overlappings in other languages than the language of the terms. An example is the following: if a term a in English and a term b in German should be matched and both have mappings to the same Spanish article, this would also be helpful.

An advantage of this approach is that the growing of Wikipedia improve the matching results. When more detailed information is available, more suitable article tiles can be returned and analyzed. The search engine can improve the matching system by giving hints and suggestions to correct spelling mistakes. Some synonyms will maybe also have similar articles that contain these terms. The disadvantages on the other side are the runtime of the matcher as well as a small recall in case of less articles related to a term.

2 Related Work

This section discusses only state of the art matchers and approaches that uses external resources. It is split into the three subsections. The first one takes care of matchers exploiting search and translation engines whereas the second one lists two matchers using WordNet. The related work section is concluded with approaches which make use of Wikipedia or Wiktionary.

2.1 Use of search and translation engines

The basic idea is to use search engines to find out, based on the retrieving results, how similar two terms are. This approach takes longer than comparing strings with some metrics calculated on the CPU but can also find non trivial mappings. For the multilingual case many matching systems use translation engines to simplify the problem to a monolingual case. Therefore all terms are translated to one pivot language which is often English.

2.1.1 WeSeE

WeSee is matching tool developed at the Technische Universität Darmstadt (Paulheim, 2012). It is based on the way of how humans would solve a matching task. They search for both terms in a search engine and examine the results to decide if the terms have something in common.

The algorithm uses the labels, comments and URI fragments as terms for searching to compare whole concepts. They use Microsoft Bing Search API as the search engine with some preprocessing of the terms, because a larger amount of queries are possible. For one term the result of the search engine is put into a *describing document* based on titles and excerpts from websites. Thus it is not necessary to parse all websites. The similarity score for two concepts is the maximum of comparing all labels, fragments and comments. The similarity between two documents is computed with a TF-IDF score.

To also match multilingual ontologies all terms are first translated into English as a pivot language. This is done by the Bing Search API's translation capabilities. The overall matching process of WeSeE is shown in figure 16.

2.1.2 MapSSS

MapSSS is a preliminary work using very simple similarity metrics. The three 'S' in the name represent the three metrics (syntactic, structural and semantic). The tool participated in OAEI 2011 (see section 4.1) and the approach is described in the results papers (Cheatham, 2011).

The structural metric consists of a comparison of terms by the Levenstein distance. The labels are lowered and preprocessed so that words with underscore or camel case are split into single words.

The structural metric interprets the ontology as a graph and acts on the direct neighborhood of already matched entities.

The semantic metric is not yet implemented, but it is announced that they would like to use Google Research API to query Google to find synonyms and translations.

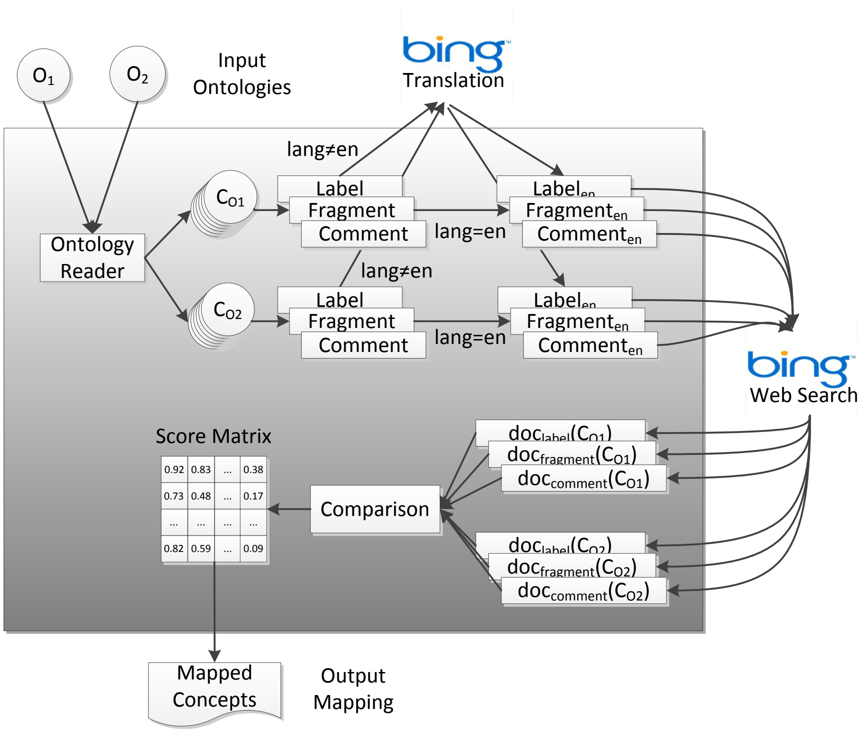


Figure 16: WeSeE matching process: All labels, comments and fragments are extracted and (if necessary) translated into English. The describing documents are computed based on the result of the search engine. After a comparison the score matrix provides the mapped concepts (see (Paulheim, 2012)).

2.2 Use of WordNet

WordNet (Miller, 1995) is the most used external resource in the current OAEI competition [7] (see section 4.1). The following matchers uses this resource to get some similarity values:

- AUTONOMSv2 (Kotis et al., 2012)
- MaasMatch (Schadd and Roos, 2012)
- Optima+ (Thayasivam et al., 2012)
- YAM++ (Ngo et al., 2011)

Since it can be store locally, it is much faster than querying online resources.

WordNet is a lexical database designed to be used by programs instead of humans. Thus it contains nouns, verbs, adjectives, and adverbs which are grouped to sets of synonyms and semantic relations between each word.

2.2.1 YAM++

YAM++ is a flexible matching tool based on machine learning and similarity flooding approaches (Ngo et al., 2011). It is an extension of the first version called YAM - not Yet Another Matcher to combine different similarity metrics by using machine learning techniques. The matching process is fully automated and uses some external resources like Bing and Wordnet. The components of the new version is illustrated in figure 17.

[7] http://oaei.ontologymatching.org/2012/

The input ontologies are first passed to the Parsing & Processing module which is based on Pellet and OWLAPI [8]. As a next step, all element based approaches are applied to find mappings with a high confidence. The terminological matcher uses different similarity metrics which are combined by a machine learning model. For the learning phase they use the Benchmark dataset from OAEI 2009 as a *gold standard*. Many String-based metrics like Levenstein and JaroWinkler are applied. The interesting parts are the Language-based metrics that are implemented. The three metrics correspond to (Lin and Sandkuhl, 2008). They use Wordnet [9] as a local thesaurus. If a non empty set of morphological forms are returned, the maximum of comparing the sets element-wise is defined as the score. A String metric is applied in case of empty sets.

The Extensional matcher tries to find similar instances and from that point discovering new mappings. On the structure level, the well-known Similarity Flooding algorithm is implemented and applied. The combined results of Element- and Structure-based approaches are filtered in the module Constraint & Selection. They use some semantic patterns to remove inconsistent alignments. To get an 1:1 alignment two assignment methods are called (Greedy and Hungarian algorithm). To also match multi-lingual ontologies they use Microsoft Bing Translator tool to translate all terms into English if necessary.

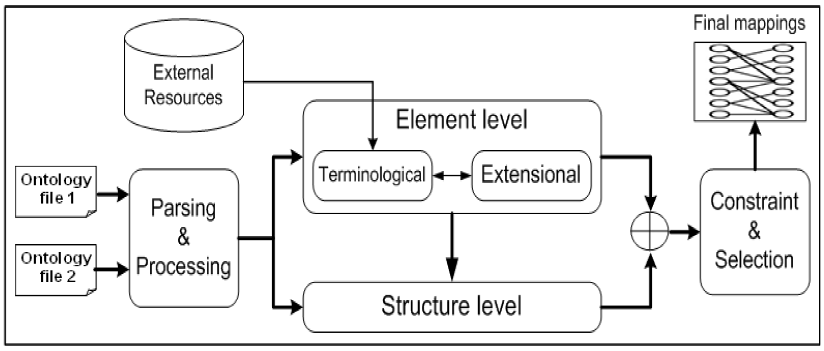


Figure 17: YAM++ reads both ontologies in a parsing and processing step. Afterwards algorithms of the element and structure level are applied. At the end a selection of the final mappings is done (see (Ngo et al., 2011))

2.2.2 MaasMatch

MaasMatch focused on resolving terminological heterogeneities between the input ontologies. They support monolingual ontologies with up to 2000 concepts (Schadd and Roos, 2012).

Altogether, 4 similarity measures are implemented and one function to combine the results to the final alignment. The **syntactic similarity** is based on 3-grams (Shannon, 1948) of names and labels of concepts compared by the Jaccard measure (Jaccard, 1901).

A Name-Path similarity is applied as a **structural similarity**. The first step is to search all terms which are descriptions of ancestors of a concept. To all these strings, a hybrid distance is applied. In detail, the maximum similarity value for the smaller set of tokens compared to the tokens of the second term are computed and divided by the amount of the smaller set. The similarity between two tokens is then computed with a specialized Levenshtein similarity (Levenshtein, 1966).

Another measure is the **virtual document similarity**. It compares virtual documents (Zhang et al., 2011) that are generated by the information of the concept and their neighbors. One document represents one concept in the ontology. Descriptions, such as labels and comments, are weighted document vectors. Thus this approach allows to give different weights to labels or names describing the concept. Moreover, two related virtual documents can also be influenced by each other.

[8] `http://owlapi.sourceforge.net/`
[9] `http://wordnet.princeton.edu`

The **lexical similarity** is based on WordNet. For every concept and each description a virtual document is created. The WordNet similarity is then computed for all combinations that might denote the meaning of the concept. Figure 18 shows the approach.

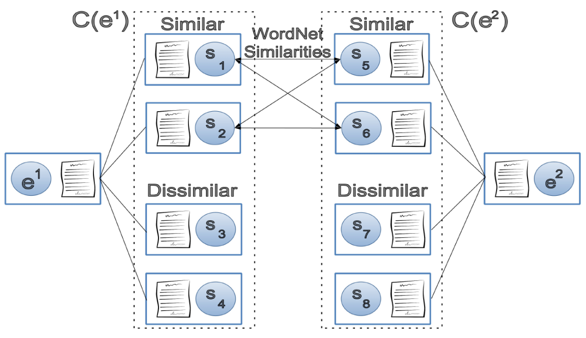


Figure 18: Lexical similarity approach for MaasMatch (see (Schadd and Roos, 2012))

The last step **aggregation and extraction** computes the average confidence of all mappings. The final alignment is then extracted by the Naive descending extraction algorithm from (Meilicke and Stuckenschmidt, 2007). It is a little bit modified to allow only mappings with a higher confidence than a given threshold.

2.3 Use of Wikipedia/Wiktionary

Wikipedia and Wiktionary are not often used in ontology matching although they are one of the largest cross domain resources available. They grow every day and never become outdated. One big disadvantage is the runtime of such approaches if no preprocessing or caching is made.

2.3.1 COMS

In (Lin and Krizhanovsky, 2011) an approach for matching multilingual ontologies based on *Wiktionary* is introduced. They developed a parser for Wiktionary and put the extracted data to a relational database. With an D2R server it is possible to let the database look like a RDF store where SPARQL queries can be processed.

The system architecture is depicted in figure 19. They use the interface of SPARQL to interact between the *COMS Ontology Matching System* and the database with the information from Wiktionary. The D2RQ Engine which contains the mapping between the MySQL Database and the appropriate ontology is the Jena framework described in section 1.3.3. They use the dump of Wikitionary which is a simple database and a own parser to create the MySQL database. The parser extracts the meaning, translations of the first meaning and semantic relations (e.g. synonyms) to be able to query translations for a term in English to other languages.

COMS (Context-base Ontology Matching System) translates all to a target language and applies then a string, structure and lexical matching algorithms. All of them run parallel and will be combined in an postprocessing step. They use the Jaro-Winkler distance (Winkler, 1999) and the SmithWaterman algorithm (Smith and Waterman, 1981) for string matching.

The structure matching consists of three approaches. The first one applies if two elements of a triple are equals. Then the third one is mapped. If a mapping between two classes exists, the superclasses of them are said to be equals. The last method is to expand the ontology as a tree (Lin and Sandkuhl, 2007) and calculate the similarity based on weights in this tree. There are levels of concepts which gets different weights. First level concepts are for example the subclasses of a class c and each property where c is the domain or range. Further levels are described in (Lin and Krizhanovsky, 2011).

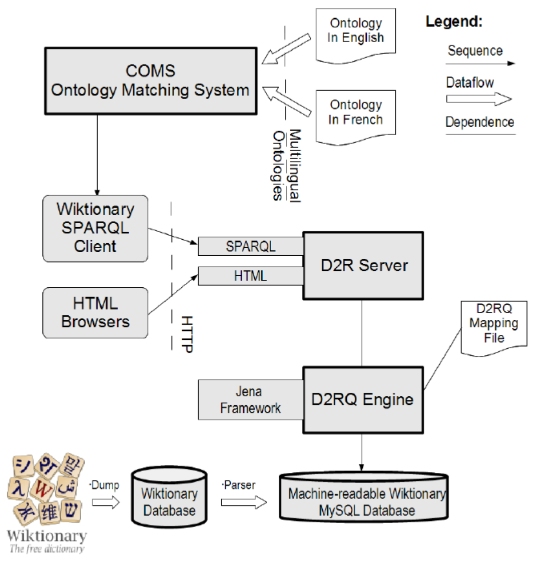


Figure 19: Architecture of the hole approach with the COMS System and Wikitionary (see (Lin and Krizhanovsky, 2011))

The lexical matching strategy is based on WordNet. They use the Jiang-Conrath measure (Jiang and Conrath, 1997) to find similar classes (the threshold is set to 1.0). After all matching strategies, the results are combined and returned.

2.3.2 BLOOMS / AgreementMaker

BLOOMS (Jain et al., 2010) make use of Wikipedia to find equal classes as well as sub- and superclass relations. This is done especially with the category hierarchy of Wikipedia articles. BLOOMS is also used in AgreementMaker to increase F-measure for biomedical ontologies (Pesquita et al., 2010).

For each class a set of trees is constructed, containing supercategories that correspond to the class name. For a class C the set of trees is called BLOOMS forest for C. The comparison between two BLOOMS forests will result in a decision if the classes should be matched and in a positive case which relation should be used in between. The performed steps can be summarized with preprocessing, constructing and comparing BLOOMS forests and postprocessing.

In the preprocessing step all properties, their restrictions and individuals are removed. For all class names, underscores and hyphens are replaced by spaces and camel case words are split into single strings. Afterwards stopwords are removed.

The interesting step is the construction of BLOOMS forest. As a first step the preprocessed term is queried on Wikipedia and the article URLs are returned. If it is a disambiguation page, it is replaced by all articles contained in the disambiguation page. Every article is then called a sense of the term. Figure 20 shows the resulting tree for two senses, respectively articles, namely “Jazz Festival” and “Event”. For every sense of a term, a tree is constructed with the term as a root and all categories and subcategories as additional nodes up to a depth level of 4 categories.

