Towards a Structured Representation of Generic Concepts and Relations in Large Text Corpora

Archana Bhattarai
Department of Computer Science
The University of Memphis
abhattar@memphis.edu

Vasile Rus
Department of Computer Science
The University of Memphis
vrus@memphis.edu

Abstract

Extraction of structured information from text corpora involves identifying entities and the relationship between entities expressed in unstructured text. We propose a novel iterative pattern induction method to extract relation tuples exploiting lexical and shallow syntactic pattern of a sentence. We start with a single pattern to illustrate how the method explores additional paterns and tuples by itself with increasing amount of data. We apply frequency and correlation based filtering and ranking of relation tuples to ensure the correctness of the system. Experimental evaluation compared to other state of the art open extraction systems such as Reverb, textRunner and WOE shows the effectiveness of the proposed system.

1 Introduction

Traditional information extraction methodologies tend to extract a predefined relation between named entities annotated in a different process. While this method might be useful and accurate for smaller data with limited entity types and relations, it cannot scale to extract entities and their relationships in web due to the sheer volume and heterogeneity of data. Thus open domain information extraction systems such as Reverb (Fader et al., 2011), TEXTRUNNER (Yates et al., 2007) and NELL (Carlson et al., 2010) have received added attention in recent times. Extracting machine readable structured information from free text is the basis of most of the semantic analytical systems. With these units of semantic information, a lot of applications requiring semantic information processing such as finding the semantic similarity between two unit of texts, semantic inference, automated question-answering etc can be visualized with better performance.

Existing work on pre-defined relation extraction have implemented methods of supervised, semi-supervised, bootstrapped and unsupervised classification(Zhao and Grishman, 2005), (Kambhatla, 2004) (Bunescu and Mooney, 2006) (Zelenko et al., 2003). For open information extraction methods, since they do not have predefined relations, it is very hard if impossible to generate labeled data for all potential relations in large text corpora. In this paper, we propose an iterative pattern induction based extraction system CREATE (Concept Representation and Extraction through Heterogenous Evidence), to extract relation tuples from large text corpora. We will start with a single selective pattern and iteratively add tuples and patterns in the corresponding collection. This method is easily usable in any domain since it does not require any labeled data. We ensure the selectivity of the pattern by filtering the patterns with statistics such as frequency and average pointwise mutual information (PMI) and specificity of the pattern. CREATE works under the assumption that sentences have a pattern of expressing information and this pattern is followed by multiple sentences. If we can explore these patterns in a language, we can extract tuples from all the sentences to build an automated system. One of the simplest cases of such a pattern is a sentence that only has two nouns and a verb in between. For example, for the sentence *"Google bought Youtube"*, the part-of-speech structure will be "NNP VBD NNP" and hence it is easy to identify two nouns as concepts and the verb as a relation between these two concepts. Thus, the tuple, *bought(Google, Youtube)* can be extracted with high confidence. The beauty of this system is that it gracefully identifies such patterns without requiring any human input and expands itself with the addition of every sentence on the system. The state of the art system that is closest to CREATE in terms of tuple generation is Reverb (Fader et al., 2011). The core idea of

Proceedings of Recent Advances in Natural Language Processing, pages 65–73,
Hissar, Bulgaria, 7-13 September 2013.

Reverb is to identify a relation and extract concepts in the immediate left and right of the relation to form a tuple. The system takes a greedy approach where it only considers concepts that are adjacent to relations. Moreover, they also ignore the information that might change the context of the tuple in the sentence. For example, for the sentence *"RSV in older children and adults causes a cold."*, Reverb extracts tuple *causes(adults, a cold)* with confidence 0.6799. This approach has two disadvantages, first; it extracts invalid tuple as it ignores complete sentence context, second; it misses correct tuple *causes(RSV, cold)* because of its greedy nature. We overcome both the disadvantages in CREATE. Although, Reverb does not require training data to extract tuples, it does require labeled data to determine the confidence of a tuple. CREATE does not require labeled data other than the seed pattern at any stage of the process. With enough iterations and larger corpus, CREATE is able to extract the tuple *causes(RSV, cold)* correctly with high confidence.

