Mapping Questions to Ontology Components for Question Answering over Linked Data

Marius Valeriu Stanciu, Eugen Vasilescu, Stefan Ruseti, Traian Rebedea
University Politehnica of Bucharest, Faculty of Automatic Control and Computers
Bucharest, Romania
{marius.stanciu1710, eugen.vasilescu}@cti.pub.ro, {stefan.ruseti, traian.rebedea}@cs.pub.ro

ABSTRACT
This paper describes a statistical method to identify ontology components within natural language questions. The main purpose for this step is to improve question answering systems over linked data by reducing the ambiguity in the subsequent matching and query generation steps. To accomplish this task, we have trained a Conditional Random Field (CRF) classifier to label sentence tokens with the core data elements of the DBpedia ontology. The classifier was trained on a manually annotated corpus labelled with ontology elements for each token. Several features were investigated for the classifier and the results (F1=0.92) prove that this task can be successfully solved using the CRF tagger.

Author Keywords
Natural Language Processing; Question Answering; CRF Tagger; Ontology; DBpedia

ACM Classification Keywords
I.2.7 Artificial Intelligence: Natural Language Processing

INTRODUCTION
Knowledge bases are playing an increasingly important role in enhancing the intelligence of the Web. Such an example is DBpedia [1], which collects information from Wikipedia and organizes it in a machine-friendly manner. The problem of question answering over linked data is not a trivial task, even with all the available information. One of its hardest sub-tasks is matching tokens with their corresponding elements in the ontology. Having this done, processing the question intent and translating it into SPARQL queries will not be so far from reachable anymore. DBpedia organizes its data elements into 3 core categories:

- Entities: the subjects of description within the ontology. Each one has a dedicated web page, and corresponds to a Wikipedia page from which the information has been aggregated. Descriptive data (literal dates, strings, numbers) or other entities are linked through properties.
- Types: classes of entities structured hierarchically. The ontology by itself contains a limited number of types consisting of relatively vague nouns (e.g. *Person*, *Religion*, *City*) but types from multiple other sources have been added. For example, Yago [11] types usually represent more specific item categories such as *President* or *American Lawyers*, but they are considerably less regulated and consistent.
- Properties: they are usually predicate-like structures that link additional data to entities. Their representation is the most unpredictable as they do not follow any obvious syntactic or semantic patterns.

RELATED WORD

Named Entity Recognition
One popular example of a well-known and similar task to ours is called Named Entity Recognition (NER). It determines and classifies named entities from a text into predefined categories like persons, organizations, date, time, or money. For example, Stanford NER [5] uses the Maximum Entropy Markov Model algorithm [8]. Our task is a bit more complex because the difference between properties and types is not so obvious, but the approach to solve the problem should not be too different

Question Answering over Linked Data
Like we mentioned in the introduction, the ultimate goal of this model is integration within a question answering system, thus it is worth mentioning a few examples and what impact our algorithm could have.

One example is QAnswer [10], which uses a pipeline architecture for processing the questions. The first and most crucial step in the algorithm is trying to detect the elements within the question (they used individuals, types and properties) and map them to their corresponding resource in the ontology. After this step is completed and relationships between elements are made, a SPARQL query is built and, using a Virtuoso-opensource endpoint (https://github.com/openlink/virtuoso-opensource), their model can generate the answer. They have developed three separate mapping algorithms based on the characteristics of every ontology element and then picked the most appropriate sequence. Since they had no information about what the words could represent, they had to construct different interpretations of the questions and, based on some scoring algorithm, they picked the best one. This approach didn't produce the desired result all the time so, some questions were compromised. If they had additional information about the question (like our algorithm could provide), they would have probably had better results.

Another question answering tool worth mentioning is Xser [12], which was ranked first in QALD-4 and QALD-5 (see https://qald.sebastianwalter.org/). They have a two-layered

architecture, where the first one tries to label all phrases with one of the following tags: entity, relation, type and variable. This is the part that interests us the most because, for doing this, they also had to train a classifier, more precisely a structured perceptron [3]. The algorithm shows good results but we believe that it could benefit from the use of a sequence labeling algorithm. Moreover, using additional features for the classifier could further improve the precision. The second step of their algorithm is to map the discovered elements to a knowledge base, like DBpedia and construct the query.

PROPOSED SOLUTION

The main challenge consists in finding correlations between the 3 core data elements described in the introduction and natural language patterns in questions.

Labels

Our algorithm should be able to classify every word as part of an ontology element (entity, type, property) or mark them as irrelevant. We needed to find a standard for doing this and, since the elements could contain more than one word (e.g. *Barack Obama*), we chose the IOB tagging standard (Inside, Outside, Beginning). We defined the following tags: EB (entity beginning), EI (entity inside), TB (type beginning), TI (type inside), PB (property beginning), PI (property inside) and N (none).