To compare two terms, all trees for these two terms are compared to each other. The similarity (also called overlap) of two trees is calculated by first removing all subtrees where the root of the subtree is contained in both trees because they are redundant. In the picture these are all nodes which have a deeper level than the gray nodes. The similarity is then computed as $\frac{n}{k-1}$, where n are all nodes which occur in both trees and k the number of nodes in the first tree. Thus, the similarity between “Event” and “Jazz Festival” is $\frac{3}{4}$, whereas “Jazz Festival” compared with “Event” has $\frac{3}{5}$ as similarity value. If all trees of one term compared to each tree of the other term has the same similarity value then the two terms are said to be equals. If otherwise the similarity of the sets compared in both orders are greater than an threshold then a subclass relation is created.

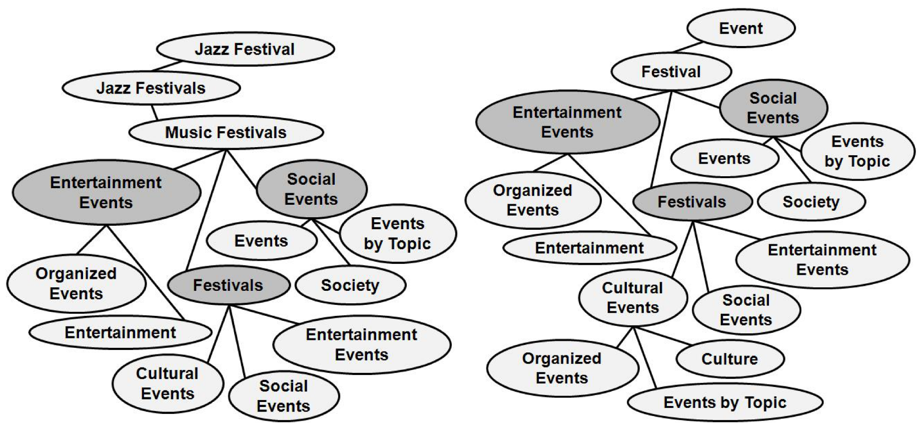


Figure 20: Two BLOOMS trees for "Jazz Festival" and "Event" (see (Jain et al., 2010))

In the postprocessing step the reasoner of Jena is called to add all transitive mappings. Finally, all alignments are added to the Alignment API.

3 Approach of WikiMatch

The basic idea of this approach called *WikiMatch* is to use Wikipedia as a huge external resource for matching ontologies. The first step is to retrieve a set of articles that describe the concept. The terms for the search query are extracted from labels, URI fragments and comments. When the answer contains a suggestion, a new query with it will be send away.

As a second step all language links per article are followed to find articles in other languages. Two concepts are then compared by how many articles overlap in these sets of articles. The idea is that the intersection of the resulting sets of articles has many elements when both terms have something in common.

3.1 Wikipedia and Programming interface

Wikipedia is a free online encyclopedia based on 23.6 million articles in 271 languages [10]. The articles are written by volunteers around the world collaboratively. The name consists of **wiki** which is an Hawaiian word for quick and *encyclo**pedia***. Owner of Wikipedia is the Wikimedia Foundation, a non-profit organization. It was launched in 2001 by online entrepreneur Jimmy Wales and academic Larry Sanger (Miliard, 2008).

Wikipedia is technically based on *MediaWiki* which was specially developed for this purpose. It is today used by many websites because of its GPL licence like *Wiktionary*, *Wikinews* and private ones. The framework is written in PHP and offers an API [11].

To access the API a simple HTTP GET request for a specific URL is required. In this work java is used to make the requests as well as reading the ontologies (see section 1.3.3 about jena). The answer is decoded as a XML document as default.

3.1.1 Creating the API endpoint

The first thing is to create the URL to the endpoint. It has a specialisied format [12] that contains the language of the Wikipedia, respectively the API. This approach uses two ways to find out which language to use. If a literal like label or comment contains a language tag, this will be used. For URI fragments which never has information about languages, the most used language tag is extracted from the ontology.

[10] `http://stats.wikimedia.org`
[11] `http://www.mediawiki.org/wiki/API:Main_page`
[12] `http://(language).wikipedia.org/w/api.php`

```
String getDefaultLanguage(Ontology o){
  Map<String, Integer> map;

  for each label in o.listResourcesWithLabels
    if(language(label) in map)
      map.put(language(label), map.get(language(label) + 1);
    else
       map.put(language(label), 1);

  String maxLanguage;
  Integer maxCount;
  for each entry in map
    if(value(entry) > maxCount){
      maxLanguage = key(entry);
      maxCount = value(entry);
    }

  return maxLanguage;
}
```

Figure 21: Algorithm for extracting the most used language tag

This applies also for literals that contain no language information at all. The algorithm to find the maximum used language tag is on the one side depicted as pseudo code in figure 21 and in java code using jena in the appendix section in listing 2.

The algorithm is quite simple. At first, all resources with labels are extracted. In a dictionary, respectively a map in java, the association between the language and their occurrence is stored. If the language (key) already exists in the dictionary, the value is increased by one. Otherwise the initial value of one is added. At the end, the maximal value is searched and the related key is returned. On the java side, jena is used to create an iterator for all resources with labels. With every loop of the labels that are related to this resource, it is checked if it occurs in the map. Everything else is just like the pseudo code.

With the created endpoint it is possible to send a request to the API. For example, the URL `http://en.wikipedia.org/w/api.php?action=query&list=search&srsearch=wikipedia` is a query for all sites containing the word “wikipedia”. After the endpoint and a closing question mark tuple of keys and values are followed. In the example the one key is *action* and the related value is *query*. Between each key and value there is an equal sign. In case of whitespace in the value, which is represented in java as a string, it is necessary to encode it at first to UTF-8.

3.1.2 Search functionality and parameters

The API offers two different search engines which differ in their purpose. The first one is *Opensearch* [13]. It is used to retrieve possible endings of a term that is a Wikipedia article at once. The autocomplete function of the web front end of Wikipedia is an use case of the *Opensearch* engine. In case of an ontology matching task the full term already exists and thus do not need to be completed. Additionally, the engine does not perform good results in case of multiple words [14].

The standard search engine performs a full search in all articles [15]. The result is a set of articles that contain the search term. There are several parameters that can be set:

[13] `http://www.mediawiki.org/wiki/API:Opensearch`

[14] An example from OAEI is *Program committee* with the URL `http://en.wikipedia.org/w/api.php?action=opensearch&search=Program%20committee&format=jsonfm`

[15] `http://www.mediawiki.org/wiki/API:Search`

- The **info** key can have values of *totalhits* and *suggestion*. If a word is misspelled and the suggestion parameter is set, then the engine will return the correct spelled word.
- **limit** is the amount of pages that can be returned at once. For the English Wikipedia and a normal user this is restricted by 50. By log in with an user that has the *apihighlimits* right this can be increased to 5000 [16].
- The **namespace** for the search can be one of Media, Special, Talk, User etc. [17]
- An **offset** parameter is chosen when a query is continued by this offset.
- The **prop** parameter indicate which properties are returned for each article. Possible values are size (size of the page), wordcount and snippet (the snippets where the search term occurred).
- If the **redirects** flag is set, then all redirect pages are included.
- The **search** parameter is the only one that is required, because it has the search term as value.
- Parameter **what** describes where to search for the given term. This can be in page titles (title), in page text (text) or a search for the exact title (nearmatch).

All parameters have to be prefixed by *sr* to indicate the search query. This search engine works much better than the *Opensearch* feature of the API and thus it will be used here.

To retrieve all pages that contain a term t, a special URL [18] is used. The *prop* parameter is empty to increase the speed and decrease the length of the answer. All redirect pages are included and the maximum amount of titles (in this case 50) is used. To indicate that this request is a search, the parameter *action* is set to *query* and *list* to *search*. The format of the answer should be in *json* (JavaScript Object Notation) to parse it easily in the java based matcher. The term t is searched in the hole articles because the *what* parameter is set to *text*. To retrieve a suggestion if the term t is misspelled, the value of *info* is *suggestion*. The result for $t = paper$ is shown in the appendix in listing 3. The returned answer contains only article titles instead of id's. There are API calls for translating from an article title to an id, but this will be another time consuming operation. Instead using article titles works and is also an unique identifier.

The second step required for this approach is to find all translations for the articles. The API offers a lot of information about articles. For example all categories, images and links (especially language links [19]) of an article. It is restricted to return only the language links for 50 titles. This is enough because the request for the articles return only 50 articles at maximum.

There are a few parameters to set:

- The **continue** parameter is an offset when more language links are available.
- **limit** is the maximum amount of links that are returned for this article.

The *action* command is *query* and to indicate that this request is for language links the *prop* attribute has to be set to *langlinks*. The article titles are split by a "|" character and assigned to the *titles* parameter. In this approach the URI [20] is used for articles $a_1, ..., a_n$. The limit is set to the maximum of 500.

[16] `http://www.mediawiki.org/wiki/API:Query_-_Lists`

[17] All namespaces with their correspoding id is enumerated by the URL `https://de.wikipedia.org/w/api.php?action=query&meta=siteinfo&siprop=namespaces`.

[18] `http://en.wikipedia.org/w/api.php?srprop=&srredirects=true&srlimit=50&action=query&srwhat=text&list=search&format=json&srinfo=suggestion&srsearch=t`

[19] `http://www.mediawiki.org/wiki/API:Properties#langlinks_.2F_ll`

[20] `http://en.wikipedia.org/w/api.php?lllimit=500&prop=langlinks&action=query&titles=a1|...|an&format=json`

3.2 Framework for multiple matchers and their evaluation

A small and simple framework for combining and evaluating the approaches was developed. To improve matching systems it is important to know which wrong mappings are made as well as which alignments are found. Therefore a new evaluation class has been implemented. It is shown in figure 22. The two errors that can occur (described in section 1.4.5) are depicted as the two methods in *Evaluator* called *getTooMuch* and *getNotFound*.

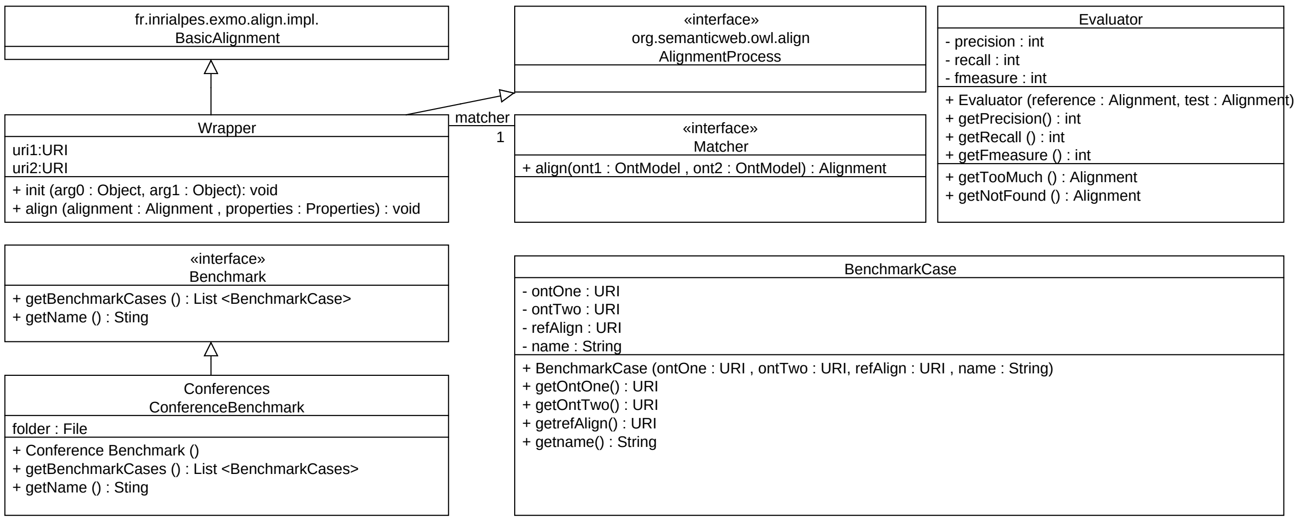

fr.inrialpes.exmo.align.impl.
BasicAlignment
«interface»
org.semanticweb.owl.align
AlignmentProcess
Evaluator
- precision : int
- recall : int
- fmeasure : int
+ Evaluator (reference : Alignment, test : Alignment)
+ getPrecision() : int
+ getRecall () : int
+ getFmeasure () : int
+ getTooMuch () : Alignment
+ getNotFound () : Alignment
Wrapper
uri1:URI
uri2:URI
+ init (arg0 : Object, arg1 : Object): void
+ align (alignment : Alignment , properties : Properties) : void
matcher
1
«interface»
Matcher
+ align(ont1 : OntModel , ont2 : OntModel) : Alignment
«interface»
Benchmark
+ getBenchmarkCases () : List <BenchmarkCase>
+ getName () : Sting
Conferences
ConferenceBenchmark
folder : File
+ Conference Benchmark ()
+ getBenchmarkCases () : List <BenchmarkCases>
+ getName () : Sting
BenchmarkCase
- ontOne : URI
- ontTwo : URI
- refAlign : URI
- name : String
+ BenchmarkCase (ontOne : URI , ontTwo : URI, refAlign : URI , name : String)
+ getOntOne() : URI
+ getOntTwo() : URI
+ getrefAlign() : URI
+ getname() : String

Figure 22: Matching framework with an interface called *Matcher* as the abstraction of every matching approach. To be conform with the *SEALS* platform a wrapper is implemented.

The simple structure of an ontology matcher (described in section 1.4.2) is implemented as an interface called *Matcher*. The parameter of the align method are two ontologies (in case of jena *OntModel*) and an input alignment. It returns a resulting alignment. Every matcher should implement this interface to abstract from the detailed implementation.

To simplify the run of different benchmarks, all of them are downloaded to a specified location saved in each benchmark like in *ConferenceBenchmark*. Each of them has a list of *BenchmarkCases* containing the two to be mapped ontologies, the reference alignment and a name for this benchmark case. The *Wrapper* class ensures that all alignments can also be evaluated with the SEALS client. It uses the *BasicAlignment* and *AlignmentProcess* from the *Alignment API*.

3.3 Structure of WikiMatch

As a first step all URI fragments, labels and comments from each concept is extracted. An URI fragment is splitted with an "#" sign. So the fragment is the bold characters in the following URL
`http://cmt#`**writePaper**. This can contain information, but can also be an identifier like
`http://cmt_en#`**c-8737916-0567046**. The language tag from labels or comments are used to build the endpoint URI for the API. In case of missing tags and for all fragments, the major language tag is used (see algorithm in figure 21).

As a second step all extracted terms are preprocessed. Since it often occurs that there are camel case words, these types are split into two words. This applies also for words with underscores and hyphens. Therefore *oneTwo*, *one-two* and *one_two* are all split into $\{one, two\}$.

The search engine used in Wikipedia tries to find all words in the search term. There is a huge difference between "a man" [21] and "man" [22]. Hence, all so called stopwords are removed. Some of them in the

[21] `http://en.wikipedia.org/w/api.php?action=query&srwhat=text&list=search&srsearch=a%20man`
[22] `http://en.wikipedia.org/w/api.php?action=query&srwhat=text&list=search&srsearch=man`

English language are for example $\{a, about, above, after, again, against, all, am, an, and, any, are, ...\}$ [23]. To be able to also deal with terms in different natural languages, any stopwords for this language are removed. Thus all stopwords for each language have to be stored.

