Unsupervised Relation Extraction From Web

Anurag Sai Vempati
Department of Electrical Engineering
Indian Institute of Technology, Kanpur, India
Email: vanurag@iitk.ac.in

Bhavishya Mittal
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
Email: bhavishy@iitk.ac.in

Advisor: Dr. Amitabha Mukerjee
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
Email: amit@cse.iitk.ac.in

***Abstract*—Extracting knowledge from web is an important NLP problem. Since any information worthwhile is mostly of the SPO (Subject, Predicate and Object) form, we will be aiming to extract such tuples from the huge web data available today. We also aim to build a query module that can search for the most relevant tuple and return it. The framework involves three main modules: A *relation extractor*, *Learning Module* and a *Query Module*. Dependency graph obtained from the Stanford parser is used to extract the tuples. An SVM is trained to learn to distinguish trustworthy tuples from the non-trustworthy to reduce the computational complexity involved in using a parser on each and every sentence. The classifier also helps in removal of noise which crept in due to the parser. Similar tuples are grouped together for obtaining a frequency count. During the query process, all the tuples with certain amount of overlap with the query are returned and ranked according to their relevance to the query.**

***Index Terms*—Relation extraction; Unsupervised Learning; Knowledge from Web.**

I. Introduction

Relation extraction is an important field in Information Extraction. A relationship extraction task requires the detection and classification of semantic relationship mentions within a set of artifacts, typically from text or XML documents. Application domains where relationship extraction is useful include gene-disease relationships [11], protein-protein interaction [12] etc. Traditional, approaches to relation extraction have focused on specific domain. These supervised approaches takes a lot of time and labour in tagging the corpus. Some previous approaches like kernel based approach [13](based on tagged corpus),unsupervised approaches [14](depend lot on chracteristics of wikipedia) are not source independent. This type of approaches can not work where the number of target relations is very large, or where the target relations cannot be specified in advance.

Open IE [1] solves this problem by identifying relation phrasesphrases that denote relations in English sentences. The automatic identification of relation phrases enables the extraction of arbitrary relations from sentences, obviating the restriction to a pre-specified vocabulary. Our work is an extension to their work. In our approach, we chunk together Noun Phrases and do a noun-phrase mapping to each of the tuple entities to get better and meaningful relations from the data. During the tuple grouping and also in the query module we look at the semantic similarity between the tuples and query to report the successful extractions. We also made our approach language independent by removing the constraints used in Open IE. More about this is discussed in the Framework section.

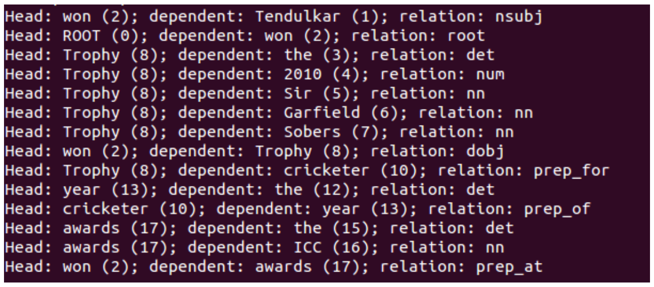
II. Framework

We will be improvising upon the framework described in [1] and [2]. We will first describe the Relation extraction module, then the Learning module and finally the Query module. We will be using some example sentences
1: *"Tendulkar won the 2010 Sir Garfield Sobers Trophy for cricketer of the year at the ICC awards."*
2: *"The American Civil War, also known as the War between the States or simply the Civil War, was a civil war fought from 1861 to 1865 in the United States after several Southern slave states declared their secession and formed the Confederate States of America."*
for illustrating some results.

A. Relation Extraction

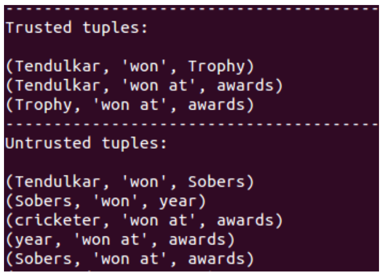
A relation extractor skims through the dataset, picking each sentence and tries to find tuples of the format: (e_i, r_{ij}, e_j) where, e_i and e_j are entities playing the role of a subject and an object and r_{ij} is a relation between them. We consider nouns as candidates for entities and verbs as a candidate for the relation with only constrain being, e_i should occur before e_j in the input sentence. Now, there might be more than two nouns in a sentence and more than one verb between them. So we need some kind of framework that can relate various words in a sentence. This is where we look out for the Stanford parser [3] which provides a dependency graph [4] between various words in the sentence in the form of a (head, dependent, relation) tuple. We use the collapsed dependencies where prepositions are incorporated into the relation, to get a more compact graph. The dependency graph obtained for the example sentence **1** can be seen in Fig.1.

