EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article ICST.ORG

# Knowledge Extraction Framework for Building a Large-scale Knowledge Base

Haklae Kim[1,*], Liang He[2] and Ying Di[2]

[1] Samsung Electronics Co., Ltd., (Maetan dong) 129, Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, 16677, Korea
[2] Samsung Electronics (China) R&D Center, Chuqiaocheng, 57# Andemen Street, Yuhuatai Distict, Nanjing, Jiangsu, PRC 210012

## Abstract

As the Web has already permeated to life styles of human beings, people tend to consume more data in online spaces, and to exchange their behaviours among others. Simultaneously, various intelligent services are available for us such as virtual assistants, semantic search and intelligent recommendation. Most of these services have their own knowledge bases, however, constructing a knowledge base has a lot of different technical issues.
In this paper, we propose a knowledge extraction framework, which comprises of several extraction components for processing various data formats such as metadata and web tables on web documents. Thus, this framework can be used for extracting a set of knowledge entities from large-scale web documents. Most of existing methods and tools tend to concentrate on obtaining knowledge from a specific format. Compared to them, this framework enables to handle various formats, and simultaneously extracted entities are interlinked to a knowledge base by automatic semantic matching. We will describe detailed features of each extractor and will provide some evaluation of them.

**Keywords:** Knowledge base; knowledge extraction, knowledge graph.

Received on 13 January, 2016, accepted on 8 March, 2016, published on 21 April, 2016



doi: 10.4108/eai.21-4-2016.151157

## 1. Introduction

Recently, mobile devices are now equipped with intelligent virtual assistant, which is an application program that can understand natural language and complete electronic tasks for users [1]. Most of major IT companies such as Apple, Google, Microsoft and Facebook have invested intelligent services that is to make assistants more contextually aware and more versatile [1][2].
In this background, knowledge is crucial for virtual assistant, because they search and discover a set of knowledge bases to handle their enquiry. However, constructing a knowledge base from various data sources is not trivial. For example, the following sentence provides a brief description about *Lionel Messi*.

*"Lionel Messi(born 24 June 1987), is an Argentine professional footballer who plays as a forward for Spanish club FC Barcelona and captains the Argentina national team."*

Although human can understand unstructured texts with its context, meaningful information on these texts should be transformed into structural formats for allowing machines to understand it. For example, *Lionel Messi* is *Person*, who has *date of birth* (i.e. 24 Jun, 1987) and his *nationality* is *Argentina*, which is a type of *country*. Simultaneously, we can deduce a relation that his *occupation* is a *football player*.
A knowledge base comprises a set of objects as entities and their various relations. Although enormous data on the Web has been expanded, and is crucial sources for

*Corresponding author. Email:haklaekim@gmail.com

constructing a knowledge base, extracting knowledge from large-scale data sources have still many technical issues such as named entity recognition, relation extraction, entity resolution, and co-reference resolution [3]. Furthermore, recent knowledge bases tend to a graph structure of inter-related objects [4], such as DBpedia[†], Wikidata[‡], YAGO[§] and Google's Knowledge Graph[**]. All extracted data is transformed into interlinked data formats based on a specific ontology model. On the other hand, objects among different knowledge bases can be inter-twined, in this sense, entity interlinking and ontology matching are essential techniques for building a graph-based knowledge base.

In this paper, we introduce a knowledge extraction framework for building a knowledge base from large-scale data sources. Although existing knowledge bases contain millions of facts about the world, they have their own emphasis points. We focus on a general framework to aggregate and extract large-scale entities and relations from heterogeneous data sources. The contribution of this paper is as follows:

- We propose a comprehensive framework for knowledge extraction with its architecture in details.
- This framework offers consistent knowledge extraction for various data formats, and enables to support simultaneous extraction processing from data sources. Because most of existing tools concentrate on a specific format of data sources, they do not have effective to extract various formats. Our approach handles multiple formats simultaneously and reduces a processing time.
- Finally, extraction performances of the framework and its components are competitive, compared to existing tools.

The remainder of this paper is organized as follows: Section 2 describes previous studies, including knowledge base and text extraction techniques. Section 3 addresses details of knowledge extraction framework, which contains several extraction modules such as unstructured data and web tables. Section 4 describes some results of experiments of the extraction framework. Finally, we conclude our work with further research and development topics in Section 5.

## 2. Related work

There are many works on automatic knowledge base construction. WikiData is a free knowledge base about the world that can be read and edit by humans and machines alike [5][6]. Things described in the Wikidata knowledge base are called items and can have labels, descriptions and aliases in all languages, which does not aim at offering a single truth about things, but providing statements given in a particular context.