Few of the properties that we exploit for the filtering of tuples are as follows:

- Patterns and tuples have dual dependence. Patterns can be used to extract tuples and tuples can be used to identify patterns.
- If a tuple is generated from two different sentences using two different patterns, then the confidence of the tuple is highly increased.
- If a pattern only produces high quality tuples, then the pattern is considered to be of high confidence.
- Web is highly redundant. This redundancy can be exploited to evaluate the correctness of a tuple.

Our approach is to learn the patterns in an iterative manner as in DIPRE (Brin, 1999) and Snowball (Agichtein and Gravano, 2000). We extend the work one step further to iteratively extract tuples with open relations from large text corpora. We follow the standard step of extracting patterns based on known tuples, extracting tuples based on known patterns and evaluating and refinining patterns based on inherent statistics to obtain high precision tuples and patterns.

We make the following contributions in this paper.

- We extend and adapt pattern based tuple extraction to perform open information extraction.
- We propose a method of domain independent pattern generation.
- With the patterns generated in step 2, we propose a method of relation tuple extraction.
- We propose an effective method to refine/rank extracted tuples and patterns without human supervision.

2 Related Work

One of the major goals of open information extraction is to build automated system that can read textual data to a deeper extent compared to bag of words model. Carlson et. al (Carlson et al., 2010) use semi-supervised bootstrapping approach to continuously read and update the knowledge base with an Expectation Maximization like algorithm. Other systems that are tied to a particular structure are (Suchanek et al., 2007), (Auer et al., 2007), (Wu and Weld, 2010) which focus on more structured part of large factual collections such as Wikipedia based on wikipedia-centric properties. The first true open information extraction system TEXTRUNNER, obtained training data applying some heuristics rules over dependency parsing of the training corpus. Using these training samples, sequence based classifiers were trained and more tuples were extracted. The WOE systems (Wu and Weld, 2010) introduced by Wu and Weld make use of Wikipedia as a source of training data for their extractors, which leads to further improvements over TEXTRUNNER (Yates et al., 2007). Wu and Weld also show that dependency parse features result in a dramatic increase in precision and recall over shallow linguistic features, but at the cost of extraction speed. Semisupervised methods start with a few manually provided domain independent extraction patterns that will extract training tuples. Statsnowball works under the principle of iterative pattern and tuple generation using Markov Logic Network (Zhu et al., 2009) and show improved extraction compared to TEXTRUNNER. Reverb (Fader et al., 2011) extracts on simple logic of extracting probable entities/concepts connected with a relation term adjacently. While it does not require seed data or training data to extract relation tuples, it depends on manually analysed data

for the confidence evaluation of a tuple. Unsupervised methods generally exploit the characteristic of the text source, perform deep or shallow parsing and extract the patterns and cluster these patterns to extract relations. Yan et. al. (Yan et al., 2009) used the characteristics of wikipedia and performed clustering of patterns to extract relations without human supervision. They report a precision as high as 84% with deep linguistic parsing. Other works (Syed and Finin, 2010) also use wikipedia for ontology development for entities. (Min et al., 2012) extract relation tuples based on entity similarity graph and pattern similarity. Probabilistic topic based models (Chang et al., 2009) (Yao et al., 2011) have also been used to infer relation between entity-pairs. These models assume relation tuples as atomic observations in documents rather than word observations in standard LDA model.

3 Problem Definition

We formulate the problem of relation tuple extraction as a binary classification problem. Given a sentence $S = (w1; w2; ..; e1..; wj; ..r1; wk..; e2; :::; wn)$ where $e1$ and $e2$ are the entities of interest, $r1$ is the relation of interest, and $w1, w2....wj...wk$ is the context of the tuple in the sentence s, the classification function,

$$f(T(S)) = \begin{cases} 1 & \text{if } e1 \text{ and } e2 \text{ are related by } r1 \\ -1 & \text{otherwise} \end{cases}$$

Here $T(S)$ is a feature set extracted from the sentence as a context. The classification model is built based on context, independent of entities and relations. A context or a pattern of a tuple in a sentence is a 4-tuple $(left, middle\_left, middle\_right, right)$ where $left$ is the sequential list of entities and words that occur before first argument in the tuple, $middle\_left$ is the list of words that occur between first argument and relation, $middle\_right$ is the list of words that occur between relation and second argument and $right$ is the list of words that occur after second argument in the sentence unless another relation is detected.

The classification function $f(T(S)) = 1$ if the pattern of the tuple T in the sentence S exists in pattern database.the degree of similarity of the context of probable tuple is greater than threshold similarity with one of the contexts existing in context-base.