We then find the POS tags of each word in the sentence and extract out all the nouns and verbs. We form all possible tuples



```
Head: won (2); dependent: Tendulkar (1); relation: nsubj
Head: ROOT (0); dependent: won (2); relation: root
Head: Trophy (8); dependent: the (3); relation: det
Head: Trophy (8); dependent: 2010 (4); relation: num
Head: Trophy (8); dependent: Sir (5); relation: nn
Head: Trophy (8); dependent: Garfield (6); relation: nn
Head: Trophy (8); dependent: Sobers (7); relation: nn
Head: won (2); dependent: Trophy (8); relation: dobj
Head: Trophy (8); dependent: cricketer (10); relation: prep_for
Head: year (13); dependent: the (12); relation: det
Head: cricketer (10); dependent: year (13); relation: prep_of
Head: awards (17); dependent: the (15); relation: det
Head: awards (17); dependent: ICC (16); relation: nn
Head: won (2); dependent: awards (17); relation: prep_at
```

Fig. 1: Collapsed Dependencies for the sample sentence **1**



```
-------------------------------------
Trusted tuples:

(Tendulkar, 'won', Trophy)
(Tendulkar, 'won at', awards)
(Trophy, 'won at', awards)
-------------------------------------
Untrusted tuples:

(Tendulkar, 'won', Sobers)
(Sobers, 'won', year)
(cricketer, 'won at', awards)
(year, 'won at', awards)
(Sobers, 'won at', awards)
```

Fig. 2: Trustworthy and non-trustworthy tuples for the sample sentence **1**

of the form (e_i, r_{ij}, e_j) where e_i, e_j are nouns and r_{ij} is a verb. For each such tuple we find the distances d_1 and d_2, the lengths of the shortest path from e_i and e_j to r_{ij} respectively.

Now for each tuple thus obtained, we consider it "trustworthy" if the sum of d_1 and d_2 is less than a tolerance level and "non-trustworthy" otherwise. The tolerance level is fixed at 3 after some experimentation. Few trustworthy and non-trustworthy tuples for the input sentence **1** are shown in Fig. 2.

As it can be seen, the tuples are not much informative. For example, *(Tendulkar, won, trophy)* has no information regarding what trophy he actually won. To overcome this problem, we mapped each entity to it's *Noun-Phrase* parent. In case there are multiple *Noun-Phrase* parents to the entity, we considered all of them. For tuples where both entities are subset of a same noun-phrase, we didn't perform the mapping. Sometimes performing the mapping has made some of the non-trustworthy tuples a trustworthy one. We have left out such tuples as well. Few of the resulting tuples for sample sentence **1** and **2** can be seen in Fig. 3 and Fig.4 respectively.

The first half of the sample sentence **2**, *"The American Civil War, also known as the War between the States or simply the Civil War"* is wrongly parsed as shown in Fig. 5. *'or'* is shown as a conjunction between *"the states"* and *"simply the Civil war"* when it should have actually been a conjunction between *"the war between the states"* and *"simply the Civil war"*. This is he reason for tuple *(The American Civil War, known as, the War between the States)* being missed out. Nevertheless, we have a more descriptive tuple *(The American Civil War, known as, the War between the States or simply the Civil War).*

B. Learning Module

Stanford parser is a heavy linguistic mechanism. It takes around 40-50 seconds for extracting tuples for a sentence of length 40 words. This makes the mechanism un-scalable to the large web data. So, we tried to train a classifier that can classify trustworthy tuples from the non-trustworthy ones. We will use the tuples obtained from the parser to train a classifier. Such a classifier also helps in removing noise from the training data provided by the parser.

