Ontology Matching with word2vec

Harmen Prins

March 15, 2016

Summary

Contents

1 Introduction

It is a Friday afternoon and you realise you want to watch a movie with some friends. So, what do you do? To achieve this goal you must take 4 steps.

Firstly, you must open up an internet Movie Database, search for movies that you like that are also playing this weekend. To do this you must either manually go through the list of movies playing this weekend or movies the Database recommends for you and check if any movies on that list match your full criteria.

Once you have found one or more movies that match your search criteria, you have to select a time at which you can see the candidate movies, so you check for every movie every available time slot at every cinema near you. The number of time slots is equal to the number of movies you chose in the previous step, times the number of cinemas near you times the number of times every movie is shown at one cinema.

The next step is to filter the number of time slots based on your agenda, so you cross-reference the available movie times with the times you are available during the weekend.

Then onto the last step, getting friends to go with you. In this step you do not want to burden your friends to much by having them pick both the movie and the time, so you will have to decide which movie to watch or when, even though you do not know if your friends will like this movie or are available at that time. Once you contact your friends, most of them do not respond as they are not online at that time or they do not feel like filling in a poll about availability or movie choices.

What would happen if this process was automated? The first step, finding movies that match two criteria would be considered trivial with modern technology. One finds the two lists of movies that are recommended for you and movies that are playing this weekend and intersect them. A ranking can be added based on how expensive the tickets are, how certain it is that you may like the movie, and other factors.

The second step, cross-referencing the list of movies with cinema play times, would already be harder. Most of the time the cinema play times are written in a human-readable format, something that is often hard to incorporate for a computer. Either the play times have to be made in a computer-readable format or a computer has to interpret them and convert them into computer-readable. However, errors might occur because the local cinema labelled a sequel as ”Movie II” whereas the list we have has it labelled as ”Movie 2”.

Checking an agenda to find an appropriate time would be easy if the times from the cinemas can be transformed to the same format as your agenda. If we are automating the process, we may be able to add more features, like changing standing appointments (of course incurring a penalty to the score of that time spot) and such.

The last step, contacting friends can be done in one click. Let us call the program that automated the first three steps a *softbot* and assume they are universal, i.e. everyone has one. Rather than bothering your friends with scheduling, your softbot can contact those of your friends and find the movie and time slot that optimize a cost function, taking into account which friends you like the most, which movie everybody likes the most and at which time the fewest dinners with wives need to be moved. Once this best time is chosen, everyone gets an invite to see the movie and once it is known who will go, a car pooling route is calculated and you can sit back an enjoy your Friday afternoon with no planning or scheduling required.

The most amazing thing about this story, is that it is already possible with the current technology. The algorithms to convert for example cinema website text to computer-readable format exist. Many open databases already contain a lot of structured data that can be used to reason about data.

Two things are required before this web of computers who communicate and understand, this Semantic Web, will come into existence. Firstly, the Semantic Web needs to be adopted by humans. They must see the value of these personal assistants that can schedule things for you, that take into account your personal context when searching the web, that understand questions and can ask you for more information that can improve the search. People have to realise that the current search engines are not good enough, and that having to read through many papers to find the one nugget of information or the one connection that you are looking for is not acceptable or necessary. Lastly, the different knowledge bases and unstructured texts need to be aligned. This means that softbots should be able to reason about facts that are contained in different domains and documents. This last problem is the one that I will work on in this thesis. Once we can combine the information from different databases and unstructured text, the Semantic Web is technically possible and only needs to be socially accepted before it will become the norm.

1.1 Short Context

On the Semantic Web, ontologies are used to define and share knowledge. The term ontology will be properly defined later on, but for now it suffices to know that an ontology is a set of rules that determine what relations are allowed and required. On the semantic web, ontologies are the foundation for communication between agents.

However, on the Semantic Web it is infeasible to use one single ontology, since the number of concepts it would have to contain is incredibly large. The larger an ontology the longer it takes to perform inference and searches on it, and thus the longer it takes to use it for communication. Therefore the semantic web will have separate ontologies for every domain. The downside of this approach is that when communication requires rules on multiple domains, which it often does, multiple ontologies need to be aligned or merged.

1.2 Overview

This thesis is structured as follows.

2 Context

In this section I will discuss all concepts that are required to understand the study.