For a term t that can be one of fragment, label or comment the Wikipedia search engine returns a set of articles called S(t). Every term of one concept is compared to every term of the other concept. The maximum value of this cross product is then the similarity for this pair. For two concepts c_1 and c_2 from ontology O_1 and O_2 this value is computed with the formula (see (Hertling and Paulheim, 2012a))

$$max_{t_i \in \{label(c_i), fragment(c_i), comment(c_i)\}, i \in \{1,2\}} \frac{\#(S(t_1) \cap S(t_2))}{\#(S(t_1) \cup S(t_2))} \tag{18}$$

The set S(t) can contain articles in different languages, but only articles in the same language are compared. If the value computed in (18) exceeds an given threshold, a mapping is created.

To match two ontologies, all data properties, object properties and classes are compared to each other by computing the cross product. This ensures that only homogeneous mappings are generated. As described in section 1.4.4 this further ensures that the mappings are conform with OWL light/DL. All in all, this approach find only mappings between same types like data property to data property or class to class. This reduces also the runtime and complexity of the matching. Another advantage is to be able to adjust the thresholds for every type of mapping individually. It is thus possible to decrease or increase the threshold for a matching between data property and data property to get better results. The whole structure of *WikiMatch* is depicted in figure 23.

The ontology is read through jena and for every fragment, label and comment, all article titles are requested. After requesting all language links per article, all titles in the same language are compared to find the maximum overlapping.

If the first query returns an suggestion (intended by the search URI), another request with the new term is made. The returned article titles are then conjoined with the titles from the result of the first search.

3.4 Search approaches

The simple idea to put the whole preprocessed term directly to the search engine, yields a better precision than recall in the first tests. This approach is called *Simple Search Approach* (SSA).

To get a better recall a second approach is also implemented and evaluated. The idea is to search the Wikipedia for each token individually and combing the results afterwards. This is called *Individual Token Search Approach* (ITSA). The tokens of a term is a result of the preprocessing step and the split of words. The idea is based on the often long comments so that no articles are returned. The reason is the search engine, because it tries to find all words in the search term and this will not work if there are too many.

The preprocessing step transforms the term *memberOfConference* first into $\{member, of, conference\}$. The token *of* is removed, because it is contained in the stopword list of the English language. After all, the *Simple Search Approach* made one request with "member conference". Whereas the *Individual Token Search Approach* create two queries with "member" and " conference".

Figure 24 shows the algorithm of WikiMatch for both search approaches.

3.5 Implementation of Simple Search Approach

The implementation of *Simple Search Approach* is done with java and the jena framework as well as the framework described in section 3.2. Therefore the ontology is read by jena and the matching systems only have to deal with *OntModel* as a representation of an ontology.

The overall matcher is contained in the class *WikiMatch*. It implements the *align* method defined by the *Matcher* interface from the framework. The first thing is to find out the language of both ontologies

[23] `http://www.ranks.nl/resources/stopwords.html`

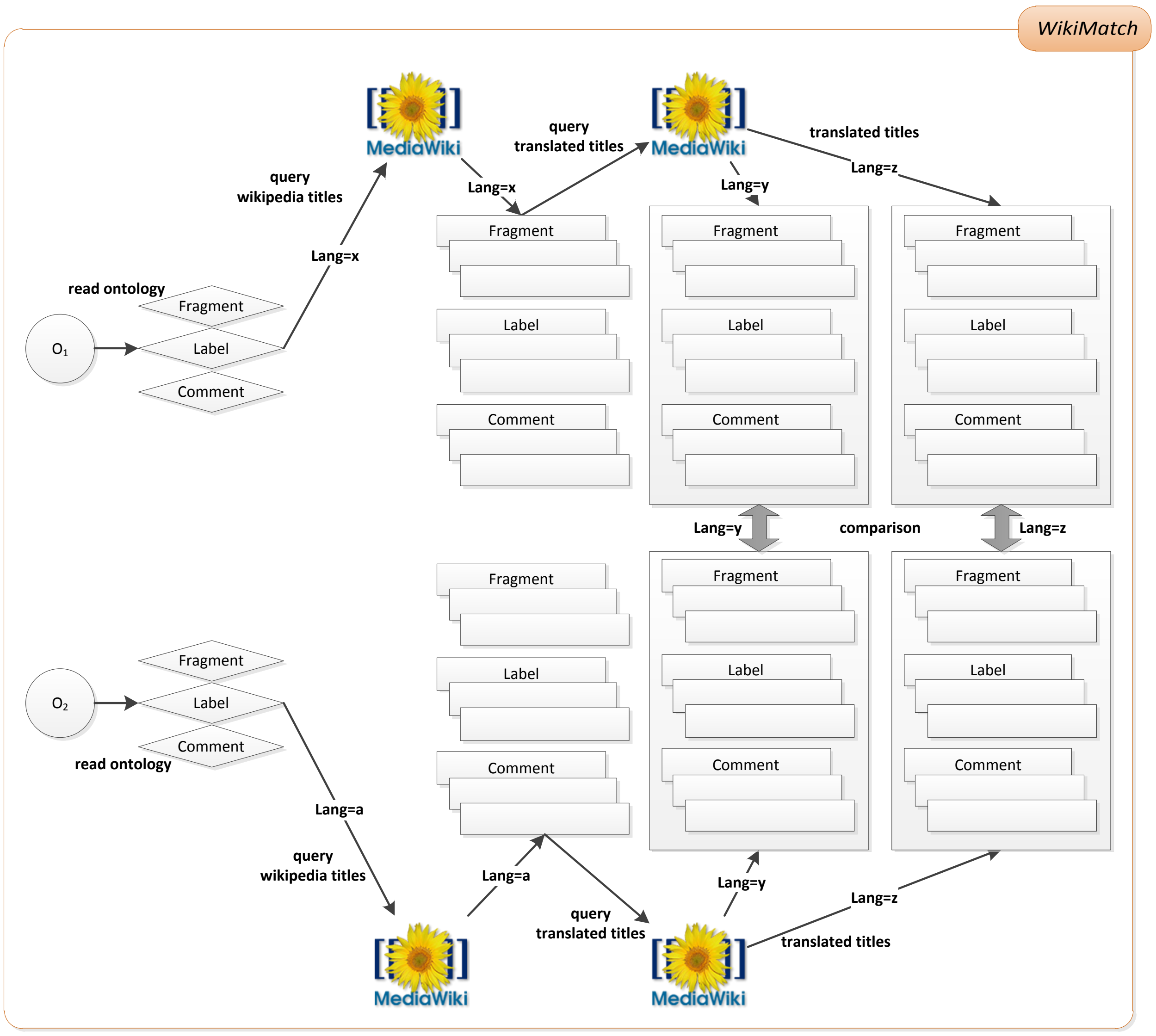


Figure 23: The structure of WikiMatch from an abstract point of view: For every fragment, label and comment in language x all Wikipedia titles are requested. As a second step, all language links are queried from these titles. This can be in language y and z. All fragment, label and comments titles are then compared to each other with respect to the language (see (Hertling and Paulheim, 2012a)).

```
float getsimilarity(term1, term2) {
  titlesForTerm1 = getAllTitles(term1);
  titlesForTerm2 = getAllTitles(term2);

  commonTitles = intersectionOf(titlesForTerm1, titlesForTerm2);
  allTitles = unionOf(titlesForTerm1, titlesForTerm2);

  return #(commonTitles) / #(allTitles);
}

List<WikipediaPage> getAllTitles(searchTerm) {
  removeStopwords(searchTerm);
  removePunctuation(searchTerm);

  if(simpleSearch) {
    resultList = searchWikipedia(searchTerm);
  }

  if(individualTokenSearch) {
    tokens = tokenize(searchTerm);
    for each token in tokens
      resultList = resultList + searchWikipedia(searchTerm);
  }

  for each page in results
    resultList = resultList + getLanguageLinks(page);

  return resultList;
}
```

Figure 24: Algorithm for WikiMatch explaining the basic idea as well as the difference between the two search approaches (see (Hertling and Paulheim, 2012a)).

by calling the *getDefaultLanguage* method (see listing 2 in the appendix) and adjust the thresholds for the *iterateOver* function. The thresholds are much lower in an multilingual case than in a monolingual case. The *iterateOver* method gets two lists of resources, respectively concepts. This can be in both cases a class, data property or object property list. It calculates the cross product of these two lists and adds an align cell to the returned alignment, when the similarity value exceeds the given threshold.

To calculate the similarity between two concepts, *WikiMatch* uses the *ResourceMatcher*. The extracted languages are set by the *setLanguage1* and *setLanguage2* methods. The same applies for the resources. If a mapping is created, it will be neccessary to have access to the URIs. This is provided by *getURI1&2*. The static *match* method compares the two *Resource* instances by calling the same method on the *Resource* class.

The constructor of the *Resource* class is private to only get an instance by calling *create*. It accepts an *OntResource* and a default language. Through this pattern it is possible to look up for already created *Resource* instances in the *create* method. The *getAll* method is only used in the *match* function to retrieve all fragments, labels and comments as a set of instances of *SubResource*. The *match* method returns the maximum value by comparing the two sets. This is done by *SubResource.match(a, b)*.

A *SubResource* represents a term like a fragment, label or comment which is stored in the attribute *text*. There is also a map for saving already made queries (*savedQueries*). The pattern to create instances also applies for this class. The interesting parts are the methods for requesting titles and translations. *AskForTitles* has the text and an integer as parameter. The loop parameter is an accumulator for a recursion, because there can be more suggestions. The method returns all titles inclusive all titles of the suggestions as a list of strings. This is the input of *askForTranslated* to find all language links in the articles. All results are stored in the *languageMap* which has the language as key and an instance of *Titles* as value.

The *Titles* class contains a map called *titles* in which the article title is the key and the position of the original title is the value of the map. It can be instantiated with an *initialList*. The two static methods in this class return the intersection and union of two *Titles* instances to compute the similarity in the *match* method in *SubResource* like in equation (18).

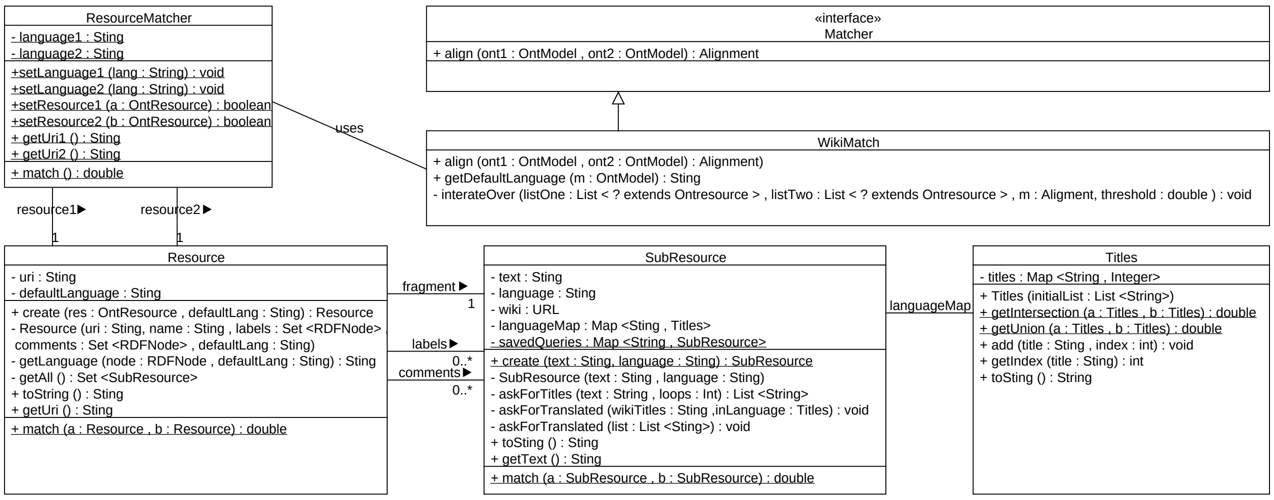

ResourceMatcher
- language1 : Sting
- language2 : Sting
+setLanguage1 (lang : String) : void
+setLanguage2 (lang : String) : void
+setResource1 (a : OntResource) : boolean
+setResource2 (b : OntResource) : boolean
+ getUri1 () : Sting
+ getUri2 () : Sting
+ match () : double
uses
«interface»
Matcher
+ align (ont1 : OntModel , ont2 : OntModel) : Alignment
WikiMatch
+ align (ont1 : OntModel , ont2 : OntModel) : Alignment)
+ getDefaultLanguage (m : OntModel) : Sting
- interateOver (listOne : List < ? extends Ontresource > , listTwo : List < ? extends Ontresource > , m : Aligment, threshold : double) : void
resource1▶
resource2▶
1
1
Resource
- uri : Sting
- defaultLanguage : Sting
+ create (res : OntResource , defaultLang : Sting) : Resource
- Resource (uri : Sting, name : Sting , labels : Set <RDFNode> , comments : Set <RDFNode> , defaultLang : Sting)
- getLanguage (node : RDFNode , defaultLang : Sting) : Sting
- getAll () : Set <SubResource>
+ toString () : Sting
+ getUri () : Sting
+ match (a : Resource , b : Resource) : double
fragment ▶
1
labels▶
0..*
comments▶
0..*
SubResource
- text : Sting
- language : Sting
- wiki : URL
- languageMap : Map <Sting , Titles>
- savedQueries : Map <String , SubResource>
+ create (text : Sting, language : Sting) : SubResource
- SubResource (text : Sting , language : Sting)
- askForTitles (text : String , loops : Int) : List <String>
- askForTranslated (wikiTitles : Sting ,inLanguage : Titles) : void
- askForTranslated (list : List <Sting>) : void
+ toSting () : Sting
+ getText () : Sting
+ match (a : SubResource , b : SubResource) : double
languageMap
Titles
- titles : Map <String , Integer>
+ Titles (initialList : List <String>)
+ getIntersection (a : Titles , b : Titles) : double
+ getUnion (a : Titles , b : Titles) : double
+ add (title : Sting , index : int) : void
+ getIndex (title : Sting) : int
+ toSting () : String

Figure 25: UML diagramm of the implementation of *Simple Search Approach*

3.6 Composition of WikiMatch and filters

WikiMatch as one matcher does the job of finding mappings between ontologies, but there are some post-processing steps to further improve the results. Since the reference alignment does not contain mappings that are in fact true, these have to be removed with a second matcher.

It is called *OriginalHostsFilter* and is also used in (Dang et al., 2012). The main approach is to allow only mappings, where the host URI is equals to the URIs from the ontologies. This makes no sense at first, but frequently used upper level ontologies (see section 1.4.6) are matched together like

$$< http://xmlns.com/foaf/0.1/firstName, \tag{19}$$
$$http://xmlns.com/foaf/0.1/firstName, \tag{20}$$
$$=, \tag{21}$$
$$1.0 > \tag{22}$$

because it is used in both input ontologies.In this case it is the *firstName* property of the so called FOAF (Friend of a Friend) ontology. It can also be removed, when choosing only different URIs in a mapping, but this variant is a little bit stricter. It allows only mappings that don't contain any other concepts than the ones in the ontology. Thus a mapping between `http://xmlns.com/foaf/0.1/firstName` and `http://myOntology#firstName` is also removed. In this case it is intended because the reference alignment does not contain any such mappings.