We use a Noun-Phrase chunker available in NLTK to extract possible candidates for entities e_i and e_j. The verbs in the sentence are considered for relation r_{ij}. We haven't enforced any constraint that the r_{ij} has to be present between e_i and e_j as in [1] but instead introduced features involving ordering of e_i, e_j and r_{ij} in the feature vector. Since many of the languages like Hindi, Telugu have subject-object-predicate ordering as opposed to a subject-predicate-object ordering in English, such an approach makes our framework language independent.

Several features are extracted from each tuple to get a feature vector. The feature vectors are normalized before training a classifier on them. The features we have considered are:

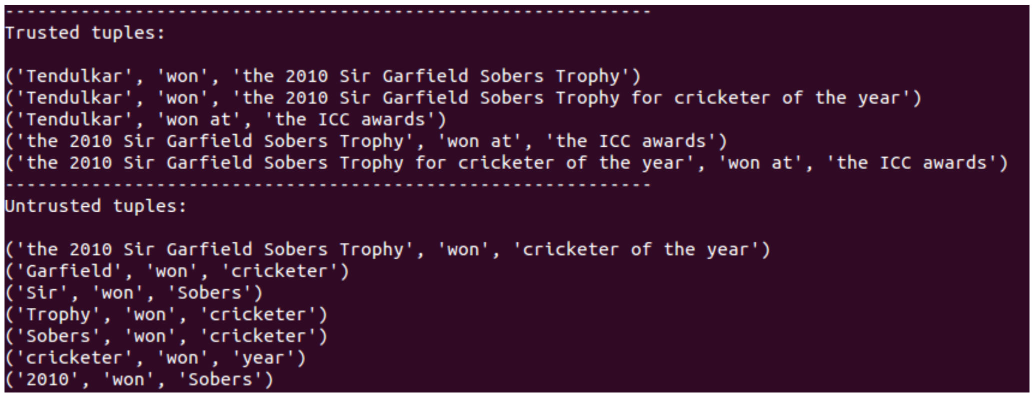
- Presence of a Parts-Of-Speech (POS) tag in e_i
- Presence of a Parts-Of-Speech (POS) tag in e_j
- Presence of a Parts-Of-Speech (POS) tag in r_{ij}
- Distance between e_i and e_j in the input sentence
- Distance between e_i and r_{ij} in the input sentence
- Distance between e_j and r_{ij} in the input sentence
- Ordering of e_i and r_{ij} (e_i followed by r_{ij} or the other way) in the input sentence
- Ordering of e_j and r_{ij} (e_j followed by r_{ij} or the other way) in the input sentence
- Scaled lengths of the words e_i, e_j and r_{ij}. $\left(\frac{length-of-entity}{length-of-sentence}\right)$

A POS tagger available in NLTK is used for getting POS features. The set of POS tags considered for entities e_i and e_j are 'NN', 'NNS', 'NNP', 'NNPS', 'JJ', 'JJR', 'JJS', 'PRP', 'PRP', 'RB', 'RBR', 'RBS', 'CC', 'CD', 'WDT', 'WP', 'WP', 'WRB'. The set of POS tags considered for relation r_{ij} are 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ', 'WDT', 'WP', 'WP', 'WRB', 'RB', 'RBR', 'RBS'. The description of each of the tags can be found at [7]. In the end, it comes out to about 60 features per tuple.

A thousand sentences are picked from Brown corpus [16] to extract the tuples. Brown corpus is chosen because it has diverse set of articles ranging from science, news editorials to religious scriptures and fiction. The feature vectors obtained from these tuples are used to train an SVM classifier with Radial Basis Function (RBF) kernel.

C. Query Module

We make a single pass over the entire corpus and find trustworthy tuples as determined by the classifier. All the similar tuples are then paired together. This gives us a count