2.1 The Semantic Web

The Semantic Web, or Web 3.0, is a concept of a Web that enhances the User Experience in ways that the current Web can not provide. Web 2.0, the current Web, models the internet as *hubs* (or sites) with a specific purpose and barely any interconnectivity. For example, Facebook is a hub for sharing content with people you consider friends, but if you want to share that content with more people you will have to go to another hub, like imgur or Pinterest. The Semantic Web, on the other hand, sees web pages as sources of *information* that can be combined whenever the user needs it. This allows not only humans to browse the internet, but also Artificial Intelligences, which will use this information to aid their users.

For example, if a user wants to see a movie this weekend, he orders his AI, or *softbot*, to find him a suitable time and movie to see. The softbot then accesses the Internet Movie Database to see which movies currently play and which of those are similar to what the user liked in the past. He will then access the pages of the local movie theaters to see when those movies play. Lastly, he will contact the softbots of the friends of the user to ask if they will also come to see that movie. After a short exchange the optimal time, place and participant set are decided, and the softbot tells the user what the results are. When the user accepts, the appointment is automatically added to his agenda.

All of this is possible, if the softbot can access all of the information reliably. In the current Web, this is not possible, as all information is written in an ambiguous, unstructured format called human language. To allow softbots to access the same information that we can, the Semantic Web proposes to add a layer to the internet where all information is saved in a unified format that is unambiguous, robot-interpretable and can be used to store any type of knowledge.

2.1.1 Goal

The goal of the Semantic Web is very pragmatic: help people in everyday activities by leveraging all information available on the internet. The fact that it is pragmatic goes a long way of making it a reality: if even one application comes into existence that uses online information in a structured way and thus makes everyday tasks a little easier, the goal is accomplished. Of course, then the goal resets and we should make more applications leveraging more information for more tasks.

Ultimately, the goal is to leverage all information that is available to an entity on the internet to improve the lives of humans. This information is not limited to the information that is online *now*, but can include sensor data from the Internet of Things, data from robots, facts deduced or statistically inferred from existing data and so on. All this information can be used to accommodate the wants and needs of humans, when asked for it but also before the users are even aware of their needs. The personal agent should be able to predict the users need and provide the tools that can satisfy the need.

To provide these tools to the user, an agent uses many different sources. For example, if the need of a researcher is knowledge on a certain topic, the agent must chart relevant topics, map relations between different domains and connect these new ideas to topics the researcher already knows about. In fact, the largest part of the job an agent has to carry out on the Semantic Web is finding relationships between pieces of information on the Web. Based on this observation, we can state that Ontology and Knowledge Base Alignment is the cornerstone of the Semantic Web, since without this vital system the Web is merely a collection of separate pieces of information. Only when those pieces are aligned can we talk of knowledge, of semantics.

To summarize, the goal of the Semantic Web is to connect all online information to help humans. Only when all information can be connected will it be possible to aid humans in their needs. But humans will only accept the Semantic Web if it is useful to them, so the Semantic Web must show its applications and usefulness before it will be used in daily life.

2.1.2 Current state

The Semantic Web is currently divided into two groups: the people who need high accuracy and the people who need high recall. The first group consist of scientists that work on specific fields that require exact knowledge, whereas the second group are data scientists that want big amounts of data available.

The group that requires high recall use hand-crafted knowledge bases, created by experts, containing only facts that are accurate, verified and relevant. This method of creating a knowledge base does not scale very well and usually consists of fewer than ten thousand facts. Because of this, the knowledge bases are restricted to small domains, including parts of medicine and cultural heritage.

The group that requires a high recall, i.e. requires a huge amount of data, automated methods are required. These methods work fast, parsing thousands of websites a minute, but are inclined to make mistakes. Often, these methods use crowd-sourcing to improve accuracy, using experts or the general public to suggest or check facts.

Beside these knowledge bases, many other pieces of the Semantic Web are already in use. For example, the W3C

Large ontologies already exist, with the largest, DBPedia, boasting almost two billion triples and the three hundred combined Large Open Data databases contain over thirty billion.