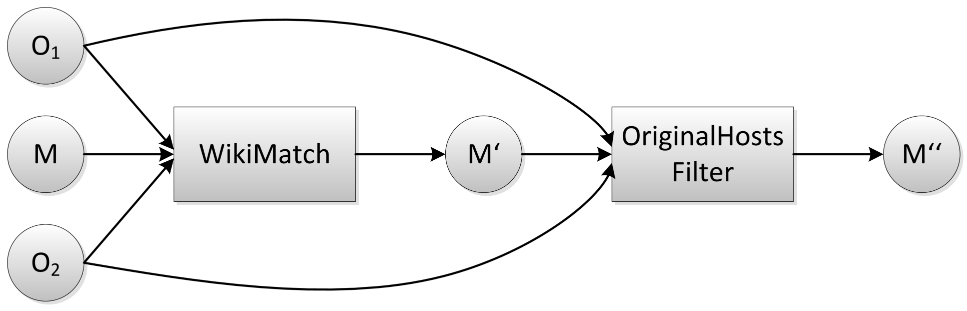


Figure 26: WikiMatch in combination with a filter

The overall stucture is depict in figure 26 with WikiMatch as first matcher and OriginalHostsFilter as a second one. Both are arranged in a sequential composition (see section 1.4.3).

4 Evaluation

The evaluation of this approach is subdivided into two parts. On the one hand the matching system with the *simple search approach* is submitted to an evaluation initiative described in section 4.1. The results are discussed in the following sections. On the other hand the different search ideas are evaluated against each other in section 4.2 with a subset of the tracks. In the results of the OAEI WikiMatch is sometimes abbreviated by Wmatch.

4.1 Ontology Alignment Evaluation Initiative

The *Ontology Alignment Evaluation Initiative* [24] (OAEI) was founded in 2004 and identifies several tracks for ontology matching (Euzenat and Shvaiko, 2007, p.195). Since the 2012 campaign there is also an track for instance matching [25]. All ontology matching tools have to report the results in the alignment format described in section 1.4.4. The *Alignment API* is very useful in this case, because it produces the alignments in the correct format.

The matching process itself is executed within the *SEALS platform* [26] to get a better feedback to the participants as well as a more automatic evaluation of the tools.

The metrics that are measured are recall, precision and the computed F-measure. The runtime as a performance measure is also evaluated to compare the matching systems. In the 2012 campaign 21 matchers participated in total.

[24] `http://oaei.ontologymatching.org`
[25] `http://oaei.ontologymatching.org/2012/`
[26] `http://www.seals-project.eu/`

4.1.1 Evaluation on *benchmark*

The benchmark track is a systematic test series. It tries to discover the strong and weak parts of a matching system. The main purpose is to measure the progress over the years with stable tracks [27].

There are 5 subtracks each systematically changed. The first test is about the domain of Bibliographic references (biblio) with 97 classes and properties. The next domains are commerce (benchmark 2), bioinformatics (benchmark 3), product design (benchmark 4) and finance with 247, 354, 472 and 633 classes and properties.

Each can be modified with the following changes (see [28]):

- the **names** can be replaced by random strings(R), synonyms (S), translations (F) and names with different conventions (C)
- **comments** can be suppressed (N) or translated(F)
- the **taxonomy** can be suppressed (N), expanded (E) or flattened (F)
- **instances** can be suppressed (N)
- **properties** can be suppressed (N) or no restrictions (R)
- **classes** can be expanded (E) or flattened (F)

The tracks are modified with one or more of these changes to find out the weakness and strangeness of the matchers. A few examples are given in the table 2.

#	Name	Comment	Taxonomy	Instance	Property	Class	Comment
101	0	0	0	0	0	0	Reference alignment
102							Irrelevant ontology
103	0	0	0	0	0	0	Language generalization
104	0	0	0	0	0	0	Language restriction
201	R	0	0	0	0	0	No names
202	R	N	0	0	0	0	No names, no comments
203	0	N	0	0	0	0	No comments (was misspelling)
204	C	0	0	0	0	0	Naming conventions
205	S	0	0	0	0	0	Synonyms
206	F	F	0	0	0	0	Translation
207	F	0	0	0	0	0	

Table 2: Some of the test cases in the benchmark test of the OAEI (see (Aguirre, 2012a))

The first test case is the same ontology. Thus it is an easy task to find all mappings with string comparison. Test 102 is a completely different ontology to see if the matcher also finds mappings that are certainly false. The following test and test 104 is an generalization in OWL Lite. Some language concepts like union are thus removed. The numbers starting with “2” are all systematic tests. 201 for example replaces all names by random strings. 202 additionally suppresses all comments. The names and comments are translated in test 206.

WikiMatch reaches an F-measure of 0.62 in the biblio track. As a results WikiMatch is much better than the baseline edna with an F-measure of 0.41. Detailed results are depicted in figure 34 in the appendix.

On the individual tracks showed in table 2, WikiMatch performs well on the track where either names or comments are removed or changed. If both information are removed the foundation of this idea can not apply anymore and will result in a few or no mappings at all. Since the approach is fully element based, the results will nearly be the same when removing structural information.

[27] http://oaei.ontologymatching.org/2012/benchmarks/index.html

[28] http://oaei.ontologymatching.org/2012/benchmarks/index.html

4.1.2 Evaluation on *anatomy*

The anatomy track consists of two large ontologies describing the Adult Mouse Anatomy as well as the human anatomy[29] (part of the NCI Thesaurus [30]). There are a huge amount of trivial mappings, because the baseline (in this case the string equivalence) has an recall of 0.622 and a high precision of 0.997. On the other hand some non-trival mappings have to be found to increase the recall. This can be done with medical background knowledge or any other resource. It is also possible to use structural approaches, because there are many *partOf* relations.

WikiMatch performs worse in this track compared to the baseline, because it finds more mappings but often not the right ones. This is expressed through the higher recall of 0.675 and the lower precision (0.864).

One non-trivial mapping that is found by *WikiMatch* is *ophthalmic artery* and *Opthalmic Artery*. The spelling mistake is corrected by the suggestion of Wikipedia.

4.1.3 Evaluation on *conference*

The conference track is composed of 16 ontologies from the conference domain, developed within the OntoFarm project (Svab et al., 2005). The reference alignments covers 7 ontologies compared to each other. Therefore 21 pairs can be automatically tested. If a matching system finds complex mappings, these are automatically evaluated [31]. The ontologies have different origins, which means that they describe the same domain with very various point of views.

In this track the semantic information is often contained in the URI fragments. Thus it is necessary for this track to evaluate this type of information.

The reference alignments are split into two variants. *Ra1* is the original reference alignment, whereas *ra2* is an entailed reference alignment. It is the transitive closure of *ra1*. Therefore this alignment is more complete and correct than the original one.

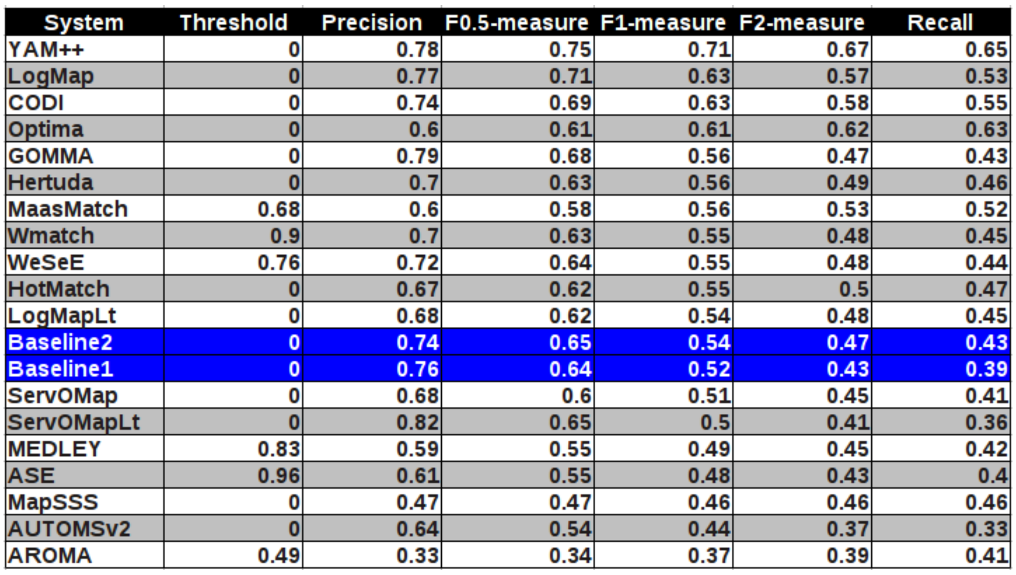
The following table is an abstract of the one provided by the OAEI [32]. It contains only the ontologies where a reference alignment is available. The table 3 shows the different ontologies and their various resources that are used to create it.

Name	Type	Classes	Datatype Properties	Object Properties	DL expressivity	Related link
Ekaw	Insider	74	0	33	SHIN	`http://ekaw.vse.cz`
Sigkdd	Web	49	11	17	ALEI(D)	`http://www.acm.org/sigs/sigkdd/kdd2006`
Cmt	Tool	36	10	49	ALCIN(D)	`http://msrcmt.research.microsoft.com/cmt`
Edas	Tool	104	20	30	ALCOIN(D)	`http://edas.info/`

Table 3: Different types of ontologies describing the same conference domain

Insider means that the ontology is developed by people with experience in the domain. The *Web* type instead means that it is based on conferences and their web pages. *Cmt* and *Edas* are tools for supporting the organization of a conference. The ontologies are extracted from these tools and thus have the type *Tool*. The results in the column *DL expressivity* are based on the Pellet reasoner.

[29] http://oaei.ontologymatching.org/2012/anatomy/index.html
[30] http://ncit.nci.nih.gov/
[31] http://oaei.ontologymatching.org/2012/
[32] http://oaei.ontologymatching.org/2012/conference/index.html



System	Threshold	Precision	F0.5-measure	F1-measure	F2-measure	Recall
YAM++	0	0.78	0.75	0.71	0.67	0.65
LogMap	0	0.77	0.71	0.63	0.57	0.53
CODI	0	0.74	0.69	0.63	0.58	0.55
Optima	0	0.6	0.61	0.61	0.62	0.63
GOMMA	0	0.79	0.68	0.56	0.47	0.43
Hertuda	0	0.7	0.63	0.56	0.49	0.46
MaasMatch	0.68	0.6	0.58	0.56	0.53	0.52
Wmatch	0.9	0.7	0.63	0.55	0.48	0.45
WeSeE	0.76	0.72	0.64	0.55	0.48	0.44
HotMatch	0	0.67	0.62	0.55	0.5	0.47
LogMapLt	0	0.68	0.62	0.54	0.48	0.45
Baseline2	0	0.74	0.65	0.54	0.47	0.43
Baseline1	0	0.76	0.64	0.52	0.43	0.39
ServOMap	0	0.68	0.6	0.51	0.45	0.41
ServOMapLt	0	0.82	0.65	0.5	0.41	0.36
MEDLEY	0.83	0.59	0.55	0.49	0.45	0.42
ASE	0.96	0.61	0.55	0.48	0.43	0.4
MapSSS	0	0.47	0.47	0.46	0.46	0.46
AUTOMSv2	0	0.64	0.54	0.44	0.37	0.33
AROMA	0.49	0.33	0.34	0.37	0.39	0.41

Figure 27: Results of WikiMatch on the conference track ra2 (see (Šváb Zamazal, 2012)).

WikiMatch reaches in ra2 an F1-measure of 0.55 and is a little bit better than baseline 2 (see figure 27). Baseline 1 lowers the locales names at first, then creates a mapping if they are equals. Baseline 2 removes additionally dashes, underscores and the keyword “has” from all terms as previous step.

WikiMatch is not able to find some more difficult mappings like `http://confOf#Member_PC` and `http://sigkdd#Program_Committee_member`, because the conference domain is not that detailed in Wikipedia. The abbreviation PC (program committee) is for example not found with a specialized search, [33] not even found though the snippets. But instead it finds a mapping like *Sponsorship* = *Sponzorship* though the suggestion feature.

WikiMatch needs 40 minutes to match all 120 testcases. It is the most time consuming system on this track of the OAEI. The main reason is the great amount of Wikipedia queries that have to be requested.

4.1.4 Evaluation on *Multifarm*

Multifarm is a track especially for multilingual ontologies. It is created by translating the conference ontologies described in section 4.1.3. The eight target languages are Chinese, Czech, Dutch, French, German, Portuguese, Russian, and Spanish (Meilicke et al., 2012). Altogether, there are 36 language pairs and 5 used ontologies (cmt, conference, confOf, iasted, sigkdd). For each pair 25 reference alignments are available, because every ontology is compared to a translated one which includes the same ontology like cmt-cmt-de-en. The evaluation can therefore be split up into two different kinds. If the ontology is the same and it is only translated to another language this is called type 2 (same ontologies). Type 1 (different ontologies) is a comparison between two different ontologies in different languages like cmt-conference-de-en. This means that cmt is translated to German and conference is translated to English.

Table 4 shows the results for the multifarm tracks. WikiMatch needs 1072 minutes (which is 17 hours and 52 minutes) to finish all $36 * 25 = 900$ tracks. It uses the language links from Wikipedia to also be able to match multilingual input ontologies. Though the external resource it is again the slowest matcher on the track. One mentionable aspect is the fact that WikiMatch is one of a few matchers which are able to match all language pairs successfully. This includes also the Chinese and Russian languages. Only YAM, GOMMA, WeSeE and WikiMatch (in order of their F-mesure; best first) were able to match Chinese against Czech. Detailed results are depicted in the appendix in figure 35.

[33] `http://en.wikipedia.org/w/index.php?search=program+committee&title=Special%3ASearch`

		Different ontologies (i)			Same ontologies (ii)		
System	Runtime	Precision	Fmeasure	Recall	Precision	Fmeasure	Recall
AUTOMSv2	512.7	.49	.36	.10	.69	.24	.06
GOMMA	35.0	.29	.31	.36	.63	.38	.29
MEDLEY	76.5	.16	.16	.07	.34	.18	.09
WeSeE	14.7	.61	.41	.32	.90	.41	.27
Wmatch	1072.0	.22	.21	.22	.43	.17	.11
YAM++	367.1	.50	.40	.36	.91	.60	.49

Table 4: Multifarm results for specific multilingual matchers (see (Trojahn dos Santos, 2012)). Runtime is measured in minutes.