Examples

To illustrate the labeling task, let us consider the following examples:

- *Which is the largest city in Australia?* In this example, we have one entity: *Australia*, one type: *city* and one property: *largest*. The other words cannot be mapped to an element in the ontology so we can consider them as irrelevant. So, the tagging sequence for the above example becomes (N, N, N, PB, TB, N, EB).
- *What did Bruce Carver die from?* In this question it is very clear that we should consider *Bruce Carver* as an entity, but it may be difficult to choose the property. Our goal for this algorithm is to identify ontology elements in phrases so that instantiating them afterwards is easier. In other words, the labels should ease the mapping of *die from* to *dbo:deathCause*. The problem here is that *Bruce Carver* has multiple other similar properties like *dbo:deathDate*, *dbo:deathPlace* or *dbo:deathYear*. By choosing *die* as the only word for the property, distinguishing between the four options might prove impossible. As a result, when prepositions offer additional contextual information, they should also be included. Concluding, the correct tagging sequence becomes (N, N, EB, EI, PB, PI).
- *Give me all video games published by Mean Hamster Software.* This is a more complex example, containing all possible tags. Firstly, we can identify *Mean Haster Software* as an entity. It contains 3 words, so the tags should be EB, EI, EI. Additionally, we have one composite type: *video game* and one property: *published by*, constructed using the same rules as the above example. Then, the correct tagging sequence becomes: (N, N, N, TB, TI, PB, PI, EB, EI, EI).

Our corpus consists of 600 questions which were manually annotated with the previously defined tags. We have used Gate [4] for simplifying the annotation process, while the questions were taken from multiple sources: QALD-6, QALD-7 and WebQuestions Semantic Parses Dataset (https://www.microsoft.com/en-us/download/details.aspx?id=52763). From this corpus, 400 questions were used for the training process, while the rest were kept for validation.

Choice of Model

We considered several algorithms used to solve labeling problems in Natural Language Processing (NLP), including Hidden Markov Models, Maximum Entropy Markov Models, Conditional Random Fields, and Neural Networks.

The Hidden Markov Model (HMM) [9] is a well-known sequence labeling algorithm, but it's not the most reliable one because it has direct dependencies only between states and their direct observations. An improvement to this algorithm brings the Maximum Entropy Markov Model [8], which is inspired from the Hidden Markov Model and the Maximum Entropy theory. It models dependence between each state and the full observation sequence explicitly, but it suffers from the label bias problem (states with low entropy transition distributions tend to ignore their observations).

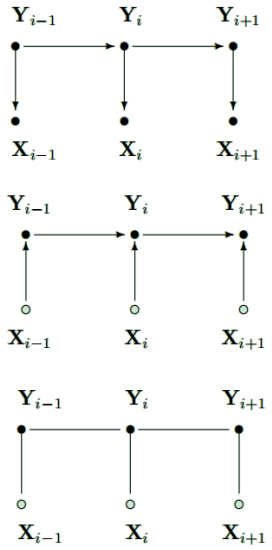

Figure 1: Difference between HMM (first), MEMM (second) and CRF (last) graphical models. From Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data [7]

Conditional Random Fields (CRF) [7] were designed to overcome the label bias problem while also taking into consideration the full context for the predictions. The differences between those three models can be clearly observed in Figure 1.

The other option that we considered was the Long Short-Term Memory (LSTM) neural network [6]. Its obvious advantage is that it automatically extracts features and it has good results in many areas. What made this approach impracticable for us is the big corpus required, one that we don't have. So, considering the arguments presented above, we chose the Conditional Random Field classifier, a choice which was proven to be a good one.

Choice of Features

Various features have been tested both individually and in relation with others. The following section is a description of the tested features and their influence over the results.

Considering entities are usually proper nouns (e.g. names of personalities, cities or organizations) within the ontology, identifying them is obviously similar with the Named Entity Recognition (NER) problem. Making use of thoroughly trained and tested NER models is definitely going to help at this stage. For this purpose, we have tested two popular models:

- Stanford 3-class NER tagger [5]: trained on various data sets, with generic classes: *Location*, *Person* and *Organization*. Even though there are 4 and 7-class versions of this labeling model, for the purpose of simply identifying named entities, the further classification would only add unnecessary complexity. Using this feature by itself results 82% accuracy over entities. However, as suspected, it does not contribute to identifying types and properties very much (less than 10% accuracy).
- Spacy NER tagger (see https://spacy.io/docs): a more modern model that shows minor improvements (about 5%) over the previous one.