Leading scientists suggest that more research should be focused on

Applications include

Overall it is clear that

2.1.3 Technologies

2.2 Ontologies

2.2.1 OWL

2.2.2 Role

2.3 Current alignment strategies

Many different alignment strategies have already been developed. All strategies follow the same two-step approach. The two steps are independent and as such different methods can be used interchangeably. The first step is to generate an initial correspondence set, where correspondences between nodes are found based on just their labels and meta-data. The second step is to use this initial set and the structure of the ontology to find more correspondences. These steps are called the terminological and the structural steps. Some strategies also use extra steps like the extensional and semantic steps, which use vector space models and inference, respectively. https://hal.inria.fr/hal-00917910/document

Popular terminological strategies are WordNet comparison and edit distance. The former uses the popular WordNet hierarchy as a distance measure between to concepts, e.g. the number of edges between the concepts and their least general common superconcept. Edit distance is a purely string-based similarity measure. The similarity is a (weighted) count of edits required to transform one word into another, where common edits are insertion, deletion and replacement. Similar methods include substring matching and n-gram matching, which compare parts of the strings to find similarities.

2.4 Problems in ontology alignment

2.5 Other related research

2.5.1 Word sense representation

Skip-gram models have been widely used to represent words as vectors. This is useful as it is easy to calculate the distance between two vectors, which is a useful property for ontology matching. This vector distance has been shown to be related to the semantic similarity between words. This semantic similarity is very useful for ontology alignment, as parts of one ontology have to be aligned with parts of another ontology depending on their semantic relation.

Recently, an effective and efficient skip-gram model has been developed. It is called word2vec, and uses a number of extensions over previous methods that enable it to efficiently learn an effective representation. Word2vec learns this representation by trying to predict a context matrix. First a one-hot encoding is made for the vocabulary. Then the model learns to predict a context matrix of C one-hot vectors based on the given one-hot input vector. The model is a neural network with one hidden layer and multiple output layers. Since the words are one-hot encoded the input to the hidden layer is equal to the row of the weight matrix that corresponds to the word. This row is the vector representation of the word.

Word2vec has been expanded in a number of different ways. For example, word order can be preserved, leading to a similarity measure that is closer to syntax, as syntax defines which words go where in a sentence. This research shows that the input does not have to be a bag-of-words representation, which will be useful later on. Another interesting development is the multi-sense word2vec, which has multiple vector representations per word, depending on the number of different definitions a word has. For example, the word bank represents both the monetary institute and the riverside, which would both have a different representation in this extension. It can differentiate between two different meanings based on the context. This will help with resolving ambiguity, a very important factor in ontology alignment.

2.6 Used technology

2.6.1 Ontology constructor

2.6.2 AgreementMaker

AgreementMaker is an ontology matching system which obtained the highest F-measure in 6 of the 7 ontology matching tracks of OAEI 2015

The system uses an extensible framework which allows its users to add new modules to the system. This is very useful for research as it allows the researcher to compare different modules by swapping in just those modules in the framework. The module-swapping technique will also be used in this work, but more of that will be discussed in $Evaluationmethods$.

3 Problem

3.1 Problem

We have seen that when information from multiple domains need to be integrated so that it is possible to reason over them together, the ontologies of those domains need to be aligned or merged. Those ontologies contain rules about concepts and concept classes in the shape of triplets, with two concepts and a relation. For example, the rule`Professors are humans` contains the concepts *Professor* and *human* and relationship *being.*

If we want to merge two ontologies, we need to find the concepts from the different ontologies that are related by an 'is-same-as' or other rule.

However, since the ontologies are made for different domains, the same concept can be represented by different labels in the two ontologies. This problem is called the synonymy problem, as synonyms are two words that refer to the same concepts.

The opposite problem is called the homonymy problem, in which two concepts are represented by the same label. Since they have the same label, naive algorithms might align the concepts as being the same, when they are not.

Then there is the problem of ambiguity, which encompasses many other problems. It encompasses the fact that some concepts are used wrongly by humans, but also the fact that some concepts are extremely similar, and may occur in the same context, but are still slightly different.

Lastly, there is the problem of different types of alignment relations. For example, one concept can be a *part of* another concept, or an example of one. These different relations all require a different approach and come with their own sets of problems.

In summary, ontology aligning is required for multi-domain communication but poses a number of problems that make it difficult for concepts, the building blocks of ontologies, to be matched.

3.2 Research questions

3.3 Hypotheses

3.4 Justification

Reliable, error proof ontology matching will be key for the distributed Semantic Web.