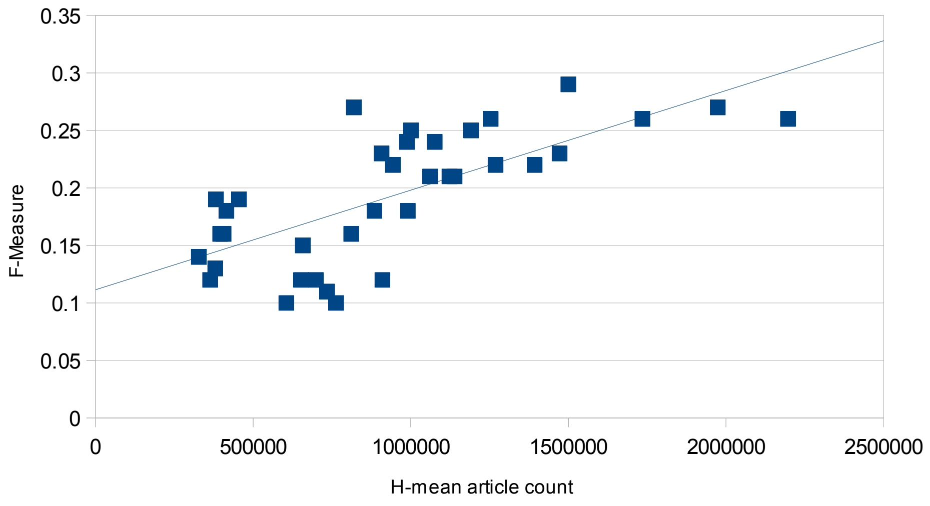


Figure 28: Multifarm results in relation to the article amount of Wikipedia in the special language. (see (Hertling and Paulheim, 2012b)).

The F-measure over all language pairs of 0.21 on type 1 track is better than on type 2 which is only 0.17. With these results WikiMatch was penultimate in the category of specific multilingual matchers in type 1 and unfortunately the last one on type 2.

The results depend on the amount of articles which are available in this language specific Wikipedia as well as the links between different languages. Figure 28 describes the results of WikiMatch in correspondence to the article amount. Figure 29 depicts the relation between the amount of language links and the results of the specified track. Both figures show that if Wikipedia has more language links and more articles, the matching results will be better. With additional 500,000 articles, the F-measure increase by approximately 0.05 percent. Such a progression needs 874 days for the Chinese Wikipedia [34] (abbreviation: zh) by taking the growth of September 2012 as a basis (which are 572 new articles per day). The English Wikipedia instead needs only 570 days to reach the same increase of articles.

4.1.5 Evaluation on *Library*

The goal of the library track is to match two lightweight ontologies with a large amount of classes. Both ontologies are imported from real world thesaurus namely STW and the TheSoz. They are developed in German but have English translations as well [35].

The goal of the OAEI is to use more real world data and to find out how the participating matchers deal with this kind of ontology. The STW Thesaurus is about any economic subject. It has 6,000 subject

[34] `http://stats.wikimedia.org/EN/TablesArticlesNewPerDay.htm`

[35] `http://web.informatik.uni-mannheim.de/oaei-library/results/2012/`

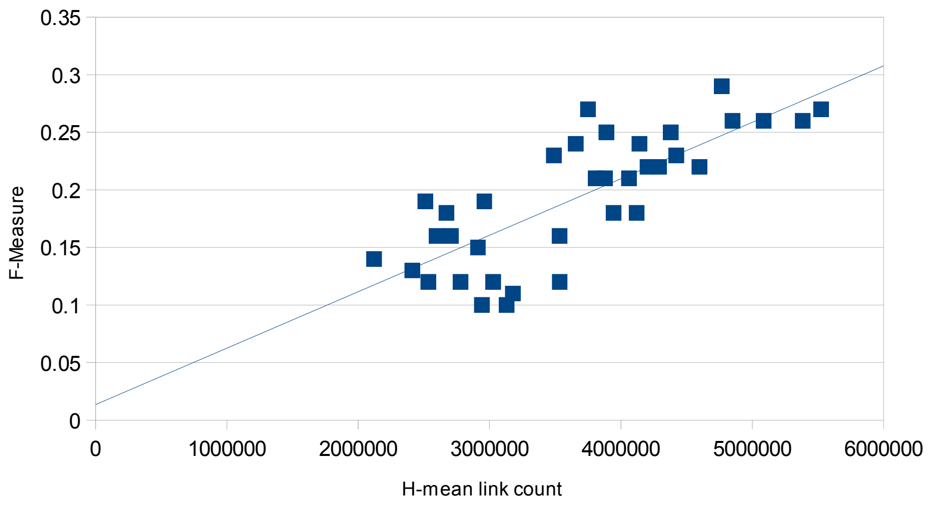


Figure 29: Multifarm results in relation to the amount of language links to other wikipedias (see (Hertling and Paulheim, 2012b)).

headings and 19,000 keywords in German and English. Moreover there is a little bit of structure with related and narrower relations. TheSoz is about the Social Sciences. There are 8,000 headings and 4,000 keywords.

A reference alignment is already made through the KoMoHe project (Mayr and Petras, 2008) by domain experts. Unfortunately, the alignment is only on the state of 2006.

To compare the matcher, three baselines are given. With *MatcherPrefDE* which compares the German labels (case insensitive) and create mappings if they are equals, it is possible to reach an F-measure of 0.717 in 42 seconds.

WikiMatch instead is not capable of finishing this task within 1 week, because of the huge amount of descriptions.

4.1.6 Evaluation on *Large Biomedical Ontologies*

The track *Large Biomedical Ontologies* consists of three ontologies describing the biomedical domain:

- FMA (Foundational Model of Anatomy [36])
- SNOMED (SNOMED CT [37])
- NCI (National Cancer Institute Thesaurus [38])

The target of this track is to find out how the matcher behaves with large ontologies. Therefore the track is also called *largebio*.

Altogether there are three subtasks: FMA-NCI, FMA-SNOMED and SNOMED-NCI. Each ontology exists in three different sizes, namely small fragments, large fragments and whole ontologies. To get an idea of the size the whole ontology of NCI has 66,724 classes and FMA has 78,989 classes. The SNOMED ontology is much larger because in this track only 40% of it are used, but these are still 122,464 classes.

The provided reference alignment is based on UMLS Metathesaurus (Bodenreider, 2004). The extraction of the alignments leads to many unsatisfiable classes [39]. With the help of LogMap's repair facility a refinement was created (Jiménez-Ruiz and Grau, 2011). Another one is provided though the Alcomo debugging system (Meilicke, 2009).

[36] http://sig.biostr.washington.edu/projects/fm/
[37] http://www.ihtsdo.org/index.php?id=545
[38] http://ncit.nci.nih.gov/
[39] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/oaei2012_umls_reference.html

WikiMatch was only able to match FMA-NCI small fragments within the given time. The input ontologies of this track have 3,696 and 6,488 classes. It lasts 65,399 seconds which are approximately 18 hours to get an F-measure of 0.831 in the original UMLS alignment. WikiMatch is on the 9th place of totally 14 matchers.

4.2 Evaluation of different search approaches

To compare the different search approaches (*simple search* (SSA), *individual token search* (ITSA)) both are evaluated on conference and multifarm track. It is often the case that simple search has higher precision but a lower recall. Therefore individual token search is also implemented and evaluated. On many tracks of the OAEI there are comments with long descriptions and therefore each term is handled individually. This approach can thus yield in better recall, but the precision will maybe decrease.

All following diagrams contain precision, recall and F-measure for different values of the threshold t calculated in equation (18).

4.2.1 Evaluation on *conference*

The conference track contains some long descriptions especially in the comments. Thus, the idea of searching each token can yield in a better F-measure.

The SSA (depicted in figure 30) reaches a maximum F-measure of 0.610 for $t = 0.5$. For thresholds above 0.25, there are equal values with very small variations. The maximum recall is 0.6 at $t = 0.05$ which is only lowered a little bit to 0.534 at $t = 0.25$. This shows that not many correct mappings are lost. The precision instead increases by 50 percent between $t = 0.05$ and $t = 0.25$.

The ITSA (depicted in figure 31) has a very different characteristic. The recall is much higher ($r = 0.867$) at the beginning because more mappings are found when using each token individually. This will be much lower when the thresholds exceeds 0.5, but the precision then increases by 52.6 percent. The maximum F-Measure can be achieved with $t = 0.7$, reaching the value of 0.611.

All in all, both search approaches yield in approximately 0.61 F-measure with very different thresholds. They have various characteristics by reaching this goal, but the result is nearly equal. The runtime of SSA is 22 minutes and 20 seconds, whereas ITSA needs a little bit longer (24 minutes and 14 seconds) [40] (Hertling and Paulheim, 2012a).

4.2.2 Evaluation on *Multifarm*

The *Multifarm* includes translated ontologies to many different languages. WikiMatch uses the language links of every article to also match this type of input ontologies. The threshold for both search approaches is much lower than on the conference track. One possible reason is that descriptions with different natural languages need a much lower amount of overlapping articles.

The SSA (depicted in figure 32) has a maximum F-measure of 0.210 with $t = 0.06$. The best reached recall is 0.35 and is continuously decreasing with greater values of the threshold. With $t = 0.3$ the value is 0.014 which is to low to get a appropriate F-Measure.

ITSA (depicted in figure 33) achieved a lower F-measure of 0.179 (with $t = 0.06$). Recall and precision behave similarly to SSA, but the best values for *Multifarm* are provided by SSA. Therefore WikiMatch with SSA is participating in OAEI.

5 Outlook

In this thesis, *WikiMatch* is introduced as an element based matcher. The approach is based on Wikipedia as an external resource. Since Wikipedia is an online encyclopedia with many volunteers around the world, it will always be up to date. 10028 articles will be added every day in the Wikipedias altogether [41]. This approach can thus handle any domains that are available in this resource.

[40] The tracks are executed on a Windows 7 64bit PC with an Intel i7(3.4 GHz) processor and 8 GB RAM
[41] `http://stats.wikimedia.org/EN/TablesArticlesNewPerDay.htm`

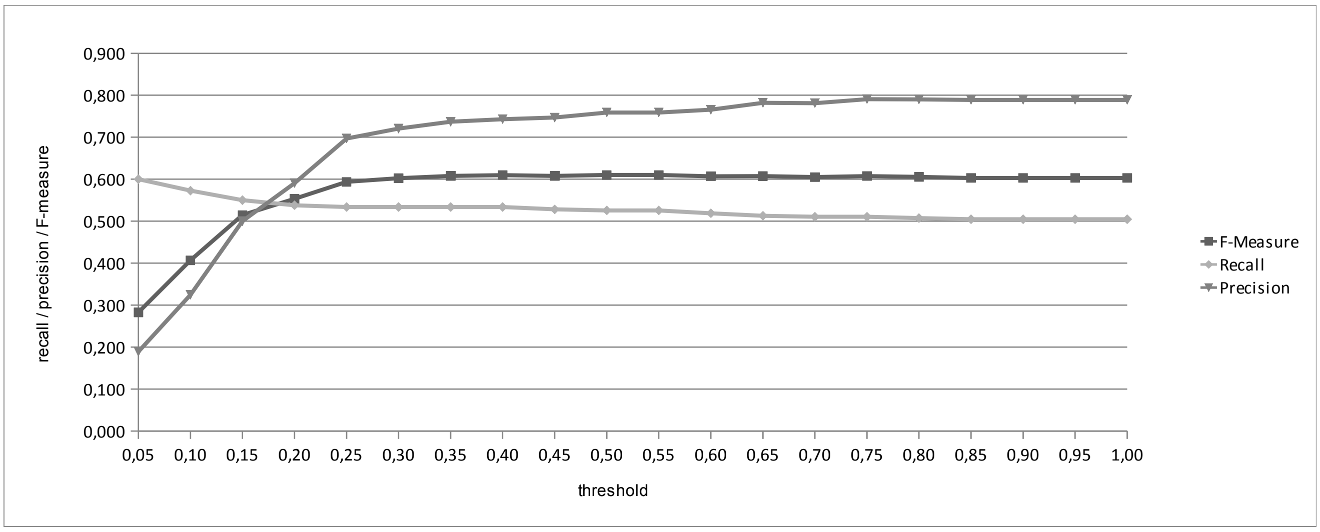


Figure 30: Results of *Simple Search Approach* on OAEI conference track. Average F-measure, precision and recall are measured (see (Hertling and Paulheim, 2012a)).

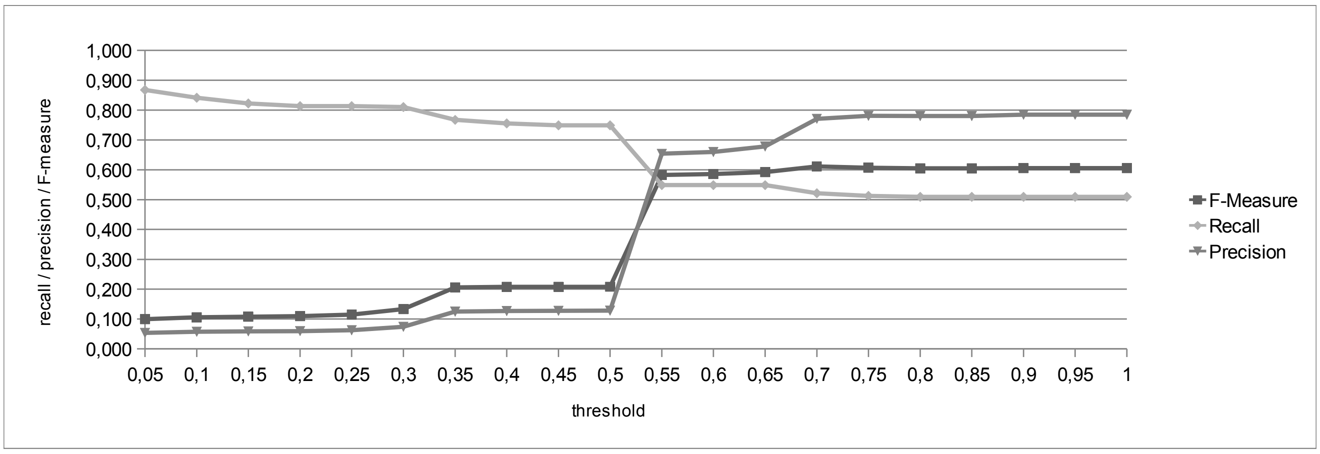


Figure 31: Results of *Individual Token Search Approach* on OAEI conference track. Average F-measure, precision and recall are measured (see (Hertling and Paulheim, 2012a)).

In the evaluation section the main problem which occurs is the runtime of this approach. Thus, the follwing sections discuss some improvements of this matcher with regard to the time as well as the results.

5.1 Runtime optimization

The most problematic characteristic of WikiMatch is the runtime and complexity. Some of the tracks in section 4 did not finish within a given time. This was sometimes more than one week. There are two reasons for the long runtime. On the one hand there are many classes on certain special tracks, so the complexity of this approach is too high, on the other hand the queries for Wikipedia take a long time.

The complexity of this approach is shown in equality (23). The “#Class” function describes the amount of classes in one ontology, whereas the “#DataProp” does the same for datatype properties and “#ObjectProp” for object properties.