DBpedia is a community-based knowledge base, which extracts structured, multilingual knowledge from Wikipedia and makes it freely available on the Web using Semantic Web and Linked Data technologies [7]. Since it covers a wide variety of topics and sets of RDF links pointing to various external data sources, it has developed into the central interlinking hub in the Web of Linked Data.

One of the projects that pursue similar goals to DBpedia is YAGO [8]. YAGO has developed to version 3 as YAGO3, an extension of the YAGO knowledge base that combines the information from the Wikipedia in multiple languages [9]. It fuses multilingual knowledge with English WordNet to build a coherent knowledge base. The main differences between YAGO and DBpedia include: one is that the DBpedia ontology is manually maintained, while the YAGO ontology is backed by WordNet and Wikipedia leaf categories. The other is that the integration of attributes and objects in infoboxes is done via mappings in DBpedia, while the YAGO implements it by expert-designed declarative rules.

The Open IE project has been developing a Web-scale information extraction system that reads arbitrary text from any domain on the Web, extracts meaningful information and stores in a unified knowledge base for efficient querying [10]. This project generates tools including TextRunner, Reverb [10], Ollie [11] and OpenIE4.0. OpenIE is schema-less, and can extract data for arbitrary relations from plain text, which is useful when lacking of seed data and training corpus comparing to the distant supervising method used in our Text2K extractor. But extracting noisy data is the main obsession of it.

Knowledge Vault [12] is developed by Google to extract facts, in the form of disambiguated triples, from the entire web. The main difference from other works is that it fuses together facts extracted from text with prior knowledge derived from Freebase [13]. Our work is most similar with this approach, except adding confidence value to every fact. But we also integrate Wikipedia data, which contains multilingual and abundant structured information. It can enrich our knowledge base with more data.

Current knowledge bases can be clustered into 4 main groups [10][12]: 1) approaches such as WikiData, DBpedia, and YAGO, which is built on Wikipedia infoboxes and other structured data sources; 2) approaches such as Reverb, OLLIE, and PRISMATIC, which use open information (schema-less) extraction techniques applied to the entire web; 3) approaches such as PROSPERA, Knowledge Vault, which extract information from the entire web, but use a fixed ontology and schema; and 4) approaches such as Probase [14],

[†] http://dbpedia.org
[‡] http://wikidata.org
[§] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago//
[**] https://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html

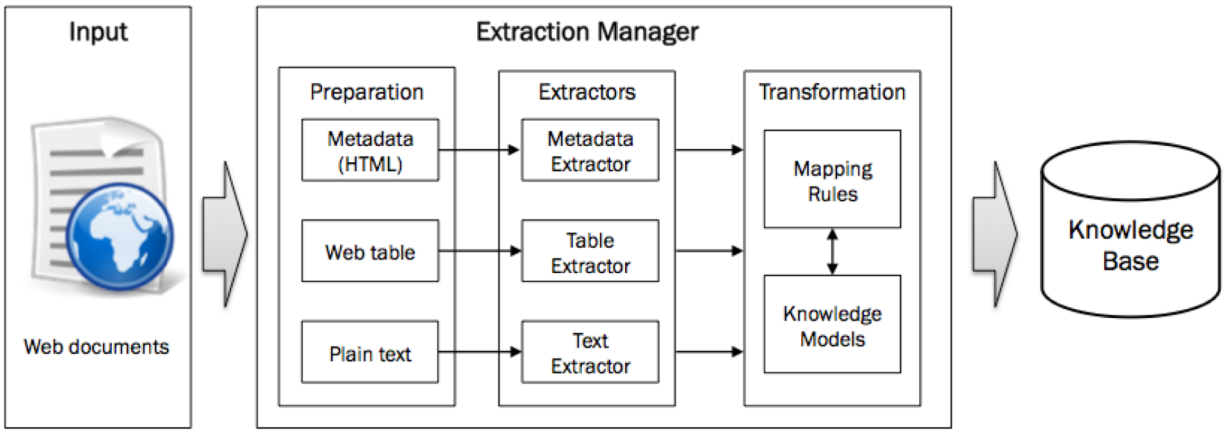


**Figure 1 Architecture of the knowledge extraction framework**

which construct taxonomies (is-a hierarchies), as opposed to general knowledge bases with multiple types of predicates. Our extraction framework covers both 1) and 3), and merges all the extracted data to provide an enriched unified knowledge base.

