# Knowledge base population using natural language inference

Recski Gabor Kovacs Adam
*Department of Automation and Applied Informatics*
*Budapest University of Technology and Economics*
{recski,adaam.ko}@aut.bme.hu

**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;

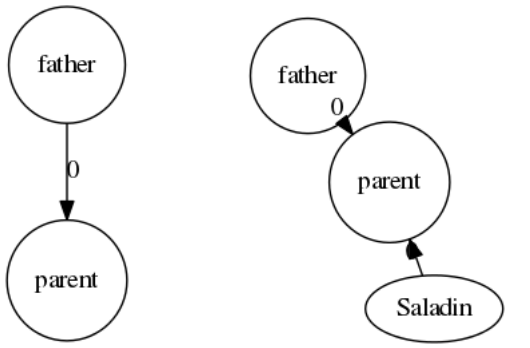
## 1 Introduction

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

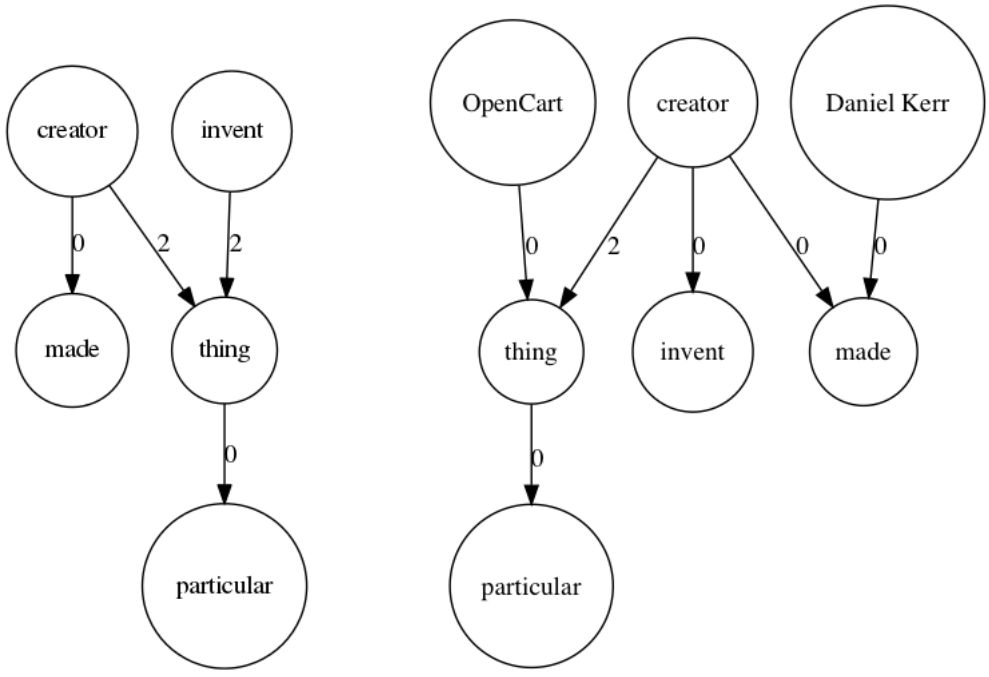
# 3 Methods

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

Knowing these facts we can make a few assumptions for increasing the possibility of the additional information's correctness. We can separate the two types of method we introduced earlier and inspect each one individually. Our initial thought was to design an algorithm that can identify the correct edges. For example, when there is only one 0 or 2 type edge in the definition graph or there is no ingoing edge to the predicate, there is a higher chance that the additional information that we gained by the mapping will be correct. Using these techniques we performed quantitative evaluation by manually inspecting a small output sample, that contained these kind of assumptions. We found out, that we have a higher success rate by closing out those automatically that we think will give us false information such as invalid information in the WikiData dataset or when there are multiple choices in the longman definition and we didn't select the ones where there was no edge or node found. By examinating the sample size of 3000 graphs, our algorithm for edge type 0 closed out 1252 of them and for edge type 2 it closed out 2647. From the remaining ones, we manually chose 100 random ones, and we found that 77 of them were near perfect. From these informations, we can assume that using the methods explained, we can reach 77 percent success rate. For gaining we could combine the algorithms designed for identifying the incorrect ones with the one choosing the right ones. This way from the remaining 1728 with type edge 0, our methods suggest that




(default) (expanded)

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

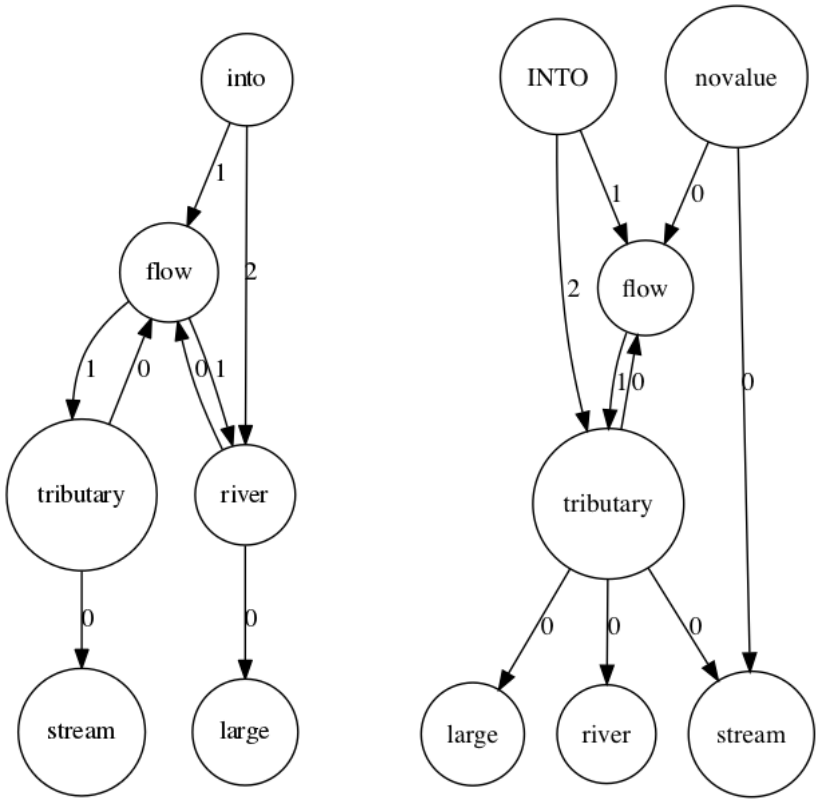
1559 is correct, 899 out of it is because of the no ingoing edge rule and 660 is coming from the single zero edge rule. With edge type 2 from the remaining 353 we think 220 is correct 55 of them is coming from the no ingoing edge, 37 because of the single edge rule and 128 containing both.