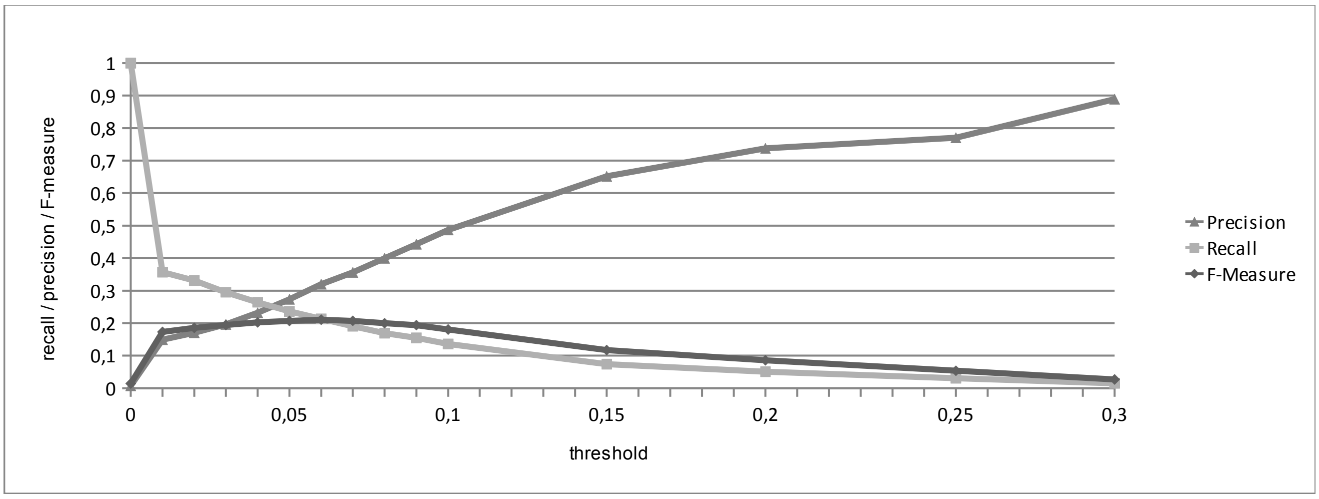


Figure 32: Results of *Simple Search Approach* on OAEI multifarm track. Average F-measure, precision and recall are measured (see (Hertling and Paulheim, 2012a)).

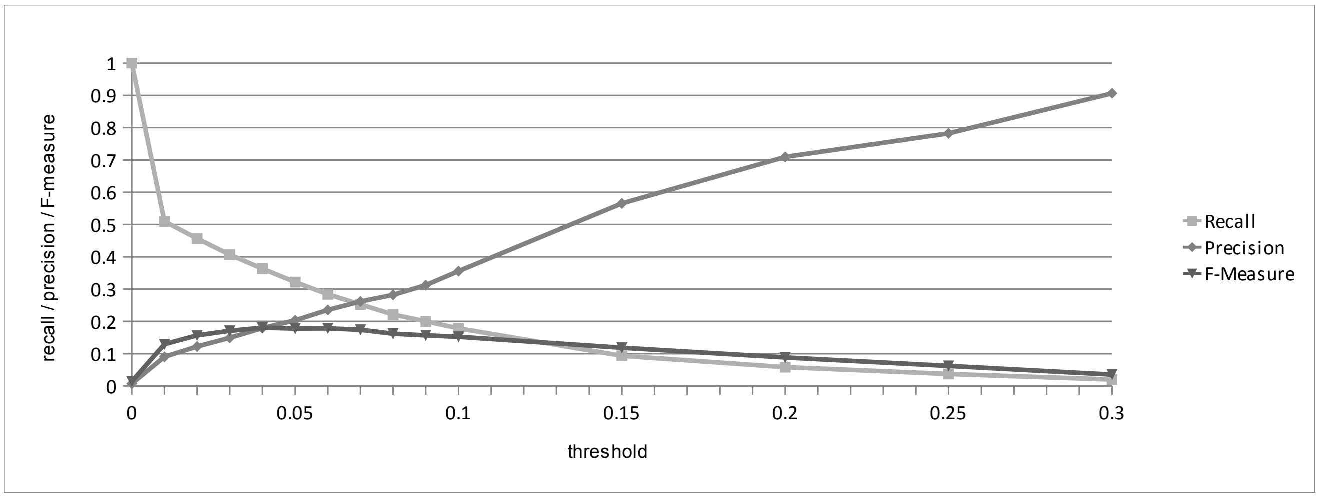


Figure 33: Results of *Individual Token Search Approach* on OAEI multifarm track. Average F-measure, precision and recall are measured (see (Hertling and Paulheim, 2012a)).

$$\mathcal{O}(\quad \#Class(O_1) * \#Class(O_2) + \tag{23}$$
$$\#DataProp(O_1) * \#DataProp(O_2) + \tag{24}$$
$$\#ObjectProp(O_1) + \#ObjectProp(O_2)) \tag{25}$$

One of the largest tracks available from the OAEI is SNOMED-NCI with the whole ontologies (see section 4.1.6). NCI contains 66,724 classes and SNOMED 122,464 classes. This results in a complexity of WikiMatch of $66,724 * 122,464 = 8171287936$ comparisons. If every comparison takes 0.5 seconds, the whole matching will need approximately 129 years. To finish this track within one week and with this complexity, one comparison may not exceed 0.074 milliseconds.

The other time consuming operation is to query all article titles and their translations from Wikipedia. For every distinct term in *Simple Search Approach* two queries are made. One query for retrieving the titles and one for requesting the language links. This is expressed by the following formula:

$$queries = \left|\{C(O_1), C(O_2), D(O_1), D(O_2), O(O_1), O(O_2)\}\right| \tag{26}$$

where C stands for all classes, D for datatype properties and O for object properties of the given ontology. If the language links can not be transmitted within one answer, another request with an offset is made until all language links are available.

5.1.1 Scalability and parallelization

In (Shvaiko and Euzenat, 2008), it is announced that one of the ten challenges of ontology matching is the performance of ontology-matching techniques. Matchers that use external resources are usually much slower than others. The main reason is that the resources are very slow and it takes time to receive the requested information. WordNet is often faster and locally available, but the processing is slower than a computation directly on the CPU.

It is often the case that these matchers use the Jaccard coefficient to get a similarity between two sets. It is defined as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{27}$$

In (Nakayama et al., 2007) it is called co-occurence analysis. They use also Wikipedia to find out some similarity values. In case of comparing two terms t_1 and t_2 based on Wikipedias search engine, the sets A and B in equation (27) contain articles where $t1$, respectively $t2$ appear. To calculate $J(A,B)$ it is thus necessary to create a query for every comparison of two terms. This yields in a quadratic number of requests. A similar approach like Google Distance (Cilibrasi and Vitanyi, 2007) has the same complexity.

WikiMatch requires only a linear number of searches calculated in equation (26). The reason is that this approach searches only for concepts and compares them locally instead of requesting combinations of concepts. Therefore this approach is scaleable to larger matching problems even if it's slow because of requesting articles for all terms that are used.

There are two possible ways to increase the absolute used runtime of WikiMatch. Since the approach is fully element based and does not need any structural information, the matching problem can be distributed to many computers and processes like in (Hu et al., 2008) and (Paulheim, 2008). An implementation is thus possible and will reduce the runtime.

The other improvement is combinable with the distribution of the matching problem. The idea is to retrieve more article titles and language links in one query. For some API calls, like searching articles and retrieving their language links, limitations are introduced. MediaWiki offers some special API permissions for researches [42]. This includes for example that 5,000 language links can be returned instead of only 500 [43]. With this step, the runtime will also be reduced and less queries have to be created.

5.1.2 Matcher pipeline

It is shown that WikiMatch is capable of finding special mappings with the help of Wikipedia. There are different ideas to improve runtime and results. One of the ideas is to build a whole pipeline of matchers to do some preprocessing as well as removing unsatisfiable mappings.

By putting a simple matcher in front of WikiMatch that compares the strings for equality, simple mappings like *book=book* can be found quicker and the follwing matchers would not have to worry about this. This will result in smaller runtimes because WikiMatch creates no query for the term *book*.

The postprocessing steps, like removing some mappings that are not in the reference mapping (described section 3.6), improve the results. Another possibility is to enforce a 1:1 mapping by removing all mappings with an equal URI but less confidence values.

To further increase the results, matchers based on the structural level can be placed after WikiMatch to use the resulting alignments as the initial ones. Some of them are explained in section 1.4.6. A

[42] `http://meta.wikimedia.org/wiki/Research:Special_API_permissions`

[43] `http://www.mediawiki.org/wiki/API:Properties#langlinks_.2F_ll`

graph based matcher can for example create a mapping between two properties of two already matched classes.

5.2 Determine the thresholds

Many approaches use thresholds to define from which similarity values on a mapping should be created. These thresholds can strongly differ, depending on the input ontologies.

One alternative can be the extraction of some characteristics of ontologies. This can include for example the average length of URI fragements, comments and labels. It is also possible to find out how strong the concepts are linked together and as a results putting a structural matcher in the pipeline or not.

All these values can be set in connection with the thresholds and the results alignment. The result will be a function depending on the characteristics of the input ontologies and returns the best threshold for this specific matching problem. Thus it will be possible to react to the ontologies.

One example for difference thresholds are the conference and Multifarm ontologies. For SSA a threshold of 0.25 or greater will result in best F-measure values, whereas the same approach returns best results in Multifarm for $t = 0.05$.

5.3 Further optimizations

Two search approaches are introduced, namely SSA and ITSA. Both have advantages as well as disadvantages. On the multifarm track the SSA performs better, but on the conference track ITSA can also provide good results. It is possible to use both approaches together to maybe get both advantages and less disadvantages.

Another improvement can be the change of comparing two concepts by only look at the overlapping articles to also look at the so called snippets. The API can return for every article of the search request also snippets where the search term occur. With this new information a mapping can for example be created when one search term occur in the snippets of the other term. Another option is to compute a TF-IDF score (Salton and Buckley, 1988) based on the article title and snippets. This can improve the recall because especially abbreviations appear in these snippets very often. For example *Universal Transverse Mercator* which has the abbreviation UTM appears in 11 of the first 20 snippets (see [44]).

5.4 Conclusion

WikiMatch as an element based matcher, with Wikipedia as an external resource, is introduced. It can handle monolingual as well as multilingual input ontologies through the help of language links. WikiMatch can handle all domains that are available on Wikipedia and is never outdated.

Some non trivial mappings that can be found with this approach are *ACM = Association for Computing Machinery* and *postleitzahl = plz* when assuming a threshold lower than $t = 0.27$. This approach can also handle spelling mistakes though the suggestion feature of the search engine. For example the mapping between *computer* and *compuer* can be found with an threshold of 0.93. The reason why this is not 1.0 is that articles for *compuer* are also taken into account.

On the other side *movie title* and *movie name* are synonyms, but the resulting articles differ. Hence, not every synonym have overlapping articles. The terms are used in slightly different contexts so that no mapping can be established.

In this thesis two different search approaches for ontology matching with Wikipedia are implemented and evaluated. It is shown that the SSA which has participated in the OAEI challenge can provide some non trivial mappings in monolingual as well as in multilingual matching problems. This can be a hint for future ontology matchers to find more correct mappings.

[44] `http://en.wikipedia.org/w/index.php?title=Special%3ASearch&profile=default&search=Universal+Transverse+Mercator&fulltext=Search`

References

Aguirre, J.-L. (2012a). Benchmark test library. `http://oaei.ontologymatching.org/2012/benchmarks/index.html`. Accessed: 01/11/2012.

Aguirre, J.-L. (2012b). Benchmark test library. `http://oaei.ontologymatching.org/2012/results/benchmarks/index.html`. Accessed: 01/11/2012.

Allemang, D. and Hendler, J. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann. Elsevier Science.

Berners-Lee, T. (2009). Semantic web and linked data. `http://www.w3.org/2009/Talks/0120-campus-party-tbl/`. Accessed: 20/08/2012.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). Web mining: information and pattern discovery on the world wide web. In *Scientific American 284 (2001), Nr. 5*, pages 34–43.

Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Research*, 32(Database-Issue):267–270.

Cardoso, J. (2007). The semantic web vision: Where are we? *Intelligent Systems, IEEE*, 22(5):84 –88.

Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., and Wilkinson, K. (2004). Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 74–83, New York, NY, USA. ACM.

Chakrabarti, S. (2002). *Mining the Web: Discovering Knowledge from Hypertext Data*. Number Teil 2 in The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann.

Chalupsky, H. (2000). Ontomorph: A translation system for symbolic knowledge. In Cohn, A. G., Giunchiglia, F., and Selman, B., editors, *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000)*, pages 471–482, Breckenridge (USA). Morgan Kaufmann Publishers.

Cheatham, M. (2011). Mapsss results for oaei 2011. In Shvaiko, P., Euzenat, J., Heath, T., Quix, C., Mao, M., and Cruz, I. F., editors, *OM*, volume 814 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Cilibrasi, R. L. and Vitanyi, P. M. (2007). The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19:370–383.

Cooley, R., Mobasher, B., and Srivastava, J. (1997). Web mining: information and pattern discovery on the world wide web. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, pages 558 –567.

Damerau, F. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Dang, T. T., Gabriel, A., Hertling, S., Roskosch, P., Wlotzka, M., Zilke, J. R., Janssen, F., and Paulheim, H. (2012). Hotmatch results for oeai 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*. To appear.

David, J., Euzenat, J., Scharffe, F., and Trojahn dos Santos, C. (2011). The alignment api 4.0. *Semantic Web – Interoperability, Usability, Applicability*, 2(1):3–10.

deBruijn, J., Ehrig, M., Feier, C., Martíns-Recuerda, F., Scharffe, F., and Weiten, M. (2006). Ontology mediation, merging, and aligning. In Davies, J., Studer, R., and Warren, P., editors, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. John Wiley & Sons.

Euzenat, J. (2004). An API for Ontology Alignment. In McIlraith, S. A., Plexousakis, D., and van Harmelen, F., editors, *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture Notes in Computer Science*, pages 698–712, Berlin, Heidelberg. Springer.

Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*. Springer-Verlag, Heidelberg (DE).

Ghidini, C., Serafini, L., and Tessaris, S. (2007). On relating heterogeneous elements from different ontologies. In Kokinov, B., Richardson, D., Roth-Berghofer, T., and Vieu, L., editors, *Proceedings of the Sixth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'07)*, volume 4635 of *Lecture Notes in Artificial Intelligence*, pages 234–247. Roskilde University, Denmark, Springer.

Hendler, J. (2009). Web 3.0 emerging. *Computer*, 42(1):111 –113.

Hertling, S. and Paulheim, H. (2012a). Wikimatch - using wikipedia for ontology matching. In *Seventh International Workshop on Ontology Matching (OM 2012)*.

Hertling, S. and Paulheim, H. (2012b). Wikimatch results for oeai 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*. To appear.

Hitzler, P., Krötzsch, M., Rudolph, S., and Sure, Y. (2008). *Semantic Web: Grundlagen*. eXamen.press. Springer, Berlin.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). Swrl: A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium.

Hu, W., Qu, Y., and Cheng, G. (2008). Matching large ontologies: A divide-and-conquer approach. *Data Knowl. Eng.*, 67(1):140–160.

Hughes, T. C. and Ashpole, B. C. (2004). The semantics of ontology alignment. In *I3CON. Information Interpretation and Integration Conference*.

Hustadt, U. (1994). Do we need the closed world assumption in knowledge representation? In Baader, F., Buchheit, M., Jeusfeld, M. A., and Nutt, W., editors, *KRDB*, volume 1 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579.

Jain, P., Hitzler, P., Sheth, A., Verma, K., and Yeh, P. (2010). Ontology alignment for linked open data. In *International Semantic Web Conference (ISWC 2010)*, pages 402–417. Springer.

Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008.