4 Create Tuple/Pattern Extraction Methodology

Given a set of documents containing sentences, our goal is to extract relation tuples with highest recall and precision. As explained earlier, our system is designed to utilize the dual dependence of tuple with pattern and pattern with tuple. As a starting point, we use a seed pattern $p = (\phi, \phi, \phi, \phi)$ that will generate tuples from text corpus. These tuples are then used to generate extraction patterns which in turn generate more tuples just like in Snowball. All the extracted tuples and patterns in the process are not guaranteed to be correct. A good tuple should be syntactically and semantically correct as well as articulate, autonomous and informative. Similarly, a good pattern should achieve a good balance between two competitive criteria; specificity and coverage. Specificity means the pattern is able to identify high-quality relation tuples; while coverage means the pattern can identify a statistically non-trivial number of good relation tuples. Hence, in the process, we have a self evaluating system which evaluates and filters out invalid tuples and patterns based on their statistical properties. The overall system can be broken down into several modules, each of which perform an isolated task such as concept extraction, relation extraction, probable tuple generation, tuple verification etc. The system architecture of the overall system has been depicted in figure 1 and the algorithm is shown in Table 1. The sub-modules are explained in detail in the subsequent sub-sections.

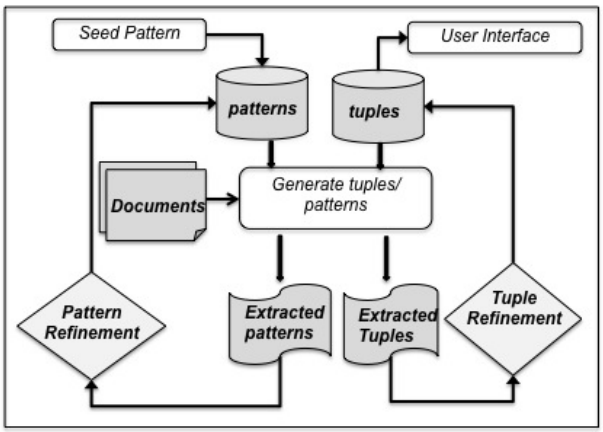

Figure 1: Overall System Architecture

Feature:We consider lexical and shallow parse information as features for relation extraction.

Lexical and shallow NLP techniques are robust and fast enough for a problem like ours where extraction needs to be performed at web scale. Although, our concept extraction module can be easily replaced with named entity extractor, we primarily use part-of-speech tagging and chunking results for concept/relation extraction. All the sentences in our data sets are parsed using a opennlp (Baldridge et al., 2004) part-of-speech tagger.

Seed Pattern: We start with a fairly general and yet very strict pattern that will extract tuples from a sentence. The seed pattern, $p_s = \{\phi, \phi, \phi, \phi\}$ meaning there is an empty left context, empty middle left context, empty middle right context and empty right context. As an example, let us consider a sentence "Temperature is ultimately regulated in the hypothalamus", our process extracts two concepts "Temperature" and "the hypothalamus" and relation "is ultimately regulated in". The left context (context before concept 1) in this case is empty, middle left context (context between concept 1 and relation) is also empty and similarly, middle right and right contexts are empty. This is a fairly specific pattern for a tuple to be valid and moreover, this pattern is domain independent and can be applied to any domain for english language. We have a running example showing the steps in table 2.

Concept Extraction Module: We extract concepts in the sentence based on noun phrases. We remove starting and trailing stopwords in noun phrases. If noun phrases contain conjunction, we break down noun phrase into two concepts.

Relation Extraction Module: To extract relations, we extract the longest sequence of words such that it starts with verb or is a sequence of noun, adjective, adverb, pronoun and determiner or a sequence of preposition, particle and infinitve marker. If any pair of matches are adjacent or overlap in a sentence, we merge them to a single relation. This method has been proven to be effective in (Fader et al., 2011).

Probable Tuple Extraction: For each relation $r \in R$ and for every combination of $c_i and c_j \in C$, such that c_i occurs before r and no other relation occurs between $c_i \; and \; r$ and c_j occurs after r and no other relation occurs between $c_j \; and \; r$ in the sentence, we create a probable tuple $t = (c_i, r, c_j)$.

