Knowledge base population using natural language inference

Recski Gabor Kovacs Adam
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
recski@aut.bme.hu
adaam.ko@gmail.com

Abstract. We present a set of pilot experiments for augmenting a generic, open-domain knowledge base using a graph-based lexical ontology of English and simple inference rules. WikiData knowledge base contains facts encoded as relation triplets, such as author(George Orwell, 1984), based on which naive speakers can easily establish additional facts such as that George Orwell is a person and 1984 is some written work, most likely a book. To automate this type of inference we need models of lexical semantics that are more explicit than the distributional models commonly used in computational semantics. The 4lang library provides tools for building concept graph representations of the semantics of natural language text, its module `dict_to_4lang` processes entries of monolingual dictionaries to build 4lang-style definition graphs of virtually any English word. The representation of "author" will likely contain edges corresponding to facts such as `IS_A`(author, person) and write(author, book). We define simple templates that use these representations for inference over WiktData facts; preliminary evaluation on small samples suggests our method's precision to be in the 0.7-0.8 range.

Keywords: semantics; inference; natural language processing;

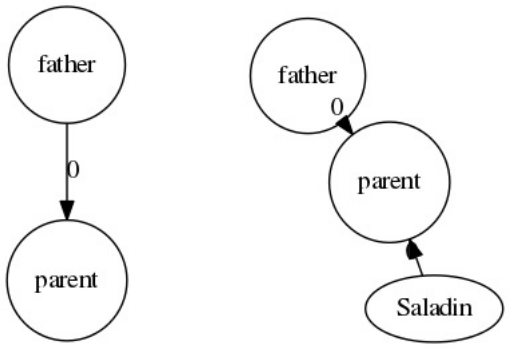
1 Introduction

In this paper I present a way of matching `WikiData` relations with arguments of `4lang` definitions. The `dict_to_4lang` tool automatically builds graphs from longman dictionary definitions. The full pipeline is available for download under an MIT license at `https://github.com/adaamko/4lang`. `WikiData` is a publicly available knowledge base and we can make triplets out of it in the form of `predicate(argument1, argument2)`. If we can make an assumption that these arguments corresponds to each other and a set of patterns can be applied to them, then we can convert a large amount of information from WikiData to the 4lang format and combine the two knowledge.

2 Combine WikiData and 4lang

The 4lang pipeline maps the output of the Standford dependency parser to subgraphs representing the words of each definition. For example `father` is defined

in longman as `male parent`. The `dict_to_4lang` tool uses this definition to build a 4lang graph seen in Figure 1(default). If we have a triplet coming from the WikiData knowledge base such as father(Az-Zahir Ghazi, Saladin) and we are ready to make an assumption that the second argument corresponds with the only IS A relation of out graph, then we can combine the fact with the longman definiton to obtain a new graph shown in Figure Figure 1(expanded). We have a new machine a IS A relation, the `Saladin` $\xrightarrow{0}$ `male` 0 edge that wasnt present before. We can see that we could obtain a completely new information which was unknown from the definition graph and from the Wikidata alone, and could only be present from the combination of the two. If we want to build 4lang graphs automatically from WikiData, we will require a method for matching these relations, as in the case above. The result will have to be reviewed, and only the reasonable ones have to be selected. If we can apply patterns to these triplets and definitons, we can have a large amount of information retrieved from the combination of the two.



(default) (expanded)