4 Method

In this section, I will describe the method that I developed. Firstly, I will describe the general idea. In the second section, I will give a theoretical justification for why this method should work and how it works conceptually. After that I will describe the different ways that the input graphs can be used to train the word2vec model and create the alignment.

4.1 Idea

As explained earlier, word2vec is a model that learns two things: implicitly it learns the semantic vector representation of words, and explicitly it learns to predict words from their context or vice versa. I will use both of these pieces of information in our alignment algorithm. The semantic representation will be used for finding nodes in the two ontology graphs that are similar, i.e. close in the vector space. This step is relatively straightforward if proper vector representations are found. However, this may be hard due to a number of problems which I have listed below.

- The problem I am trying to solve is the problem of differently labelled nodes referring to the same concept. For example *writer* and *author* will not be matched by a string matcher, but should be matched. The two concepts may also have different labels in their context even though their contexts refer to the same concepts.

- The size of the training corpus should be large enough for proper representation to be learned for every concept. It should also be relevant to the labels that are in the ontologies. This is a problem since the ontologies may be relatively small, so they need to be extended while keeping relevance.

- Some labels may be ambiguous, for example homonyms, which are concepts that have the same label but represent a different context.

To solve these problems I will adapt word2vec to graphs and extend it with multi-sense embeddings. Also hot-starting is considered as an improvement.

4.2 Theory

4.3 Possibilities

4.3.1 Training corpus

Word embedding models need to be trained on a large corpus. These corpora need to cover the concepts that are contained in the ontology, but also need to be large enough to build good embeddings. Since the ontologies themselves do not necessarily contain enough examples to embed the concepts properly, I need to use ontologies that are likely to contain the concepts I want to align. As a benchmark, I will use the WordNet and NELL ontologies. To check if adding the ontologies that are to be aligned helps, I will also train a model on an ontology consisting of WordNet, NELL and the two alignment candidates.

4.3.2 Node vectorisation

The most basic node2vec model converts a node to a vector purely based on its own label. This method has one advantage over simple string matching: if no node is found in the other ontology that matches exactly, we can still find a node that is similar since it is close in the vector space. Therefore, this method should already be an improvement over the most basic string matching algorithms. It may even be competitive with more advanced string matchers (that look at substrings of labels). For example, *ear lobe* and *ear* will be matched by a substring matcher, but are also semantically similar since they will often be mentioned in the same context.

4.3.3 Combine with neighbours

To improve the model, we can add context information from the neighbours of a node. The most basic context adaptation would be to create a vector as the average of the context vectors. The resulting vector should be combined with the node vector of the node that is being investigated. This can be done by averaging or weighted averaging where the context is weighed more if there are more neighbours, though not necessarily linearly. These possibilities were also investigated.

4.3.4 Select from senses

Another way to take into account the influence of neighbouring nodes is to train a multi-sense node embedding model as described in appropriate section, and selecting from the different node senses based on which is closest to the context average.

4.3.5 Adding edge labels

To add more information to the model, edge labels can also be added to the context of a node. This method effectively doubles the training data and context size when creating the context average. This should make the model more robust, although edge labels may be duplicate (one has many unique family members) and less informative in general, so a lower weight may be appropriate. Another problem with this method is that the relation between a neighbour node and its corresponding edge is lost, since they are just treated as independent contexts.

4.3.6 Select from senses

This method is exactly the same as the selecting from node senses with just the neighbour nodes except that you now also take into account the edge labels.

4.3.7 Combining edges with nodes

As can be read in appropriate section, it is possible to drop the bag-of-words assumption that the context is sequence invariant. This means we can for example have the context be (previous word, next word) and those words will be treated differently. Similarly we can separate the edge and node labels and treat them differently. This ensures that the model will find any relationship between the edge and the node if it exists and will take this into account. For example, the system might use just the information 'isMarriedTo' to infer that someone is a human, but may infer that 'isMarriedTo' in combination with a man usually refers to a woman.