Tuple Pattern Extraction: For each tuple $t = (c_i, r, c_j)$ in sentence s, we extract the sequence of words in sentence that occurs between beginning of sentence and concept c_i. If a relation occurs before c_i, we start with the end of closest relation. This is the left context. Similarly we extract middle_left context as the sequence of words between c_i and relation r. Middle_right context is the sequence of words between relation r and c_j. Right context is the sequence of words between c_j and either another relation r_p (if exists) or end of the sentence. We experiment with three types of patterns, first: purely lexical(only use lexicons for pattern generation), second: purely syntactic (only use part of speech tags for pattern generation) and third: mixed pattern(a combination of lexicons and part of speech tags. For mixed pattern, we replace all nouns, verbs, adjectives and adverbs with their part of speech tags and leave preposition, particle and other words to use lexicons.

Iteration: Our system is an iterative process and gets better qualitatively and quantitatively with each iteration. The number of iteration is highly dependent on the application of interest, pattern database size, size of corpus and time sensitivity of the system. We experimented on a smaller sample of data to see the convergence of the algorithm. We also iterated over a large corpus to see the effect of iteration on number of patterns and tuples. Since the extraction algorithm is based in active learning methodology, the system can perform quite well with iteration count as small as 2 in large corpus.

Algorithm 1 Iterative Pattern Induction

Input: $Pattern, P = \{seed\_pattern\}$,
$Tuples, T = \{\phi\}$
$Sentences, S = \{s_1, s_2, ...s_n\}$

Output: $Patterns, P = \{p_1, p_2, ...p_x\}$,
$Tuples, T = \{t_1, t_2, t_3.....t_y\}$

for every $S_i \in S$ **do**
  $C_{prob} = \{c_1, c_2, ..c_j\} \leftarrow extractConcepts(S_i)$
  $R_{prob} = \{r_1, r_2...r_h\} \leftarrow extractRelations(S_i)$
  $p_{sent} = replaceConceptsRelations(C_{prob}, R_{prob})$
  $T_{prob} = \{t_1, ..t_u\} \leftarrow extractProbableTuples(C_{prob}, R_{prob})$
end for
for every $t_j \in T_{prob}$ **do**
  $pattern, p_i = extractPatternFor(S_i, p_s)$
if $p_i \in P \;\&\&\; t_j \notin T$
  $T.add(t_j), P.update(p_i)$
else if $p_i \notin P \;\&\&\; t_j \in T$
  $P.add(p_i), T.update(t_j)$
else if $p_i \in P \;\&\&\; t_j \in T$
  $P.update(p_i), T.update(t_j)$
end if
end for