Figure 1: (default) father definition (expanded) father with gained information

3 Methods

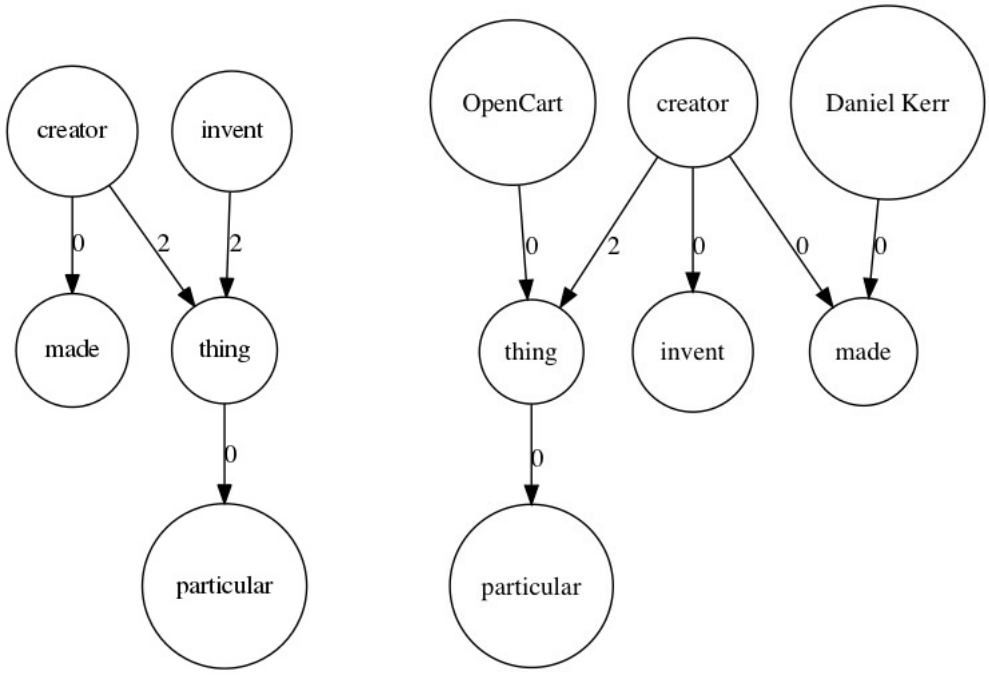
From the examination of the WikiData triplets, we can have a suspicion that if we say have a 0 edge in our definition graph `predicate` $\xrightarrow{0}$ `X` then in our new graph coming from the combination of the WikiData and the 4lang graph, a machine looks like `arg2` $\xrightarrow{0}$ `X` most likely going to have a place. And if we have an edge `predicate` $\xrightarrow{2}$ `X` in our original graph, then we will have an edge `arg1` $\xrightarrow{0}$ `X` in the newly constructed graph. As we can see in Figure 2(default) and Figure 2(expanded) new machines appeared such as `OpenCart` $\xrightarrow{0}$ `thing` and `Daniel Kerr` $\xrightarrow{0}$ `made` both of these appear to be valid information thanks to out pattern. Of course this is an ideal situation, this will not be always the case, there are many factors to be considered, when we apply these patterns

to out data. We have to take into account the fact, that the triplets coming from the WikiData are not always going to be valid information. This case can be seen in Figure 3(default) and Figure 3(expanded), where one of the arguments of a WikiData triplet was `novalue`, so the edge created from the triplet does not holds any information. There are cases, when the originally created graph is not completely parsed right from the definition. `flag` is definition in longman is: `piece of cloth with a coloured pattern or picture on it that represents a country`. The definition graph built from this definition is in Figure 4. The machine `flag` $\xrightarrow{0}$ `piece` obviously does not contain valid information, so the triplet flag(Belgium,flag of Belgium) with our current pattern would not add additional information to it. Our parser does not handle when there are multiple choices in a definition. For example longman defines employer `a person, company, or organization that employs people`. The graph constructed in Figure 5(default) and Figure 5(expanded). We have an IS a edge `Central Intelligence Agency` $\xrightarrow{0}$ `person` which we can presume is not a valid assumption, it would be rather a company. The next case, where out pattern can fail is when the WikiData and the longman has different definition of a word, it was the case when we examined the word Developer, which definition in longman was `a person or company that makes money by buying land and then building houses, factories etc on it` but the triplet in WikiData assumed it was a Software Developer as we can see Developer(De Blob, Blue Tongue Entertainment).

4 Testing and Conclusions

We designed two types of algorithms, firstly we found out that we have higher success rate by closing out those automatically what we think will give us false informations such as when the triplet's predicate in the wikidata doesn't have both of the arguments, or when there are multiple choices in the longman definition such as `a person, company, or organization that employs people`. We didn't select the ones where there was no edge or no node found. After closing out the graphs we think will give incorrect prediction, we designed algorithm to keep graphs from the remaining ones by identifying those we think have a higher chance of being right. These algorithms included keeping the ones when there is only one 0 or 2 type edge in the definition graph or there is no ingoing edge to the predicate. These two methods were essentially designed to keep the graphs that is we think is simpler by definition, and doing these we have a higher chance that the additional information we gained by the mapping will be correct.