4.3.8 Hot start

It is hard to classify the algorithm in terms of the normal ontology alignment methods. It is not purely string matching, since it takes into account information from the context. However, it is also not structural, as it can work without a seed alignment and uses information other than the structure as well. It is not logical since it does not use inference. Nor is it terminological, which looks at dictionaries or other ontologies to find matches. However, it does use other ontologies (or text corpi) implicitly, so if it must be grouped it would be terminological. The algorithm can find matches based purely on the given node and edge labels. However the algorithm might benefit from a hot start. Such a seed alignment might allow the algorithm to train with certain words that are known synonyms as if they were the same word, thus increasing the number of training samples per word (on known relevant words) and decreasing sparsity. This may help the model, improving performance. However, since some information from other algorithms is now used, it would not be fair to compare the results to the cold start algorithm results. For example, it may just add the results from the generic string matcher to its own results, improving performance, without learning any new relationships. A new testing method had to be designed for this algorithm. This method would have to compare it to the string matchers, to see if it improved over their results, rather than compare it to the cold start algorithm. However, if we compare the cold and hot algorithms with the generic string matchers, we may be able to compare them indirectly.

4.4 Discarded possibilities

4.4.1 Training on text

Rather than training on ontologies, it would be possible to train the model on text. Since there is more text available than ontologies when counting the number of training samples, it would make sense to use a purely text-trained model. However, this method has downsides which made me decide not to use it. Namely, the context of a word in a text corpus is very different from a node in an ontology. A word is surrounded by the words directly around it, or the sentence it is in, or the paragraph or entire document. However all these contexts are just words, usually represented as a bag of words to ensure enough data for the model. In an ontology, the context is a combination of a node with an edge, which have a very different relation than merely co-occurring. The relation may be inferred from one sentence, or many different documents.

4.4.2 Just context nodes

4.4.3 Just edges

4.4.4 Context matrix representation

4.5 Implementation

5 Evaluation

The goal of the research is to find an ontology alignment algorithm A and proof that it performs better than current techniques. However, as current techniques may combine many different algorithms and sources of information, maybe such a comparison is not fair, nor is a comparison with individual algorithms as they may be optimised for being combined. So, instead we may want to see if performance is improved when adding A to an ensemble of algorithms $E = (A_1, .., A_n)$.

5.1 Evaluation methods

The evaluation should be done using a performance measure $P(Al, T)$ that takes the alignment an algorithm produces given an ontology and the ground truth. If both the alignment and truth are considered sets of alignments, i.e. $Al = \{a_1, .., a_n\}$ and $T = \{a'_1, .., a'_m\}$, then the True Positives $TP = \{a|a \in Al, a \in T\}$, False Positives $FP = \{a|a \in Al, a \notin T\}$ and False Negatives $FN = \{a|a \in T, a \notin Al\}$ determine the performance of the alignment. The True Negatives do not matter, as there are extremely many, and the goal of ontology matching is to find True Positives. Therefore the Jaccard similarity should be used as a performance measure.

The Jaccard similarity is calculated as follows

$$J = \frac{TP}{TP + FP + FN}$$

i.e. the fraction of the true positives over the sum of the true positives and mistakes.

The Jaccard similarity is a number between 0 and 1, with a higher number being better. Therefore if we compare two alignments, the one which is most similar to the truth, i.e. has a higher Jaccard similarity, is the better one.

However, if we want to find out if an algorithm improves an ensemble, we calculate the improvement as follows:

$$I = J_{E \cup \{A\}} - J_E$$

where J_E is the Jaccard coefficient of an ensemble E. To see if an algorithm is useful to an ensemble, the improvement score I has to be *significantly* higher than zero. To check if a score is significant, a statistical significance test will have to be carried out. The test works as follows: generate N solutions randomly and score them. Then score the method you want to investigate. If the score is in the top p%, it is significant, otherwise it is not. In this case the solutions are $E \cup \{R\}$, where R is an algorithm that generates random alignments.

5.2 Data

The data used comes from a list of data sets that the OAEI provides. Every data set contains two ontologies that need to be aligned and a ground truth alignment. The goal is straightforward: find the alignment given only the ontologies and whatever outside resources are needed.

5.2.1 Parts of OWL ignored

6 Algorithm

6.1 Idea

6.2 Theory

6.3 Product

7 Results

8 Conclusions

8.1 Recap research questions

8.2 Recap hypotheses

8.3 Conclusion per hypothesis

8.4 Overview of results

8.5 Final conclusion

9 Discussion

9.1 Interpretation

9.2 Unexpected results

9.3 Expected results

9.4 Future research

A Appendix 1

B Appendix 2

References