The Wiki-Smasher, a Patty based fact replacer.

Alexander Tingström, Patrik Wallin, Robert Vegh, and Daniel Kruczek

Kungliga Tekniska Högskolan, Stockholm 10044, Sweden
atin@kth.se patwal@kth.se rvegh@kth.se kruczek@kth.se

Abstract. In this paper a program capable of finding facts in Wikipedia and replacing them with random facts of the same type is presented. The program utilized the syntactically-typed relational database PATTY to find patterns between arguments and to replace the second argument, thus changing the factual content. This program had mixed success rates with facts sometimes being exchanged with nonsensical sentences. On analysis of the data it was determined that grammatical heuristics and improved sentence isolation could be introduced to increase the quality of the data even though limitations in PATTY disallow perfect sentence replacement.

Keywords: PATTY, Natural Language Processing, Wikipedia, Databases, AI

1 An introduction to PATTY

The database used to locate facts for replacement and the corresponding new fact was generated by a system called PATTY, which constructed its relational database from text material taken from the whole of Wikipedia (June 21st, 2011). The idea behind PATTY is to automatically extract relational information from natural language text. However, there are many ways to express the same thing in natural language so the relations must be generalized and organized with care.

PATTY works with an external knowledge base (YAGO or Freebase) and processes a natural language text in four main steps to obtain a relational database.

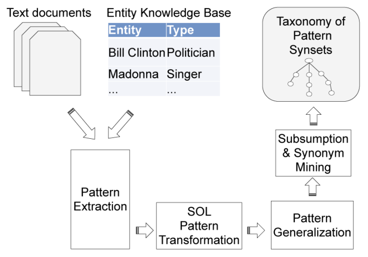

Fig. 1. A brief overview of the four-step process utilized by PATTY. Picture taken from [1].

1.1 Pattern Extraction

The first step is to extract patterns from the text corpus. A pattern is a string that connects a pair of entities in a sentence. The first step is to extract such patterns from the text, this is done using the Stanford Parser. However, only sentences with an actual pair of entities are of interest, so only sentences with at least two entities are emitted. This process yields basic patterns to work with.

1.2 SOL Pattern Transformation

The next step is to turn these basic patterns into SOL (Syntactic, Ontological, Lexical) patterns. These SOL-patterns are abstractions of the basic text patterns and contain words, POS-tags (part-of-speech), wildcards and ontological types. the POS-tags stands for a word of a specific part-of-speech class, such as verb, noun, etc. An ontological type is a semantic class name, for instance <Politician>. These ontological types are provided by the external knowledge base.
These SOL-patterns are generated by looking at N-grams of the text patterns, processing the more frequent ones and giving them type signatures via the external knowledge base. As an example, a SOL-pattern could look like this: <person>'s [adj] voice in * <song>

1.3 Pattern Generalization

The third step is to generalize these patterns further. This can be done in many ways, as an example words can be replaced by POS-tags, more wildcards can be introduced or an ontological type can be replaced by a more general type (<singer> can become <performer>). To do this, N-grams are utilized once more, where the relatively infrequent N-grams get replaced by POS-tags or types.

1.4 Subsumption and Synonym Mining

The last step is to arrange these SOL-patterns into sets of synonym expressions with a hierarchy tree. The notion of synonym sets are determined via support sets. A support set is all the pairs of entities that occur with a given pattern, thus a way to define synonym patterns is to look at their support sets and make them synonyms if their support sets are equal. Lastly, some patterns are more general than others, for example <artist> sang <song> is more general than <artist> covered <song>. This hierarchy is built by looking at intersections in the support sets. If one pattern has a support set that is a subset of another patterns support set it will be under that pattern in the hierarchy. The information is now ready to be inserted into a database where relational queries can be made.

2 The Wiki-Smasher

The Wiki-Smasher is implemented using a 4 module system described below. It uses the Stanford Part-Of-Speech tagger [2], Python and SQLite.

Module 1: POS-Tagger

In this module the Stanford POS-tagger [2] is used in order to find all nouns in the given corpus, as all PATTY arguments are nouns. The nouns are indexed and put into a list for faster processing.

Module 2: Patty-Translator

In the second module, with the indexed noun list as input, the nouns are matched to the PATTY arguments. At first there is a preprocessing step. For each noun that is followed directly by another noun, it and the following noun are queried to the database to determine whether or not they match an argument. This is repeated until The longest possible list of nouns that still matches an argument is found. This is then remade into a single entry in the noun list and the indexing remade accordingly. In the next step the nouns are matched to the arguments in PATTY. For each noun the module searches the database for a set number of arguments and adds them to an argumentlist with a corresponding index.

Module 3: Tupler

This module is where the PATTY arguments are distilled from the text. It receives as input a list of arguments of interest and a corresponding index list. Given these datasets it traverses the argument list one argument at a time, checking in both directions to see if there are connections between the arguments. So as to reduce the incidence of false positives it limits its search to arguments that are at a distance $D < D_{max}$ apart. The value for D_{max} can and should be adjusted based on the input data. Once a match is found it inserts the related PATTY tuple into the output list.