We examinated the WikiData database containing 86,3 million triplet with 893 unique predicates with mean 96660 and variance 818079. Our system only handles one worder predicates, so we pre-filtered the dataset keeping 195 unique predicates generating 19,6 million triplet with mean 100772 and variance 566557. After pre-filtering we could apply our methods. First we kept only those triplets whose were complete, so both of their arguments were present. This method



(default) (expanded)

Figure 2: (default) creator definition (expanded) creator with gained information

didn't decrease the number of predicates, only the triplets keeping 14,2 million of them. From now on we can apply the rule specific filtering. For the first and the second rule(for edge type 0 and type 2) we closed out the ones with multiple choices in their definition as shown in Figure 5(default) and the ones we can't apply our pattern. For the rule targeting zero edges, we closed out 87 predicates, keeping only 108 with triplets count 9276533(9.2 million), mean 85893 and variance 388140. For the second rule we closed out 168 predicates with remaining 27 generating 1425311(1.4 million), mean 52789 and variance 99993. These statistics show that there are more predicates with the first rule where we can apply our pattern, and we close out more predicates for the second rule. For the remaining triplets we apply the algorithm trying to identify the graphs with higher success rate. For the first rule this means we keep 84 predicates producing 8420734(8.2 million) triplets with mean 100246 and variance 437205. For the second rule we keep 25 predicates generating 831288 triplets with mean 33251 and variance 54688. So our methods suggest that for the first rule we have 8.2 million and for the second rule we have 831 thousand newly generated correct edges.

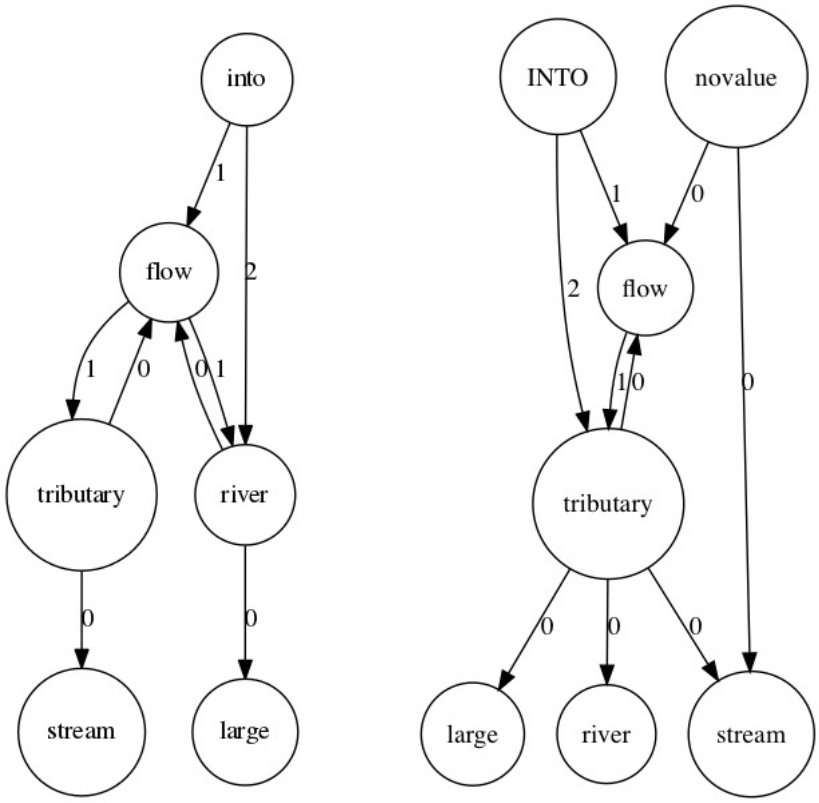


Figure 3: (default) tributary definition (expanded) tributary with gained information