Web table extraction is a hotspot in the last decade; yet, limited studies have been published out. Munoz, E. et el. open a tool, DRETa [15], to extract RDF triples from generic Wikipedia tables by using DBpedia as a reference knowledge base. However, DRETa is only served for Wikipedia tables and strongly relies on DBpedia. Limaye, G. et al. [16] propose a machine learning techniques to annotate table's cells with entity, type and relation information. It focuses on table annotation regardless of detecting tables from web pages.

## 3. Knowledge Extraction Framework for Large-scale Web Data

This framework is to extract a set of meaningful entities and its values from a large-scale web data. Figure 1 illustrates a high-level architecture of the knowledge extraction framework, which comprises of three types of extractors including metadata, web tables and plain texts in HTML pages. Input formats are a collection of URL or large-scale data dumps such as Wikipedia or crawled web data, and each extractor depended on data formats handles extraction procedures with different processing logics. For example, a web table is obtained through detecting the HTML anchors indicating table chunks, such as <table>, <class="wikitable">, or is extracted by training a table detection model using machine learning techniques (see Section 3.2 in detail). After this extraction by individual extractor, they are transformed structured data by using knowledge model that is already defined for representing a knowledge base, and is stored or updated into a triple store. Each extraction techniques and features are described in the following section.

### 3.1 Metadata

The metadata extraction is to collect knowledge (e.g. products, people, organizations, places, events and resumes) from markup standards such as RDFa [17], micro-data [18] and micro-formats [19] of web documents. This extractor contains two core modules; one is markup filtering, and the other is a specific extraction rule for data formats. A markup filtering is to detect a set of markups from HTML sources. Because a number of web pages do not utilize markup standards, a set of rules for detecting corresponding tags can be used for analyzing a specific part of web pages. Furthermore, it can reduce time-consuming tasks by aggregating a fixed quantity of web pages. When a web page does not match a filtering rule, the rest of this source is not collected and simultaneously is not expanded any other data sources from the page. Currently, extraction rules support several markup formats as follows:

- RDF/XML, Turtle, Notation 3
- RDFa with RDFa1.1 prefix mechanism
- Microformats: Adr, Geo, hCalendar, hCard, hListing, hResume, hReview, License, XFN and Species
- HTML5 Microdata: (such as Schema.org)

Its implementation is based on the Anything To Triples (Any23)[††], which is a library for extracting structured data in RDF format from a variety of Web documents. After extracting data, it is transformed and interlinked to a knowledge base using some existing vocabularies, such as schema.org[‡‡], purl.org[§§] and ogp.me[***].

[††] https://any23.apache.org/
[‡‡] http://schema.org
[§§] http://purl.org
[***] http://ogp.me

## 3.2 Web tables

At present, millions of data are represented by web table formats on the Web. In particular, data on this format can be useful sources for enriching a knowledge base, because it has some structures using specific tags. We propose an approach to extract entities and relations from web tables based on a reference knowledge base.

**Table 1 Selected features of web tables**

| Feature | Description |
|---|---|
| c | Average number of columns |
| dC | Standard deviation of number of columns |
| r | Average number of rows |
| dR | Standard deviation of number of rows |
| cl | Average overall cell length |
| dCL | Standard deviation of cell length |
| CLC | Average cumulative length consistency |
| CTC | Average content type consistency |

The approach consists of two stages: table detection for identifying tables from web pages, and triple transformation for generating triples from web tables. Table detection is to identify tables of visual formatting from web pages, and then to screen out those with relational information. Besides, it transforms relational tables from HTLM into a unified format where each row represents entity relations. Hence, we propose an improved approach based on the work of Yalin and Hu [20] to detect tables, including the follow parts:

(i) Table wrapper is to embed a table of visual formatting in a HTML document. Generally, most of web tables are embedded in <table></table>, yet, part of them are in <ul></ul>, <li></li>, etc. Moreover, different embedment possesses its own table structure definition. Therefore, to detect web tables as much as possible, machine learning technique, particularly, Decision Tree is employed.

(ii) Feature selection is a crucial step for our machine learning based method. Table 1 describes the features we selected.

(iii) Relational Table Classifier: For table detection task, we propose to use Decision Tree on our table training set of feature vectors with true/false table labels to obtain a relational table classifier.

(iv) Special Case Pre-processing: Above 3 steps illustrate a general processing method for relational table detection. However, for several websites, they define their own HTML anchors to describe tables, such as <table> and <class="wikitable">. Therefore, a mapping list between websites and tags for relational tables is predefined before table detection, which can reduce web table detection time, including table wrapper parsing and table feature calculation.