# 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).

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




(default) (expanded)

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

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

[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.

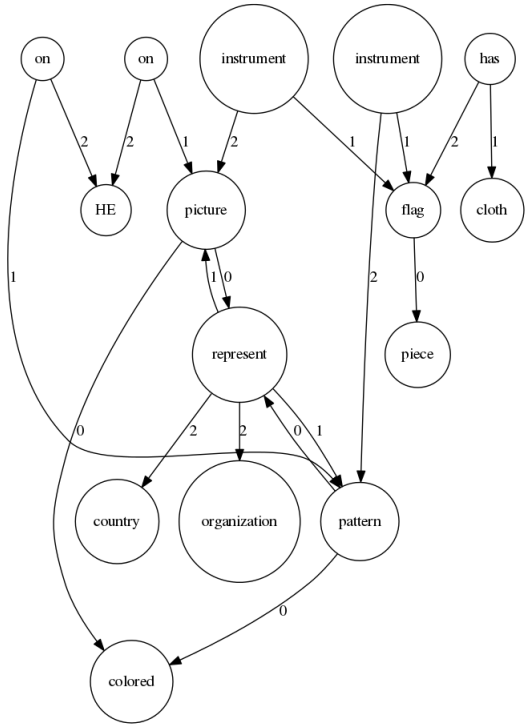
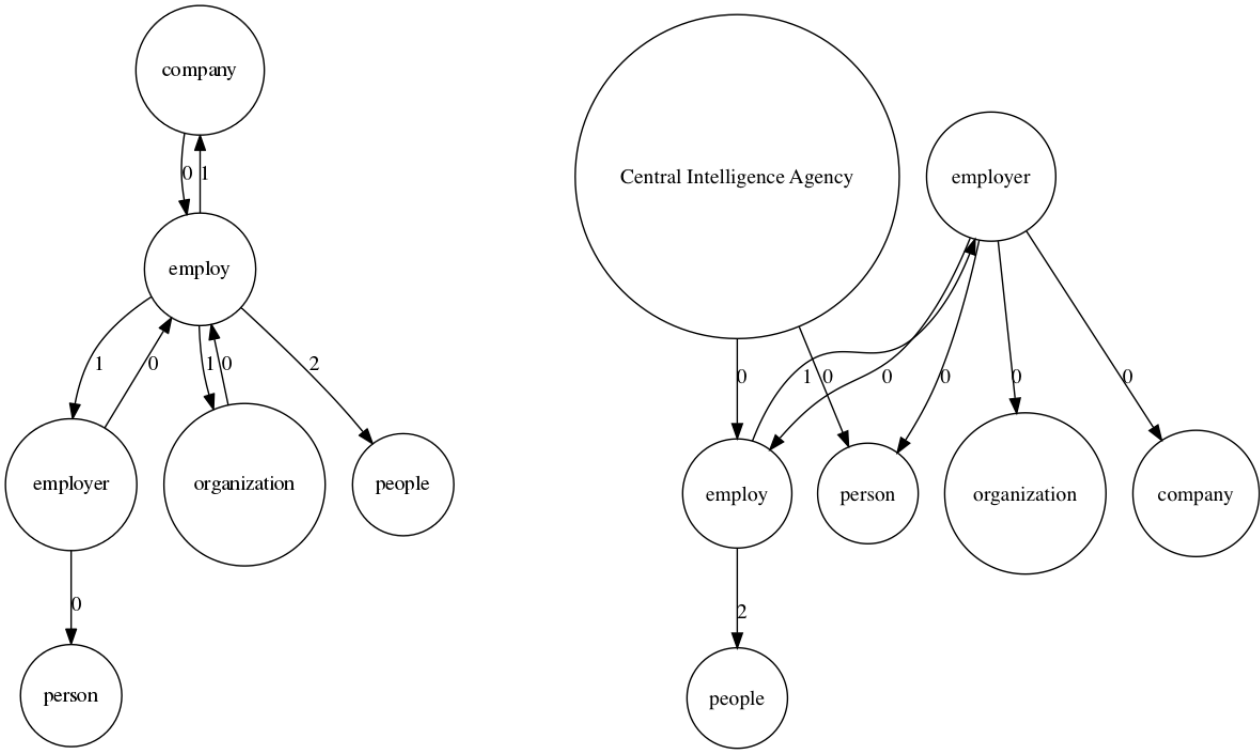



Figure 4: Flag definition

[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/`.




(default) (expanded)

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