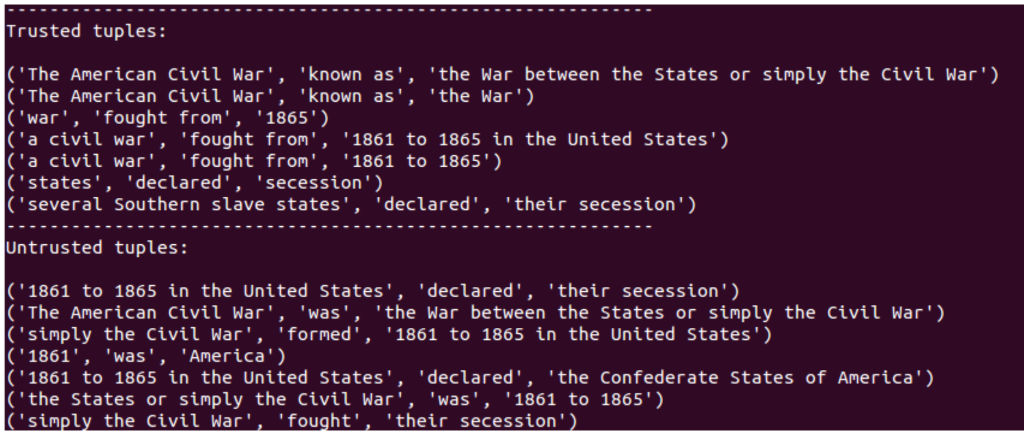


```
------------------------------------------------------------
Trusted tuples:

('Tendulkar', 'won', 'the 2010 Sir Garfield Sobers Trophy')
('Tendulkar', 'won', 'the 2010 Sir Garfield Sobers Trophy for cricketer of the year')
('Tendulkar', 'won at', 'the ICC awards')
('the 2010 Sir Garfield Sobers Trophy', 'won at', 'the ICC awards')
('the 2010 Sir Garfield Sobers Trophy for cricketer of the year', 'won at', 'the ICC awards')
------------------------------------------------------------
Untrusted tuples:

('the 2010 Sir Garfield Sobers Trophy', 'won', 'cricketer of the year')
('Garfield', 'won', 'cricketer')
('Sir', 'won', 'Sobers')
('Trophy', 'won', 'cricketer')
('Sobers', 'won', 'cricketer')
('cricketer', 'won', 'year')
('2010', 'won', 'Sobers')
```

Fig. 3: Trustworthy and non-trustworthy tuples for the sample sentence **1** with noun-phrase mapping



```
------------------------------------------------------------
Trusted tuples:

('The American Civil War', 'known as', 'the War between the States or simply the Civil War')
('The American Civil War', 'known as', 'the War')
('war', 'fought from', '1865')
('a civil war', 'fought from', '1861 to 1865 in the United States')
('a civil war', 'fought from', '1861 to 1865')
('states', 'declared', 'secession')
('several Southern slave states', 'declared', 'their secession')
------------------------------------------------------------
Untrusted tuples:

('1861 to 1865 in the United States', 'declared', 'their secession')
('The American Civil War', 'was', 'the War between the States or simply the Civil War')
('simply the Civil War', 'formed', '1861 to 1865 in the United States')
('1861', 'was', 'America')
('1861 to 1865 in the United States', 'declared', 'the Confederate States of America')
('the States or simply the Civil War', 'was', '1861 to 1865')
('simply the Civil War', 'fought', 'their secession')
```

Fig. 4: Trustworthy and non-trustworthy tuples for the sample sentence **2** with noun-phrase mapping

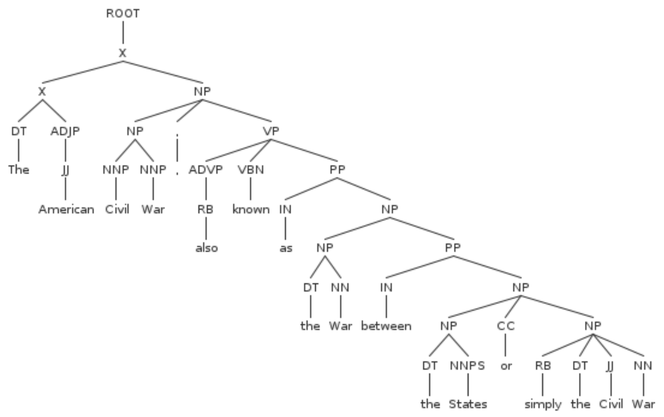


Fig. 5: Failure of the parser in parsing the sentence *"The American Civil War, also known as the War between the States or simply the Civil War."*

of each unique tuple and the ones with higher count must have appeared quite often and are hence reliable. Since tuples don't occur in exactly same wording, we need a framework to group tuples based on their semantic content. We will be following the method illustrated in [9] for achieving this.