Jiménez-Ruiz, E. and Grau, B. C. (2011). Logmap: Logic-based and scalable ontology matching. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N. F., and Blomqvist, E., editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 273–288. Springer.

Kavšek, B., Lavrač, N., and Jovanoski, V. (2003). Apriori-sd: Adapting association rule learning to subgroup discovery. In R. Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R., and Borgelt, C., editors, *Advances in Intelligent Data Analysis V*, volume 2810 of *Lecture Notes in Computer Science*, pages 230–241. Springer Berlin / Heidelberg.

Kim, W. and Seo, J. (1991). Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, 24(12):12–18.

Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. In Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., and Uschold, M., editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle.

Klyne, G. and Carroll, J. J. (2004). Resource description framework (RDF): concepts and abstract syntax. W3C recommendation, World Wide Web Consortium.

Kosala, R. and Blockeel, H. (2000). Web mining research: a survey. *SIGKDD Explor. Newsl.*, 2(1):1–15.

Kotis, K., Katasonov, A., and Leino, J. (2012). Automsv2 results for oaei 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*. To appear.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707.

Lin, F. and Krizhanovsky, A. (2011). Multilingual Ontology Matching based on Wiktionary Data Accessible via SPARQL Endpoint. In *Russian Conference on Digital Libraries 2011 (RCDL'2011)*, pages 1–8.

Lin, F. and Sandkuhl, K. (2007). A new expanding tree ontology matching method. In Meersman, R., Tari, Z., and Herrero, P., editors, *OTM Workshops (2)*, volume 4806 of *Lecture Notes in Computer Science*, pages 1329–1337. Springer.

Lin, F. and Sandkuhl, K. (2008). A survey of exploiting wordnet in ontology matching. In Bramer, M., editor, *Artificial Intelligence in Theory and Practice II*, volume 276 of *IFIP International Federation for Information Processing*, pages 341–350. Springer Boston.

Maedche, A. and Staab, S. (2001). Ontology learning for the semantic web. *Intelligent Systems, IEEE*, 16(2):72 – 79.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press, New York.

Mayr, P. and Petras, V. (2008). Building a terminology network for search: the komohe project. *CoRR*, abs/0808.0518.

Meilicke, C. (2009). The relevance of reasoning and alignment incoherence in ontology matching. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., and Simperl, E. P. B., editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 934–938. Springer.

Meilicke, C., Castro, R. G., Freitas, F., van Hage, W. R., Montiel-Ponsoda, E., de Azevedo, R. R., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Tamilin, A., Trojahn, C., and Wang, S. (2012). MultiFarm: A Benchmark for Multilingual Ontology Matching. *Journal of Web Semantics*.

Meilicke, C. and Stuckenschmidt, H. (2007). Analyzing mapping extraction approaches. In Shvaiko, P., Euzenat, J., Giunchiglia, F., and He, B., editors, *Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) Collocated with the 6th International Semantic Web Conference (ISWC-2007)* *and the 2nd Asian Semantic Web Conference (ASWC-2007)*, volume 304 of *CEUR Workshop Proceedings*, Busan, Korea. CEUR-WS.org.

Miliard, M. (2008). Wikipediots: Who are these devoted, even obsessive contributors to wikipedia? `http://www.cityweekly.net/utah/article-5129-feature-wikipediots-who-are-these-devoted-even-` `html`. Accessed: 20/08/2012.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Nakayama, K., Hara, T., and Nishio, S. (2007). Wikipedia mining for an association web thesaurus construction. In Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., and Godart, C., editors, *WISE*, volume 4831 of *Lecture Notes in Computer Science*, pages 322–334. Springer.

Ngo, D., Bellahsene, Z., and Coletta, R. (2011). YAM++ – Results for OAEI 2011. In *Sixth International Workshop on Ontology Matching (OM 2011)*.

Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Paulheim, H. (2008). On applying matching tools to large-scale ontologies. In Shvaiko, P., Euzenat, J., Giunchiglia, F., and Stuckenschmidt, H., editors, *OM*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Paulheim, H. (2011a). Semantic web lecture slides. `http://www.ke.tu-darmstadt.de/lehre/ws-11-12/semantic-web/folien-teil-1`. Accessed: 20/08/2012.

Paulheim, H. (2011b). Semantic web lecture slides. `http://www.ke.tu-darmstadt.de/lehre/archiv/ws-11-12/semantic-web/folien-teil-12`. Accessed: 20/08/2012.

Paulheim, H. (2011c). Semantic web lecture slides. `http://www.ke.tu-darmstadt.de/lehre/archiv/ws-11-12/semantic-web/folien-teil-6`. Accessed: 20/08/2012.

Paulheim, H. (2011d). Semantic web lecture slides. `http://www.ke.tu-darmstadt.de/lehre/archiv/ws-11-12/semantic-web/folien-teil-9-ontology-matching`. Accessed: 20/08/2012.

Paulheim, H. (2012). Wesee-match results for oeai 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*. To appear.

Paulheim, H., Plendl, R., Probst, F., and Oberle, D. (2011). Mapping pragmatic class models to reference ontologies. In *Data Engineering Workshops (ICDEW), 2011 IEEE 27th International Conference on*, pages 200 –205.

Pesquita, C., Stroe, C., Cruz, I. F., and Couto, F. M. (2010). Blooms on agreementmaker: results for oaei 2010. In Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Mao, M., and Cruz, I. F., editors, *OM*, volume 689 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Quasthoff, M., Völkel, M., and Meinel, C. (2010). Unsupervised matching of object models and ontologies using canonical vocabulary. In Paschke, A., Henze, N., and Pellegrini, T., editors, *I-SEMANTICS*, ACM International Conference Proceeding Series. ACM.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, 2nd edition. Accessed: 20/08/2012.

Ritze, D., Völker, J., Meilicke, C., and Sváb-Zamazal, O. (2010). Linguistic analysis for complex ontology matching. In *OM*.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24:513–523.

Sasaki, Y. (2007). The truth of the f-measure. In *MIB -School of Computer Science, University of Manchester*.

Schadd, F. C. and Roos, N. (2012). Maasmatch results for oaei 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*. To appear.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423.

Sheth, A. P. and Kashyap, V. (1992). So far (schematically) yet so near (semantically). In Hsiao, D. K., Neuhold, E. J., and Sacks-Davis, R., editors, *DS-5*, volume A-25 of *IFIP Transactions*, pages 283–312. North-Holland.

Shvaiko, P. and Euzenat, J. (2005). A survey of schema-based matching approaches. In Spaccapietra, S., editor, *Journal on Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*, pages 146–171. Springer Berlin Heidelberg.

Shvaiko, P. and Euzenat, J. (2008). Ten challenges for ontology matching. In Meersman, R. and Tari, Z., editors, *OTM Conferences (2)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1164–1182. Springer.

Sirin, E. and Parsia, B. (2004). Pellet: An owl dl reasoner. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004*, volume 104 of *CEUR Workshop Proceedings*.

Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Svab, O., Svatek, V., Berka, P., Rak, D., and Tomasek, P. (2005). Ontofarm: Towards an experimental collection of parallel ontologies. In *Proceedings of the 5th International Semantic Web Conference ISWC-05*. Poster Track.

Thayasivam, U., Chaudhari, T., and Doshi, P. (2012). Optima+ results for oaei 2012. In *Seventh International Workshop on Ontology Matching (OM 2012)*. To appear.

Trojahn dos Santos, C. (2012). Multifarm results for oaei 2012. `http://oaei.ontologymatching.org/2012/multifarm/results/index.html`. Accessed: 01/11/2012.

Šváb Zamazal, O. (2012). Results of evaluation for the conference track within oaei 2012. `http://oaei.ontologymatching.org/2012/conference/eval.html`. Accessed: 01/11/2012.

Visser, P. R. S., Jones, D. M., Capon, B. T. J. M., and Shave, M. J. R. (1997). An analysis of ontological mismatches: Heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA.

Völkel, M. and Sure, Y. (2005). Rdfreactor – from ontologies to programmatic data access. In *Poster Proceedings of the Fourth International Semantic Web Conference*.

Winkler, W. E. (1999). The state of record linkage and current research problems. Technical Report Statistical Research Report Series RR99/04, U.S. Bureau of the Census, Washington, D.C.

Zhang, H., Hu, W., and Qu, Y. (2011). Constructing virtual documents for ontology matching using mapreduce. In Pan, J. Z., Chen, H., Kim, H.-G., Li, J., Wu, Z., Horrocks, I., Mizoguchi, R., and Wu, Z., editors, *JIST*, volume 7185 of *Lecture Notes in Computer Science*, pages 48–63. Springer.

Search major language tags

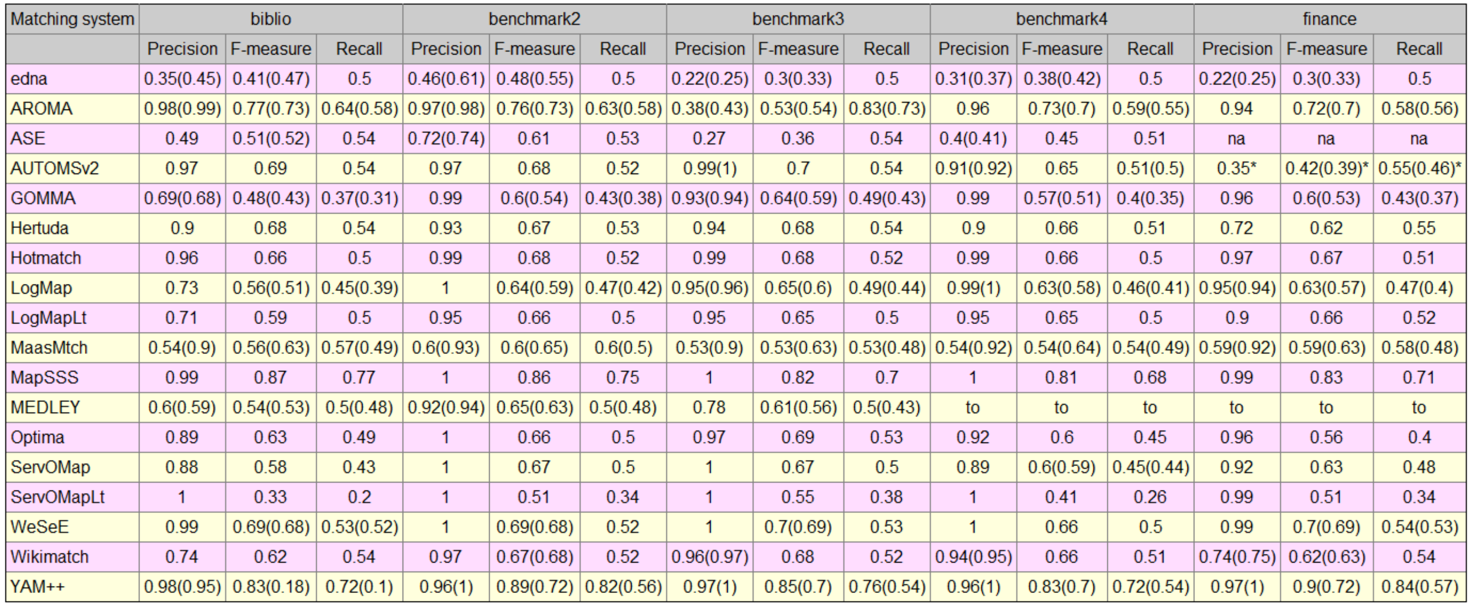
```java
public String getDefaultLanguage(OntModel m) {

        Map<String,Integer> map = new HashMap<String,Integer>();
        ResIterator it = m.listResourcesWithProperty(RDFS.label);

        while(it.hasNext()) {
            StmtIterator labels = it.next().listProperties(RDFS.label);
            while(labels.hasNext()) {
                String language = labels.next().getLiteral().getLanguage();
                if(language.length() > 0){
                  Integer count = map.get(language);
          map.put(language, count != null ? count+1 : 1);
                }
            }
        }

        // maximum
        String languageMax = new String("en");
        int countMax = 0;
        for(Entry<String,Integer> entry : map.entrySet()) {
            if(entry.getValue() > countMax) {
              languageMax = entry.getKey();
              countMax = entry.getValue();
            }
        }

        return languageMax;
}
```

Listing 2: Extract major language tag from an ontology in java

Answer of Wikipedia in JSON

```
{"query":
  {"search":[
    {"ns":0,"title":"Paper"},
    {"ns":0,"title":"Pap\u00ebr"},
    {"ns":0,"title":"Paper mill"},
    {"ns":0,"title":"Newspaper"},
    {"ns":0,"title":"Postage stamp paper"},
    {"ns":0,"title":"Rock-paper-scissors"},
    {"ns":0,"title":"Banknote"},
    {"ns":0,"title":"Toilet paper"},
    {"ns":0,"title":"Pulp and paper industry"},
    {"ns":0,"title":"Paper recycling"},
    {"ns":0,"title":"Papermaking"},
    {"ns":0,"title":"Pulp (paper)"},
    {"ns":0,"title":"Rzeczpospolita (newspaper)"},
    {"ns":0,"title":"Paper prototyping"},
    {"ns":0,"title":"Paper negative"},
    {"ns":0,"title":"Electrical insulation paper"},
    {"ns":0,"title":"Cartridge paper"},
    {"ns":0,"title":"H.I.V.E."},
    {"ns":0,"title":"Paper data storage"},
    {"ns":0,"title":"Paper (Queen Latifah song)"},
    {"ns":0,"title":"Form (document)"},
    {"ns":0,"title":"Paper (Krayzie Bone song)"},
    {"ns":0,"title":"Paper (album)"}
  ]},
  "query-continue":{
    "search":{
      "sroffset":50
    }
  }
}
```

Listing 3: Wikipedia answer for searching *paper* (extract)