Table 1: Iterative Pattern Induction Algorithm

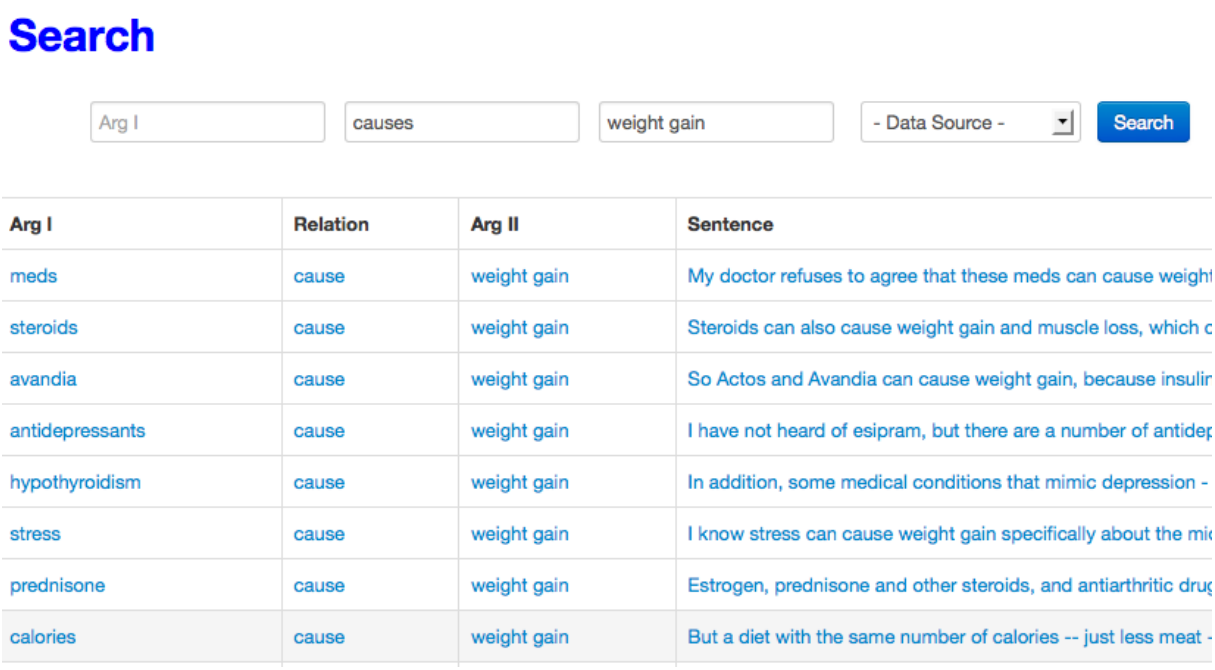

Figure 2: Concept based Search User Interface

Parameter	Value
seed pattern	$(\phi, \phi, \phi, .)$
sentence	Sunscreen may also cause drying of skin.
concepts	Concept1=Sunscreen, Concept2=skin
relations	relation=may also cause drying of
sentence pattern	Concept1 relation Concept2.
probable tuple	may_also_cause_drying_of(sunscreen, skin)

Table 2: Running Example of Tuple and Pattern Extraction

5 Tuple Refinement

5.1 Tuple and Pattern Filtering

We employ a holistic approach for concepts and relations extraction that enforces coherence in relations and concepts in tuples . To ensure validity of extracted tuples, we select patterns and tuples that occur more than α (3 in our experiments) and β (2 for medical and 1 for wikipedia for our experiments) times respectively. Also, total frequency of a pattern p in a relation r is defined as the sum of the frequencies of p in all entity pairs that have relation r. We define confidence of a tuple as follows:

$$Conf(t) = \frac{\sum_{p \in P_t} f(p_i)}{f(p_{max_t}) log(N)} \quad (1)$$

where $f(p_i)$ is the frequency of pattern p_i for relation r such that tuple t also has relation r. Here, $f(p_{max_t})$ is the frequency of pattern that has maximum frequency for relation r and N is the total number of distinct patterns that match tuple t. Note here that confidence conf(t) can be greater than 1 depending on the number of patterns that extract tuple t.

5.2 Tuple relevance

Traditional vector space model based relevance cannot be applied to concept based relevance paradigm. Hence we employ PMI based relevance for tuple retrieval. If e1 is the query entity for which search is executed, then the relevance of a tuple is calculated in terms of PMI between query entity e1 and second argument in tuple that contains e1 as first argument. PMI between entities e1 and e2 is defined as

$$PMI(e_1, e_2) = log \frac{P(e_1, e_2)}{P(e_1, e)P(e_2, e)} = logN \frac{n_{12}}{n_1.n_2} \quad (2)$$

$$NPMI(e_1, e_2) = \frac{PMI(e_1, e_2)}{-logP(e_1, e_2)} \quad (3)$$

where N: the total number of tuples in the corpus, $P(e_1, e_2) = n_{12}/N$=the number of sentences containing tuples that have e_1 and e_2 as

arguments, $P(e_1, e) = n_1/N$: the probability that the entity e_1 cooccurs with entity e in tuples, $P(e_2, e) = n_2/N$: the probability that the entity e_2 cooccurs with entity e in tuples.

6 Prototype and Experiments

6.1 System Prototype

We built the system prototype based on the process explained in this paper for two datasets, namely; wikipedia and medical sites. We crawled 10 medical information sites and collected sentences talking about medicine. The prototype provides a tuple searching interface and a concept-graph based navigation system. We demonstrate the usefulness of the system with medical information and evaluate against few relations in wikipedia. Figure 2 shows a snapshot of the prototype for medical data for another example.

6.2 Comparison with Open Information Extraction Systems

We compared the result of our system with other systems such as Reverb, TextRunner and WOE. For evaluation purpose, we used the test set of 500 sentences used in Reverb system evaluation(Fader et al., 2011). The figures shows the quantitative comparison of our system compared to reverb and woe. It has to be noted however that this result does not evaluate the iterative process of create. The distinctive advantage of create is seen when applied to a relatively larger corpus where the system is applied iteratively.