To relate semantic content of two tuples we need to first disambiguate sense of each word in it. For example, the word interest has different sense in *"...bank paid out an interest..."* and *"...he lacks interest..."*. Wordnet provides synsets for each word with different senses of that word in each of them. Each sysnset in-turn belongs to a hierarchy of synsets (called taxonomy) consisting of hypernym, hyponym, meronym for nouns and hypernym, troponym for verbs. Each of these can be described as [8]:

- *Hypernym*: Y is a hypernym of X if every X is a (kind of) Y ('Automobile' is a hypernym of 'truck')
- *Hyponym*: X is a hyponym of Y if Y is a hypernym of X
- *Meronym*: Y is a meronym of X if Y is a part of X ('window' is a meronym of 'building')
- *Troponym*: The verb Y is a troponym of the verb X if the activity Y is doing X in some manner ('to lisp' is a troponym of 'to talk')

To disambiguate a word, a k-neighbourhood context around it is picked (k words to the left and right). Each synset (s_w) the word w belongs to, is pair-wise compared with the synsets of the hypernyms,hyponyms, meronyms and troponyms of s_w. The sysnset s_w that has the maximum overlap with the rest is chosen as the sense of that word. Since this problem has a complexity which is exponential in the number of sysnsets for a word, beam-search is used to narrow down the search space.

For a 2-neighbourhood context, it can be put down mathematically as:

$$OverallScore(s_1, s_2) = Score(hype(s_1, hypo(s_2)) + Score(sense(s_1), hypo(s_2)) + Score(hype(s_1), sense(s_2))...$$

where, s_1 and s_2 are the synsets of word in consideration and it's neighbour. $hype(s)$, $hypo(s)$ are the hypernym and hyponym of s and $sense(s)$ is the description of the sense of sysnset s. For example, the word 'pine' has two synsets s_1, s_2 with senses *"kind of evergreen tree with needle-shaped leaves"* and *"waste away through sorrow or illness"* respectively. Of all the synsets the word belongs to, we now know which synset it is in based on the sentence it was used in.

To find semantic similarity between two sentences, we perform sense disambiguation of words in both the sentences and get the sysnsets to which each word belongs to. A matrix R is built where the $(i, j)^{th}$ entry is inversely proportional to the distance between the synsets of i^{th} word of first sentence and j^{th} word of the second sentence along the taxonomy tree. This matrix essentially signifies semantic similarity between the words of each sentence. The problem of evaluating semantic similarity between the two sentences is then framed as the problem of computing a maximum total matching weight of a bipartite graph in which the sentences are two sets of disjoint nodes. The similarity is scored on a scale of 0-1.

Two tuples are grouped together if the semantic similarity score is above a threshold which is chosen as 0.7 after some experimentation. Now given a tuple with one or two missing fields, our system searches the database and returns tuples with fields matching the ones specified in the query. The results are ranked based on the extent of match between the query and the tuple. We once again make use of synsets to return similar meaning tuples on query.

III. Results

We extracted trustworthy tuples from the AIMed corpus [15]. AIMed is corpus of 1955 sentences from MEDLINE abstracts annotated with gene/protein names and protein-protein interactions. We show here some of the extracted tuples and the results of queries we have made.

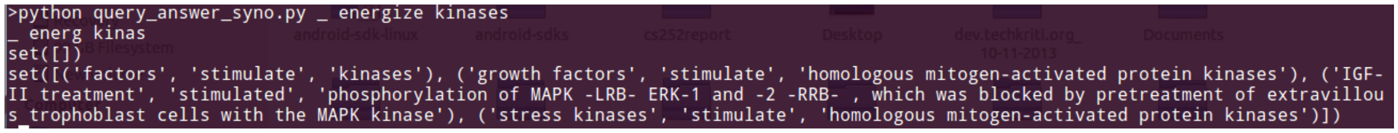
Some of the trusted tuples identified by our classifier:

Interesting ones:

('Injections', 'transmit', 'infections')
('green tea', 'had', 'significant chemopreventive effects')
('zinc deficiency', 'affects', 'cognitive development')
('Exposure to paclitaxel for 8 h or less', 'produced', 'biphasic survival curves')
('a low fat diet', 'reduce', 'DDT bioaccumulation')
('Raman', 'comprised', 'a near infrared laser')
('asthmatic patients', 'contained', 'high levels of IgE antibody')
('poxvirus', 'causes', 'tumors')
('The gastrointestinal tract of numerous animal species', 'contains', 'melatonin')
('osteoblasts', 'secrete', 'the primary endocrine source')
('A58', 'disrupts', 'reverse transcription')
('PPAR alpha agonists', 'repress', 'human fibrinogen gene expression')
('Human coronary artery', 'expressed', 'MCP-1 mRNA')
('dihydropyrimidine dehydrogenase', 'affect', 'tumor sensitivity and resistance')
('the use of dexamethasone', 'reduce', 'delivery of chemotherapeutic agents')
('The beta-lactamase characterized from strain SLO74', 'named', 'OCH-1')

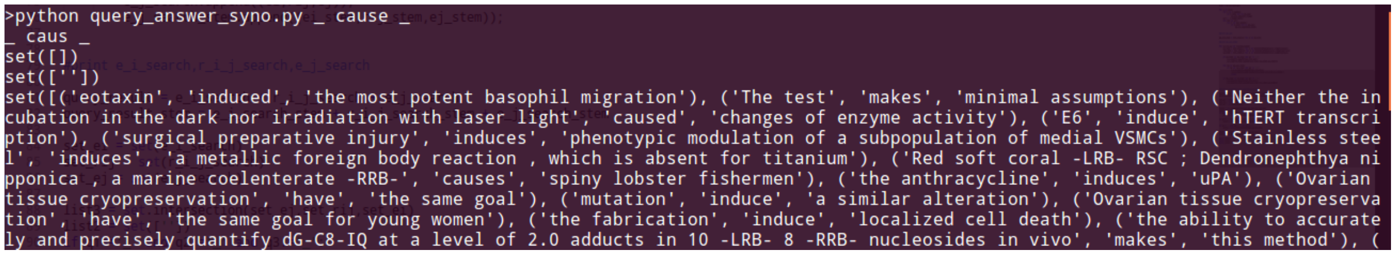
The ones where context matters:

('Mice', 'expressing', 'mutation')
('Mice', 'develop', 'exaggerated inflammation')
('infected persons', 'developed', 'significant clinical consequences')
('cells', 'express', 'detectable levels of cell surface CD34')
('the cells', 'produced', 'VEGF')



```
>python query_answer_syno.py _ energize kinases
_ energ kinas
set([])
set([('factors', 'stimulate', 'kinases'), ('growth factors', 'stimulate', 'homologous mitogen-activated protein kinases'), ('IGF-
II treatment', 'stimulated', 'phosphorylation of MAPK -LRB- ERK-1 and -2 -RRB- , which was blocked by pretreatment of extravillou
s trophoblast cells with the MAPK kinase'), ('stress kinases', 'stimulate', 'homologous mitogen-activated protein kinases')])
```

Fig. 6: Resulted tuples for the query: ***WHAT energize kinases***



```
>python query_answer_syno.py _ cause _
_ caus _
set([])
set([''])
set([('eotaxin', 'induced', 'the most potent basophil migration'), ('The test', 'makes', 'minimal assumptions'), ('Neither the in
cubation in the dark nor irradiation with laser light', 'caused', 'changes of enzyme activity'), ('E6', 'induce', 'hTERT transcri
ption'), ('surgical preparative injury', 'induces', 'phenotypic modulation of a subpopulation of medial VSMCs'), ('Stainless stee
l', 'induces', 'a metallic foreign body reaction , which is absent for titanium'), ('Red soft coral -LRB- RSC ; Dendronephthya ni
pponica , a marine coelenterate -RRB-', 'causes', 'spiny lobster fishermen'), ('the anthracycline', 'induces', 'uPA'), ('Ovarian
tissue cryopreservation', 'have', 'the same goal'), ('mutation', 'induce', 'a similar alteration'), ('Ovarian tissue cryopreserva
tion', 'have', 'the same goal for young women'), ('the fabrication', 'induce', 'localized cell death'), ('the ability to accurate
ly and precisely quantify dG-C8-IQ at a level of 2.0 adducts in 10 -LRB- 8 -RRB- nucleosides in vivo', 'makes', 'this method'), (
```