To this point every one of our methods worked automatically, now we manually inspect the remaining predicates to gain information about the success rate of our methods. From the 84 predicates with the first rule, 55 was near perfect, reaching around 65 success rate, but generating 89 percent of the triplets. With numbers: 7624815(7.6 million) triplet, mean 138633, variance 532524. We chose 84 predicates out of the 108, because we thought they had better success rate. We inspected the 24 ones to validate our methods. The numbers show that only 12 of them were correct reaching only 50 percent success rate. For the rule number two this algorithm chose 25 predicates, 17 was near perfect reaching around 65 percent success rate as well but generating 90 percent of the data with 742349 triplets(mean: 46396, 62058). We have to notice that the errors doesn't always mean that the information we gained was incorrect, sometimes it just didn't generate useful informations, when we generate edges such as `X` $\xrightarrow{0}$ `something`. Our method is still not perfect, when a definition of

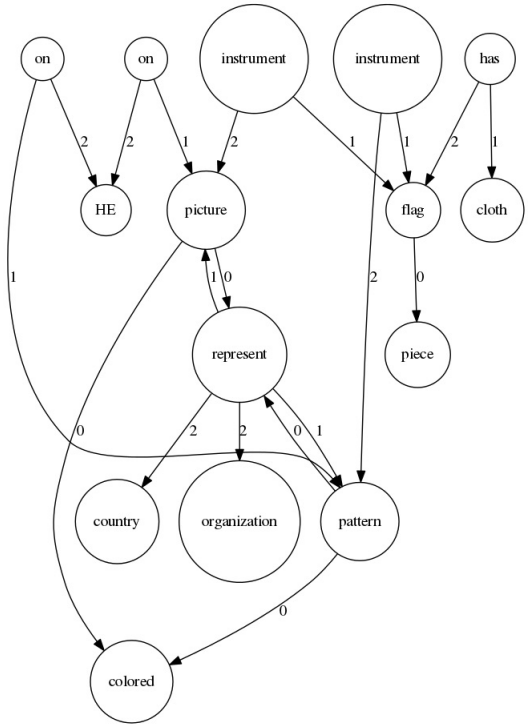


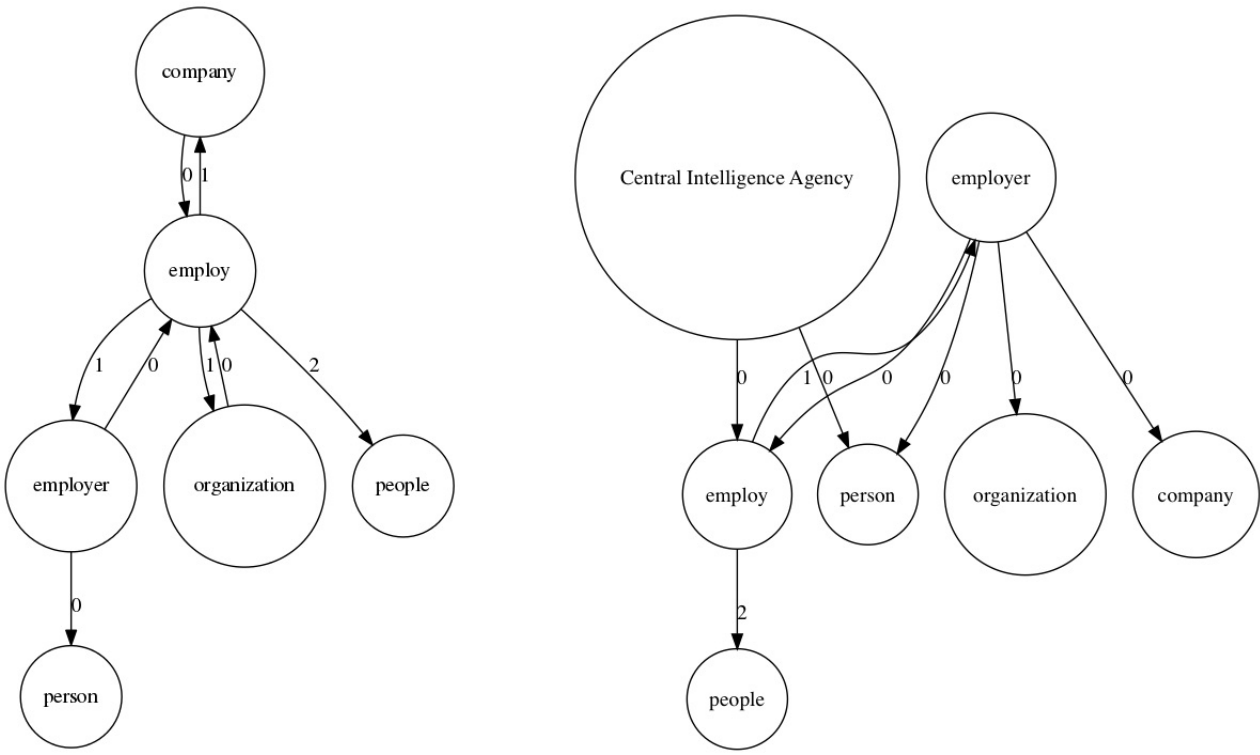
Figure 4: Flag definition

the predicate doesn't match the context of the triplet, another example for the first rule is the triplet (Ophiostoma ainoae, host, Scots Pine) with the generated graph shown in Figure 6(host), where host was interpreted as an army, wrongly. This is the case for the second rule for the triplet (2009 Cypriot Second Division, promoted, Alki Larnaca F.C.) with the generated graph shown in Figure 6(promote) where promote wasn't interpreted as promote a football club.

5 Bibliography