Matching system	biblio			benchmark2			benchmark3			benchmark4			finance		
	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall	Precision	F-measure	Recall
edna	0.35(0.45)	0.41(0.47)	0.5	0.46(0.61)	0.48(0.55)	0.5	0.22(0.25)	0.3(0.33)	0.5	0.31(0.37)	0.38(0.42)	0.5	0.22(0.25)	0.3(0.33)	0.5
AROMA	0.98(0.99)	0.77(0.73)	0.64(0.58)	0.97(0.98)	0.76(0.73)	0.63(0.58)	0.38(0.43)	0.53(0.54)	0.83(0.73)	0.96	0.73(0.7)	0.59(0.55)	0.94	0.72(0.7)	0.58(0.56)
ASE	0.49	0.51(0.52)	0.54	0.72(0.74)	0.61	0.53	0.27	0.36	0.54	0.4(0.41)	0.45	0.51	na	na	na
AUTOMSv2	0.97	0.69	0.54	0.97	0.68	0.52	0.99(1)	0.7	0.54	0.91(0.92)	0.65	0.51(0.5)	0.35*	0.42(0.39)*	0.55(0.46)*
GOMMA	0.69(0.68)	0.48(0.43)	0.37(0.31)	0.99	0.6(0.54)	0.43(0.38)	0.93(0.94)	0.64(0.59)	0.49(0.43)	0.99	0.57(0.51)	0.4(0.35)	0.96	0.6(0.53)	0.43(0.37)
Hertuda	0.9	0.68	0.54	0.93	0.67	0.53	0.94	0.68	0.54	0.9	0.66	0.51	0.72	0.62	0.55
Hotmatch	0.96	0.66	0.5	0.99	0.68	0.52	0.99	0.68	0.52	0.99	0.66	0.5	0.97	0.67	0.51
LogMap	0.73	0.56(0.51)	0.45(0.39)	1	0.64(0.59)	0.47(0.42)	0.95(0.96)	0.65(0.6)	0.49(0.44)	0.99(1)	0.63(0.58)	0.46(0.41)	0.95(0.94)	0.63(0.57)	0.47(0.4)
LogMapLt	0.71	0.59	0.5	0.95	0.66	0.5	0.95	0.65	0.5	0.95	0.65	0.5	0.9	0.66	0.52
MaasMtch	0.54(0.9)	0.56(0.63)	0.57(0.49)	0.6(0.93)	0.6(0.65)	0.6(0.5)	0.53(0.9)	0.53(0.63)	0.53(0.48)	0.54(0.92)	0.54(0.64)	0.54(0.49)	0.59(0.92)	0.59(0.63)	0.58(0.48)
MapSSS	0.99	0.87	0.77	1	0.86	0.75	1	0.82	0.7	1	0.81	0.68	0.99	0.83	0.71
MEDLEY	0.6(0.59)	0.54(0.53)	0.5(0.48)	0.92(0.94)	0.65(0.63)	0.5(0.48)	0.78	0.61(0.56)	0.5(0.43)	to	to	to	to	to	to
Optima	0.89	0.63	0.49	1	0.66	0.5	0.97	0.69	0.53	0.92	0.6	0.45	0.96	0.56	0.4
ServOMap	0.88	0.58	0.43	1	0.67	0.5	1	0.67	0.5	0.89	0.6(0.59)	0.45(0.44)	0.92	0.63	0.48
ServOMapLt	1	0.33	0.2	1	0.51	0.34	1	0.55	0.38	1	0.41	0.26	0.99	0.51	0.34
WeSeE	0.99	0.69(0.68)	0.53(0.52)	1	0.69(0.68)	0.52	1	0.7(0.69)	0.53	1	0.66	0.5	0.99	0.7(0.69)	0.54(0.53)
Wikimatch	0.74	0.62	0.54	0.97	0.67(0.68)	0.52	0.96(0.97)	0.68	0.52	0.94(0.95)	0.66	0.51	0.74(0.75)	0.62(0.63)	0.54
YAM++	0.98(0.95)	0.83(0.18)	0.72(0.1)	0.96(1)	0.89(0.72)	0.82(0.56)	0.97(1)	0.85(0.7)	0.76(0.54)	0.96(1)	0.83(0.7)	0.72(0.54)	0.97(1)	0.9(0.72)	0.84(0.57)

Figure 34: Detailed benchmark results from (Aguirre, 2012b))

algo	AUTOMSv2			CODI			GOMMA			Hertuda			HotMatch			LogMap			LogMapLt			MaasMtch			MapSSS			MEDLEY			Optima			WeSeE			Wmatch			YAM		
test	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R
cn-cz	NaN	NaN	0.00	0.00	NaN	0.00	0.38	0.34	0.31	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	0.02	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.50	0.29	0.21	0.35	0.21	0.15	0.40	0.35	0.31
cn-de	NaN	NaN	0.00	0.00	NaN	0.00	0.42	0.34	0.29	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	0.02	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.70	0.25	0.15	0.13	0.08	0.06	0.40	0.35	0.30
cn-en	NaN	NaN	0.00	0.00	NaN	0.00	0.59	0.41	0.32	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.01	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.56	0.31	0.22	0.15	0.10	0.07	0.48	0.41	0.36
cn-es	NaN	NaN	0.00	0.00	NaN	0.00	0.34	0.33	0.31	0.00	0.01	1.00	0.01	0.01	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.02	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.53	0.34	0.25	0.19	0.14	0.11	0.41	0.20	0.13
cn-fr	NaN	NaN	0.00	0.00	NaN	0.00	0.38	0.35	0.33	0.00	0.01	1.00	0.03	0.01	0.01	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.01	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.52	0.32	0.23	0.14	0.09	0.07	0.44	0.39	0.35
cn-nl	NaN	NaN	0.00	0.00	NaN	0.00	0.23	0.25	0.27	0.00	0.01	1.00	0.01	0.01	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.01	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.42	0.23	0.16	0.17	0.10	0.07	0.38	0.34	0.30
cn-pt	NaN	NaN	0.00	0.00	NaN	0.00	0.37	0.31	0.27	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.01	0.01	0.03	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.53	0.30	0.21	0.14	0.09	0.07	0.40	0.35	0.31
cn-ru	NaN	NaN	0.00	0.00	NaN	0.00	0.29	0.27	0.25	0.00	0.01	1.00	0.01	0.01	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.01	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.45	0.27	0.20	0.22	0.13	0.09	0.41	0.33	0.27
cz-de	NaN	NaN	0.00	0.39	0.10	0.06	0.17	0.24	0.41	0.00	0.01	1.00	0.00	NaN	0.00	0.39	0.10	0.06	0.30	0.09	0.05	0.04	0.06	0.22	0.12	0.07	0.05	0.43	0.19	0.12	0.00	0.01	1.00	0.82	0.41	0.27	0.27	0.24	0.22	0.50	0.45	0.41
cz-en	NaN	NaN	0.00	0.31	0.07	0.04	0.36	0.36	0.37	0.00	0.01	1.00	0.00	NaN	0.00	0.25	0.05	0.03	0.18	0.04	0.02	0.04	0.06	0.23	0.12	0.08	0.06	0.32	0.28	0.26	0.00	0.01	1.00	0.68	0.48	0.37	0.28	0.24	0.22	0.58	0.58	0.57
cz-es	NaN	NaN	0.00	0.44	0.11	0.07	0.22	0.30	0.48	0.00	0.01	1.00	0.00	NaN	0.00	0.44	0.11	0.07	0.36	0.11	0.07	0.03	0.06	0.21	0.17	0.11	0.08	0.33	0.13	0.08	0.00	0.01	1.00	0.56	0.47	0.41	0.26	0.25	0.24	0.56	0.20	0.12
cz-fr	NaN	NaN	0.00	0.09	0.01	0.01	0.10	0.16	0.39	0.00	0.01	1.00	0.03	0.01	0.01	0.09	0.01	0.01	0.07	0.01	0.01	0.02	0.04	0.14	0.02	0.01	0.01	0.20	0.08	0.05	0.00	0.01	1.00	0.61	0.47	0.38	0.25	0.20	0.17	0.54	0.53	0.52
cz-nl	NaN	NaN	0.00	0.38	0.09	0.05	0.14	0.21	0.48	0.00	0.01	1.00	0.00	NaN	0.00	0.19	0.04	0.02	0.15	0.04	0.02	0.04	0.07	0.24	0.09	0.05	0.04	0.21	0.09	0.06	0.00	0.01	1.00	0.65	0.48	0.39	0.24	0.21	0.18	0.56	0.55	0.53
cz-pt	NaN	NaN	0.00	0.52	0.15	0.09	0.31	0.37	0.45	0.00	0.01	1.00	0.00	NaN	0.00	0.47	0.13	0.07	0.40	0.13	0.08	0.03	0.06	0.22	0.19	0.12	0.09	0.41	0.18	0.11	0.00	0.01	1.00	0.60	0.44	0.35	0.12	0.12	0.13	0.57	0.57	0.57
cz-ru	NaN	NaN	0.00	0.00	NaN	0.00	0.17	0.21	0.27	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.01	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.61	0.47	0.38	0.18	0.17	0.16	0.52	0.49	0.47
de-en	0.81	0.38	0.25	0.60	0.20	0.12	0.43	0.41	0.39	0.00	0.01	1.00	0.00	NaN	0.00	0.60	0.22	0.13	0.52	0.20	0.12	0.04	0.06	0.23	0.22	0.16	0.13	0.34	0.27	0.22	0.00	0.01	0.76	0.69	0.39	0.28	0.26	0.28	0.30	0.56	0.52	0.48
de-es	0.72	0.35	0.24	0.29	0.06	0.03	0.30	0.35	0.43	0.00	0.01	1.00	0.00	NaN	0.00	0.46	0.12	0.07	0.22	0.06	0.03	0.04	0.06	0.23	0.22	0.15	0.12	0.32	0.15	0.10	0.00	0.01	1.00	0.65	0.41	0.30	0.21	0.24	0.27	0.56	0.20	0.12
de-fr	0.91	0.32	0.20	0.20	0.04	0.02	0.15	0.21	0.37	0.00	0.01	1.00	0.00	NaN	0.00	0.20	0.04	0.02	0.15	0.04	0.02	0.03	0.05	0.18	0.19	0.13	0.10	0.31	0.13	0.09	0.00	0.01	1.00	0.81	0.41	0.27	0.24	0.25	0.26	0.50	0.46	0.43
de-nl	0.81	0.39	0.26	0.23	0.05	0.03	0.17	0.24	0.43	0.00	0.01	1.00	0.00	NaN	0.00	0.21	0.04	0.02	0.17	0.04	0.02	0.04	0.06	0.23	0.21	0.15	0.12	0.24	0.12	0.08	0.00	0.01	1.00	0.66	0.37	0.26	0.27	0.25	0.24	0.44	0.40	0.36
de-pt	0.83	0.35	0.22	0.35	0.08	0.04	0.34	0.36	0.39	0.00	0.01	1.00	0.00	NaN	0.00	0.33	0.07	0.04	0.26	0.07	0.04	0.03	0.05	0.18	0.10	0.06	0.04	0.33	0.15	0.10	0.00	0.01	1.00	0.65	0.35	0.24	0.20	0.22	0.24	0.45	0.42	0.39
de-ru	NaN	NaN	0.00	0.00	NaN	0.00	0.35	0.33	0.31	0.00	0.01	1.00	0.01	0.01	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.01	0.01	0.04	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.73	0.42	0.29	0.27	0.26	0.26	0.53	0.47	0.43
en-es	0.60	0.42	0.33	0.23	0.04	0.02	0.44	0.40	0.37	0.00	0.01	1.00	0.00	NaN	0.00	0.50	0.15	0.09	0.18	0.04	0.02	0.05	0.08	0.28	0.24	0.18	0.15	0.28	0.28	0.27	0.00	0.01	1.00	0.65	0.52	0.44	0.24	0.30	0.40	0.58	0.23	0.15
en-fr	0.56	0.31	0.22	0.21	0.04	0.02	0.33	0.36	0.41	0.00	0.01	1.00	0.01	0.01	0.00	0.28	0.06	0.04	0.14	0.04	0.02	0.05	0.09	0.33	0.19	0.13	0.10	0.34	0.33	0.33	0.00	0.01	1.00	0.69	0.48	0.37	0.25	0.27	0.29	0.56	0.53	0.51
en-nl	0.64	0.39	0.28	0.36	0.10	0.06	0.36	0.38	0.41	0.00	0.01	1.00	0.00	NaN	0.00	0.29	0.08	0.04	0.30	0.10	0.06	0.05	0.09	0.31	0.21	0.15	0.12	0.27	0.24	0.21	0.00	0.01	1.00	0.65	0.49	0.40	0.28	0.29	0.30	0.56	0.53	0.51
en-pt	0.62	0.37	0.26	0.35	0.08	0.04	0.53	0.45	0.39	0.00	0.01	1.00	0.00	NaN	0.00	0.28	0.06	0.03	0.24	0.06	0.04	0.04	0.06	0.23	0.11	0.07	0.05	0.30	0.30	0.29	0.00	0.01	1.00	0.63	0.51	0.43	0.20	0.26	0.35	0.58	0.56	0.54
en-ru	0.00	NaN	0.00	0.00	NaN	0.00	0.45	0.34	0.27	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.01	0.01	0.04	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.61	0.43	0.33	0.22	0.25	0.29	0.51	0.47	0.44
es-fr	0.61	0.37	0.27	0.09	0.01	0.01	0.21	0.29	0.50	0.00	0.01	1.00	0.00	NaN	0.00	0.31	0.07	0.04	0.07	0.01	0.01	0.05	0.08	0.29	0.09	0.06	0.04	0.15	0.06	0.04	0.00	0.01	1.00	0.63	0.52	0.44	0.20	0.24	0.31	0.57	0.20	0.12
es-nl	0.53	0.38	0.29	0.00	NaN	0.00	0.25	0.33	0.51	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.03	0.05	0.17	0.02	0.01	0.01	0.07	0.03	0.02	0.00	0.01	1.00	0.56	0.46	0.40	0.25	0.27	0.30	0.49	0.16	0.10
es-pt	0.56	0.44	0.37	0.47	0.22	0.15	0.37	0.44	0.54	0.00	0.01	1.00	0.00	NaN	0.00	0.53	0.24	0.16	0.47	0.23	0.15	0.06	0.11	0.40	0.31	0.23	0.19	0.37	0.22	0.16	0.00	0.01	1.00	0.58	0.51	0.45	0.18	0.25	0.41	0.59	0.25	0.16
es-ru	0.00	NaN	0.00	0.00	NaN	0.00	0.21	0.21	0.20	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.57	0.47	0.40	0.25	0.28	0.31	0.55	0.19	0.11
fr-nl	0.51	0.27	0.19	0.45	0.13	0.07	0.14	0.21	0.44	0.00	0.01	1.00	0.01	0.01	0.00	0.44	0.13	0.07	0.38	0.12	0.07	0.04	0.07	0.24	0.17	0.11	0.09	0.32	0.16	0.11	0.00	0.01	1.00	0.59	0.43	0.34	0.25	0.28	0.31	0.48	0.47	0.47
fr-pt	0.62	0.35	0.24	0.00	NaN	0.00	0.26	0.32	0.43	0.00	0.01	1.00	0.01	0.01	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.03	0.06	0.21	0.03	0.02	0.01	0.17	0.08	0.05	0.00	0.01	1.00	0.65	0.50	0.41	0.18	0.23	0.31	0.53	0.53	0.54
fr-ru	0.00	NaN	0.00	0.00	NaN	0.00	0.21	0.24	0.30	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.02	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.62	0.47	0.37	0.26	0.28	0.30	0.48	0.46	0.44
nl-pt	0.59	0.37	0.27	0.19	0.04	0.02	0.26	0.32	0.43	0.00	0.01	1.00	0.00	NaN	0.00	0.09	0.01	0.01	0.07	0.01	0.01	0.03	0.05	0.18	0.04	0.02	0.02	0.12	0.06	0.04	0.00	0.01	1.00	0.62	0.47	0.38	0.16	0.20	0.27	0.54	0.51	0.49
nl-ru	0.00	NaN	0.00	0.00	NaN	0.00	0.17	0.22	0.28	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	0.02	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	0.43	0.55	0.42	0.34	0.31	0.31	0.30	0.44	0.42	0.40
pt-ru	0.00	NaN	0.00	0.00	NaN	0.00	0.38	0.30	0.24	0.00	0.01	1.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.00	0.02	0.00	NaN	0.00	0.00	NaN	0.00	0.00	0.01	1.00	0.60	0.45	0.36	0.15	0.17	0.20	0.47	0.44	0.42

Figure 35: Multifarm results split up into language pairs (see (Trojahn dos Santos, 2012))