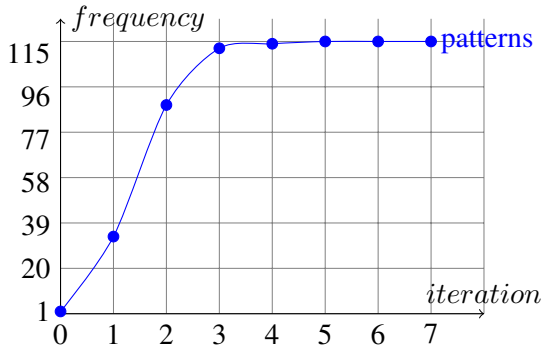

Figure 3: Effect of Iteration on Number of patterns

Figure 3 and figure 4 show the effect of iteration with the CREATE algorithm. It shows that in initial iterations, there is a rapid increase in number of patterns and tuples. However it starts to converge with higher iterations. For proof of concept,

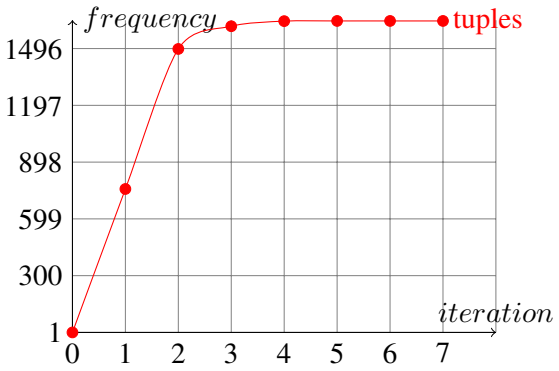

Figure 4: Effect of Iteration on Number of tuples

we experimented with a sample data that we created with medical sentences. It shows that tuple and pattern generation converges in 5 iterations.

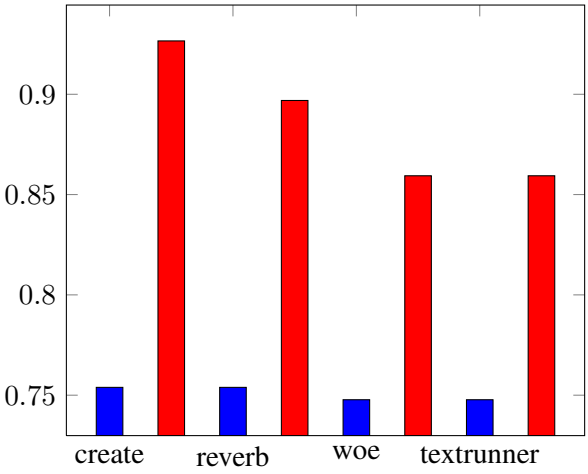

Figure 5: Comparison of CREATE performance with Reverb, WOE and TextRunner

Figure 5 shows the comparison of CREATE with Reverbm WOE and TextRunner. We see improved recall at around 92% and precision around 75% for create which outperforms all other systems. Similarly, figure 6 shows the effect of iteration on the performance of CREATE system. We see the same effect of rapid increase in performance in initial iterations and then it gets stabilized after few iterations.

We also experimented with the performance based on different patterns. Figure 7 shows that recall for POS pattern is the highest but the precision is highest with mixed pattern.

6.3 Wikipedia Tuple Extraction

We used Semantically Annotated Snapshot of the English Wikipedia (Atserias et al., 2008) to extract

Relation	Gold Data	Create (total/correct)	Precision	Recall
bornIn(x,Atlanta)	440	341/303	88.8	68.8
bornIn(x,Zurich)	108	87/75	86.23	69.4
graduatedFrom(x,Stanford)	456	403/345	85.6	75.6
graduatedFrom(x,Princeton)	582	464/385	82.9	66.1
presidentOf(x,United States)	44	65/39	60	88.86

Table 3: Data statistics for wikipedia.