2.1 Module 4: Swapper

The final module completes the process and outputs an edited article. It is passed a list of PATTY tuples and a corresponding index list, using the PATTY database it then searches for arguments that have the same pattern-ID as the tuple and replaces the second argument with a randomly chosen argument. Then using the supplied index and wordlist the relevant changes can be made and the article replaced.

3 Sample replacements

An example run of the Wiki-smasher yielded the following text, a part of a wiki-article about Alfred Nobel can be found in appendix 1.

3.1 Successful replacements

<University of Kent> held 350 different patents. Here Alfred Nobel has been replaced i a meaningful replacement.

Alfred Nobel was a descendant of the <Block Design> scientist Olaus Rudbeck 1630–1702 . Swedish was replaced with 'Block Design'.

3.2 Unsuccessful replacements

The <International Rice Research Institute> married in 1827 and had eight children. A nonsensical statement.

Alfred Nobel was a descendant of the <Glentoran F.C.> scientist Olaus Rudbeck 1630–1702. Another nonsensical statement. Unfortunately these replacements often outnumber the successful ones, which can be seen in the appendix. The discussion will mainly deal with these errors.

4 Discussion

4.1 D_{max} vs sentence isolation

While the idea of narrowing the scope of argument searching via a maximum distance approach certainly reduces the number of false positives, a more effective approach would be to limit argument searches to one sentence at a time. This is because syntactical relations are limited by their very nature to one sentence at a time. Furthermore the chance of multiple relational patterns occurring in one sentence is rare. To utilize this to its fullest potential and further decrease the amount of false positives, sentences should be isolated by splitting on punctuation, and then operated upon.

4.2 Grammatical heuristics

The largest source of human detectable error in the output article was grammatical flaws. These flaws are the direct result of PATTY's design. PATTY attempts to place a level of abstraction on the arguments so that multiple relations are condensed into one relational pattern. This results in obviously erroneous arguments after replacement. This can be circumvented by implementing a grammatical heuristic which rates outputs based on their grammatical correctness and only allows correct sentences to be emitted.

4.3 Unavoidable error

It is important to note that PATTY incorrectly classifies relations around 15% of the time[1]. This means that regardless of how well the Wiki-Smasher is implemented there will still be errors. Short of improving PATTY there is nothing that can can be done to avoid this.

References

1. Nakashole, N., Weikum, G., Suchanek, F.: Discovering semantic relations from the web and organizing them with PATTY. SIGMOD Record, Vol. 42 No. 2 (June 2013)
2. Kristina Toutanova and Christopher D. Manning.: Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.

Appendix 1 - Sample Article

Note: All words in brackets have replaced words in the original article.

Alfred Bernhard Nobel 21 October 1833 – 10 December 1896 was a Swedish chemist , engineer , innovator , inventor and armaments manufacturer . He was the inventor of dynamite . Nobel also owned Bofors , which he had redirected its previous role as primarily an iron and steel producer to a major manufacturer of cannon and other armaments . <Cadet Forces Medal> held 350 different patents , dynamite being the most famous . His fortune was used posthumously to institute the <Guggenheim Fellowship> Prizes . The synthetic element nobelium was named after him . His name also survives in modern-day companies such as Dynamit <National academy> and AkzoNobel , which are descendants of mergers with companies <Organization XIII> himself established . Alfred Nobel at a young <Governors State University> . Born in Stockholm , <Zagreb> Nobel was the fourth <University of Mississippi> of Immanuel Nobel 1801–1872 , an <Columbia University> and engineer , and Carolina Andriette Ahlsell Nobel 1805–1889 . The <World Peace Corps Mission> married in 1827 and had eight children . The family was impoverished , and only Alfred and his three brothers survived past childhood . Through his father , Alfred Nobel was a descendant of the <United States> scientist Olaus Rudbeck 1630–1702 , and in his turn the boy was interested in engineering , particularly explosives , learning the basic principles from his father at a young age . Alfred Nobels interest in technology was inherited from his <Technology (album)> , an alumnus of Royal Institute of Technology in <Technology (album)> . <Stuttgart> various business failures , Nobels father moved to Saint Petersburg in 1837 and grew successful there as a manufacturer of machine tools and explosives . He invented modern plywood and started work on the torpedo . In 1842 , the family joined him in the city . Now prosperous , his parents were able to send Nobel to private tutors and the boy excelled in his studies , particularly in chemistry and languages , achieving fluency in English , French , German and Russian . For 18 months , from 1841 to 1842 , Nobel went to the only <National Hockey League> he ever attended as a child , the Jacobs Apologistic School in Stockholm . As a young man , Nobel studied with <National academy> Nikolai Zinin then , in 1850 , went to Paris to further the work and , at 18 , he went to the United States for four years to study chemistry , collaborating for a short period under inventor John Ericsson , who designed the American Civil War ironclad USS Monitor . Nobel filed his first patent , for a gas meter , in 1857 .