Fig. 7: Resulted tuples for the query: ***WHAT cause WHAT***

Absurd ones:

('her risk', 'underestimated', '1')
('vivo activation', 'alpha', '14 NKT cells')
('Mutagenesis', 'GR', 'cDNA results')
('the accumulation', 'tissues', '300-1000 to 1')
('the background', 'electrolyte', 'ion sizes')
('older school', 'going', 'children')

Adding context to the tuples is something we decided to incorporate in our future work. In case of the absurd tuples, problem mainly occurred due to wrong POS tags. For example, 'tissues', 'GR', 'alpha', 'electrolyte' are wrongly detected as verbs and '300-1000 to 1', '1' as noun-phrases. Lack of context and incorrect POS tags cover almost 90% uninteresting tuples.

Few example queries and the results obtained can be seen in Fig. 6 and 7. As it can be seen, the query could contain even two missing fields. The results are ranked in best first fashion. In the query ***WHAT energize kinases***, the system understood 'stimulate' is a synonym of 'energize' and hence returned tuples containing 'stimulate' as well. In the query ***WHAT cause WHAT***, the system returned tuples with 'induce', 'makes' and 'have' in the relation field because they are all synonyms of 'cause'.

IV. Future Work

We hope to capture context in the tuples as well. For example, the tuple *(several southern slaves states, declared, their secession)* which is classified as a trustworthy tuple for the input sentence **2** makes sense only in the context of 'the Civil War'. Moreover, expanding our tuples to incorporate a 'location' and 'time' attributes enables the system to learn the spatio-temporal knowledge as well, thus making it more informative.

We can make logical inferences from the tuple database such as:

$$\textit{(Benzo[a]pyrene, is, carcinogen)} \ \& \ \textit{(carcinogens, cause, cancer)} \implies \textit{(Benzo[a]pyrene, causes, cancer)}$$

Eventually one can also build a rich database like YAGO [10], a tree-like structure where, each node is a word and edges are the relation connecting the two nodes. This helps in better data visualization and improved query results.

Our classifier, though very computationally cheap, doesn't perform satisfactorily. So, adding better features to help the classifier in discerning trustworthy tuples from the non-trustworthy ones is something we will be looking into.

Finally, we haven't been able to perform any evaluation due to unavailability of the ground-truth. We hope to crowd-source for some annotated data and perform evaluation.

References

[1] Banko, Michele, et al. "Open Information Extraction from the Web." IJCAI. Vol. 7. 2007.
[2] Fader, Anthony, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2011.
[3] Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430.
[4] Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In LREC 2006.
[5] Jython libraries for Stanford Parser by Viktor Pekar. https://github.com/vpekar/stanford-parser-in-jython
[6] Python implementation of Dijkstras algorithm by David Eppstein UC Irvine, 4 April 2002. http://code.activestate.com/recipes/577343-dijkstras-algorithm-for-shortest-paths/
[7] http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[8] http://en.wikipedia.org/wiki/WordNet
[9] http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement
[10] Hoffart, Johannes, et al. "Yago2: exploring and querying world knowledge in time, space, context, and many languages." Proceedings of the 20th international conference companion on World wide web. ACM, 2011.
[11] Hong-Woo Chun, Yoshimasa Tsuruoka, Jin-Dong Kim, Rie Shiba, Naoki Nagata, Teruyoshi Hishiki, Jun-ichi Tsujii (2006). "Extraction of Gene-Disease Relations from Medline Using Domain Dictionaries and Machine Learning". Pacific Symposium on Biocomputing.
[12] Minlie Huang and Xiaoyan Zhu and Yu Hao and Donald G. Payan and Kunbin Qu and Ming Li (2004). "Discovering patterns to extract protein-protein interactions from full texts". Bioinformatics 20 (18). pp. 36043612

[13] Dmitry Zelenko, Chinatsu Aone, Anthony Richardella (2003)."Kernel Methods for Relation Extraction"

[14] Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang and Mitsuru Ishizuka (2009). "Unsupervised relation extraction by mining Wikipedia texts using information from the web"

[15] ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/ courtesy: Raymond J. Mooney, University of Texas at Austin

[16] Francis, W. Nelson, and Henry Kucera. "Brown corpus manual." Brown University Department of Linguistics (1979).