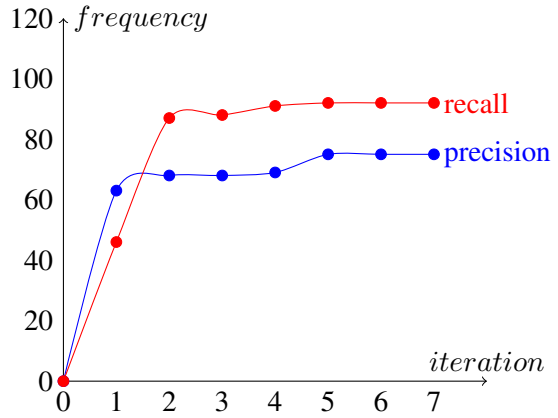

Figure 6: Effect of Iteration on Tuple Extraction Performance with confidence 0.6

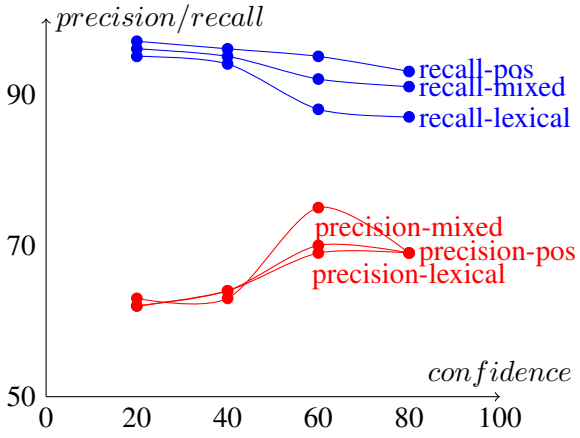

Figure 7: Precision/ Recall variance with Confidence

relation tuples as the first large dataset. The SW1 corpus is a snapshot of the English Wikipedia dated from 2006-11-04 processed with a number of public- available NLP tools. We chose to use this data as it has been processed and has information on shallow parsing such as POS tags and named entities on seven categories. To demonstrate the interchangeability of concept extraction module , we used the named entities as concepts

Data	Wikipedia	Medical
Document count	1431178	348284
Sentence count	36117170	4049238
Tuple count	6945440	1535293
Relation count	1847116	706359
Relation with freq > 9	1131	1865
Concept count	2673192	106263
Extraction latency (for single iteration)	5 hrs	2hrs

Table 4: Data Statistics.

for relation extraction. We then generated tuples from data. Since it is not possible to evaluate all the relation tuples extracted from wikipedia, we performed samples evaluation of the system for few sampled relations and tuples. We compared the performance of our system based on precision and recall compared to Dbpedia. The evaluation in terms of precision and recall is shown in Table 4. Precision and recall are given by the following equations

$$precision = \frac{|(correct\ docs) \bigcap (retrieved\ docs)|}{|(retrieved\ docs)|} \quad (4)$$

$$recall = \frac{|(correct\ docs) \bigcap (retrieved\ docs)|}{|(relevant\ docs)|} \quad (5)$$

7 Conclusion

We have qualitatively and quantitavely demonstrated the effectiveness and usefullness of our system and overall relation extraction systems. With increasng data being available, the value and importance of systems such as CREATE is ever increasing. We have demonstrated the prospects of relation extraction systems. At the same, we also need to be aware of the challenges that need to be solved before we can realize a fully functional machine reading system.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM.

Jordi Atserias, Hugo Zaragoza, Massimiliano Ciaramita, and Giuseppe Attardi. 2008. Semantically annotated snapshot of the english wikipedia. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Jason Baldridge, Tom Morton, and Gann Bierner. 2004. The opennlp maxent package in java. *URL: http://maxent. sourceforge. net*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2009. *Open information extraction for the web*. University of Washington.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183. Springer.

Razvan Bunescu and Raymond Mooney. 2006. Subsequence kernels for relation extraction. *Advances in neural information processing systems*, 18:171.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, volume 2, pages 3–3.

Jonathan Chang, Jordan Boyd-Graber, and David M Blei. 2009. Connections between the lines: augmenting social networks with text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

Nilesh Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, Sathiya Keerthi, and Srujana Merugu. 2009. A web of concepts. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12. ACM.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Bonan Min, Shuming Shi, Ralph Grishman, and Chin-Yew Lin. 2012. Towards large-scale unsupervised relation extraction from the web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8(3):1–23.

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*, pages 697–706. ACM.

Zareen Syed and Tim Finin. 2010. Unsupervised techniques for discovering ontology elements from wikipedia article links. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 78–86. Association for Computational Linguistics.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1021–1029. Association for Computational Linguistics.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466. Association for Computational Linguistics.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 419–426. Association for Computational Linguistics.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th international conference on World wide web*, pages 101–110. ACM.