You have to use the `\makeAutBib` command for your bibliography. The input parameter of the command is the comma separated list of bib files without the extension. You can also find a sample bib file on the Workshop's web site where also the style file and this sample LaTeX template are provided. You can cite any of the entries of the bib files for example [1] or [2]. The advantage of using bib files is that only those bibliographies will appear in the References section that are cited in the text [3].

If you have any question related to writing your LaTeX paper please contact the chair of the conference(e-mail: aacs@aut.bme.hu).



(default) (expanded)

Figure 5: (default) employer definition (expanded) employer with gained information

You can submit your paper via the web page of the conference. The submission of the papers will be opened soon.

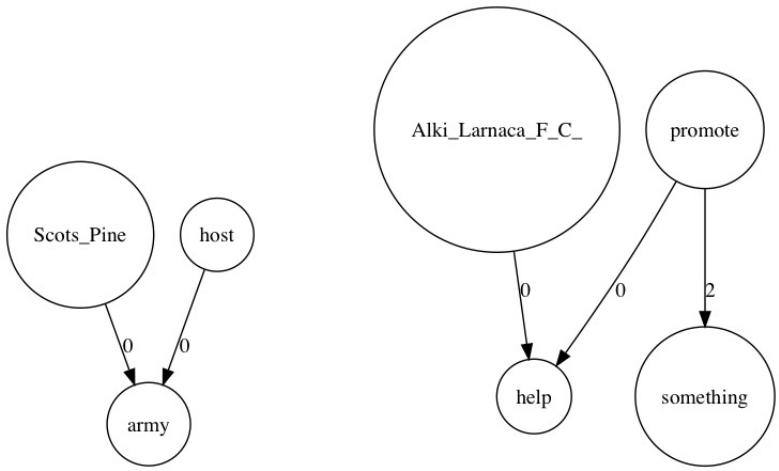
Acknowledgments

The author would like to express his thanks to Istvďż˝n Vajk [1] for his support as a scientific advisor. This work has been supported by the ... [2]

References

[1] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. of ACM SIGMOD International Conference on Manage-*

[1] Please mention the name of your advisor in the Acknowledgements section.
[2] Please mention the institution or organization that has supported your research work.



(host) (promote)

Figure 6: (host) definition (promote) definition

ment of Data (W. Chen, J. Naughton, and P. A. Bernstein, eds.), pp. 1–12, ACM Press, 05 2000.

[2] D. Burdick, M. Calimlim, and J. Gehrke, "Mafia: A maximal frequent itemset algorithm for transactional databases.," in *Proc. of the 17th International Conference on Data Engineering, (ICDE'01),* `hhtp://avalon.aut.bme.hu/~reni`, pp. 443–452, IEEE Computer Society, 2001.

[3] Reni, "Matlab scripts." `http://avalon.aut.bme.hu/~agi/research/`.