On the other hand, types and properties do not follow such predictable patterns, making their identification less trivial. Analyzing DBpedia types reveals that they are usually vague common nouns (e.g. *person, football player, athlete*) and they are in close relation with entities or *WH* question words (e.g. *Where, What, Who*). Additionally, properties are usually predicate-like structures, connecting a subject to an object. Both types and properties are highly flexible regarding formulation and they are prone to polysemy, making string matching techniques less effective. This implies that part of speech and syntactic dependency information is essential to their identification. Again, this knowledge is available as widely-popular models in form of Part-of-Speech (POS) tagging and dependency parsing algorithms. The most promising models we considered are:

- Stanford POS tagger part of the NLTK package [2]: individual tests show an average of 43% labeling precision.
- Spacy POS tagger: this tagger offers two format variants, a coarse-grained one, based on the 12-class Google Universal POS Tags, and a fine-grained one, based on the 36-class Penn Treebank specification. Considering the relatively small train set, the lower detail specification shows minor improvements over the fine-grained one, averaging at 49% accuracy. However, upon future train set expansion, the more detailed one should be considered.
- Spacy Dependency Parser: used to extract the phrase type and the graph direction (whether tokens are the governing or the dependant part of a relation). This feature by itself shows modest results (average of 23%) However in relation to the others, it adds a considerable contribution.

Additionally, adding the following string based features showed significant improvements over the average precision.

- Lemmatized tokens: reducing words to their lemmatized version implied the emergence of more common text patterns, essential indicator for property and type recognition.
- Token index within sentence divided by the length of the sequence: types have the tendency to appear at the beginning of questions (excepting syntactic inversions, which are quite common in interrogative sentences) while properties and entities exhibit less regular positions.
- Word suffixes: similar words tend to have the same suffixes. For example, *son* is a suffix that appears mainly in person names (e.g. *Johnson, Anderson, Jackson, Dawson*), while *ies* commonly appears in tokens defining types (e.g. *cities, movies or countries*)
- Capital first letter: every word that starts with an uppercase letter has a great chance to be an entity. This helps the algorithm to be more confident about entity predictions.

Considering the highly context-dependent nature of the discussed elements, the relationship between the above-mentioned indicators is preserved and enforced using various templates, illustrated in formulas (1), (2), and (3).

- Unigrams:

$$w_{i-1}, w_i, w_{i+1} \quad (1)$$

- Bigrams:

$$(w_i, w_{i+1}), (w_{i-1}, w_i) \quad (2)$$

- Trigrams:

$$(w_{i-2}, w_{i-1}, w_i),\ (w_{i-1}, w_i, w_{i+1}),\ (w_i, w_{i+1}, w_{i+2}) \quad (3)$$

Tag	Precision	Recall	F1-Score	Support
EB	0.97	0.90	0.93	125
EI	0.90	0.86	0.88	73
N	0.96	0.97	0.97	369
PB	0.89	0.91	0.90	99
PI	0.83	0.89	0.85	53
TB	0.82	0.85	0.83	84
TI	0.81	0.73	0.75	13

Table 1. Precision and recall for each labeled tag from the test set. Average overall F1-score is 0.9

RESULTS

After a thorough analysis of feature combinations and templates, we finally settled on using mainly n-grams of Spacy POS, NER and dependency labels together with the above-mentioned string-based features. The model results are illustrated in Table 1. As can be observed, the highest confidence is achieved for entities with over 90% accuracy, result which was expected due to their predictable structure (almost every entity starts with a capital letter, for example). Types and properties can be easily confused, but the accuracy doesn't drop below 80%. Features like the POS tags and the dependency type returned by the dependency parser were essential in obtaining these results.

CONCLUSIONS

In this paper, we have proposed to use a Conditional Random Field (CRF) model to label question tokens as core data elements from the DBpedia ontology in order to improve question answering systems over linked data, such as DBpedia. The features we used are mostly generated by widely-popular Natural Language Processing tools such as POS taggers, NER taggers and dependency parsers. The ultimate goal of this method is to remove ambiguity and improve further matching in the context of question answering over linked data. Even though training and testing were done over a relatively small manually annotated corpus, the model shows good results with average labeling accuracy of 92%.

ACKNOWLEDGMENTS

This research was partially supported by University Politehnica of Bucharest through the Excellence Research Grants Program UPB–GEX 13/30.09.2016.

REFERENCES

1. Soren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *In 6th Int'l Semantic Web Conference, Busan, Korea*. Springer, 11–15.
2. Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, 69–72.
3. Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 1–8.
4. Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6).* http://tinyurl.com/gatebook
5. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *In ACL.* 363–370.
6. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
7. John Lafferty, Andrew McCallum, Fernando Pereira, and others. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, Vol. 1. 282–289.
8. Andrew Mccallum and Dayne Freitag. 2000. Maximum entropy markov models for information extraction and segmentation. Morgan Kaufmann, 591–598.
9. Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.
10. Stefan Ruseti, Alexandru Mirea, Traian Rebedea, and Stefan Trausan-Matu. 2015. QAnswer-Enhanced Entity Matching for Question Answering over Linked Data.. *In CLEF (Working Notes)*
11. Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web. ACM,* 697–706.
12. Kun Xu, Sheng Zhang, Yansong Feng, and Dongyan Zhao. 2014. Answering natural language questions via phrasal semantic parsing. In *Natural Language Processing and Chinese Computing*. *Springer*, 333–344.