## 3.3 Unstructured and plain text

Although many metadata standards are existed, most of web documents contain unstructured and plain data. A various research efforts and open source tools have been introduced in this area [10][11][12][13]. However, most of them have some limitations recognizing exact entities and values from unstructured data. Let us consider the following example:

*Douglas Noel Adams was an English writer. He is a humorist, and dramatist.*

Using traditional approaches and tools, some facts are extracted: Douglas Noel Adams is a Person type from first sentence. However, various types can be extracted, because "He" in second sentence refers to the same person, and it can be deduced that Douglas Noel Adams is humorist and Douglas Noel Adams is dramatist.
To extract entities and its relations from plain texts, various approaches are applied to the extraction framework. As illustrated in Figure 2, there are four main components.

### 3.3.1 Pattern training and Pre-processing

As the first phase, pattern training is based on knowledge seed and training corpus. A semi-supervised learning algorithm is employed for this task, which makes use of a weakly labeled training set. It has the following steps: 1) It may have some labeled training data (relations seed data), 2) It "has" access to a pool of unlabeled data (training corpus), 3) It has an operator that allows it to sample from this unlabeled data and label them. This operator is expected to be noisy in its labels, 4) The algorithm then collectively utilizes the original labeled training data if it had and this new noisily labeled data to give the final output (trained patterns). Table 2 shows a brief example for pattern training: 1) a seed is a *movie:writer (movie, writer)* pair, relation is *movie:writer*. The first item is a movie name, and the second one is a writer of the movie, 2) from the training corpus, co-occurrence corpus is extracted, 3) it is transformed into uniform formats for obtaining a set of original patterns, and 4) generate final pattern by doing stemming and replace operation to original patterns.

**Table 2 Pattern training examples**

| Step | Task | Example |
|---|---|---|
| 1 | Seed | (Mirele Efros, Jacob Gordin) |
| 2 | Co-occurrence corpus | Mirele Efros was an 1898 Yiddish play by Jacob Gordin. |
| 3 | Original Pattern | $ARG1 was an 1898 Yiddish play by $ARG2 |
| 4 | Final Pattern | $ARG1 be a * play by $ARG2 |

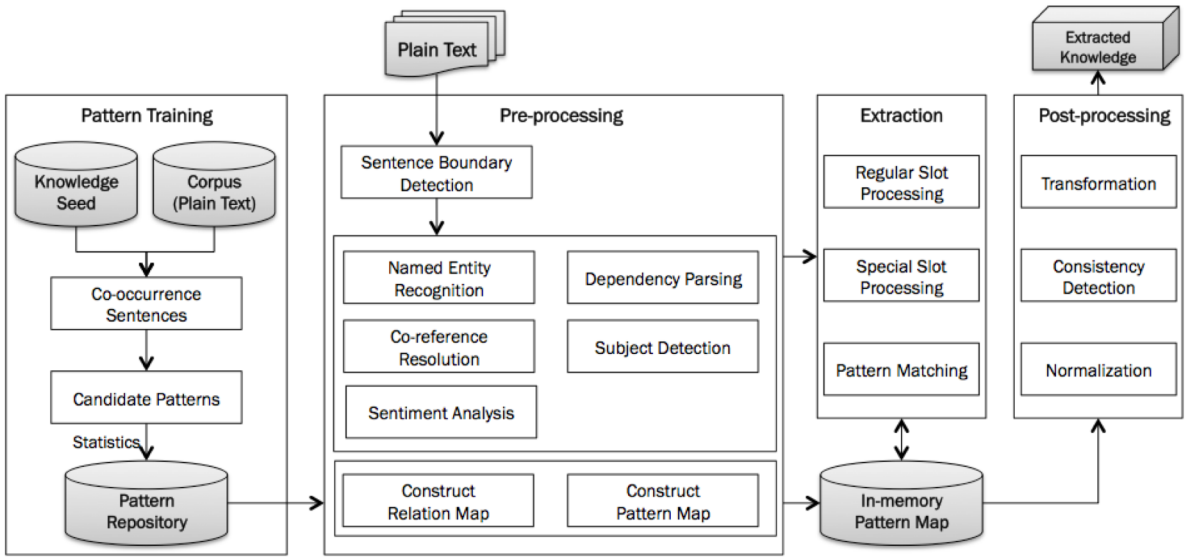


**Figure 2 A high-level workflow of unstructured data extraction**

Pre-processing phase mainly contains several tasks. Sentence boundary detection is to get each sentence from text, which is the smallest unit for extraction. Stemming is helpful to correctly extract between different tenses and forms efficiently. Named Entity Recognition (NER) is the key step, which recognizes named entities, as a subject or an object, in a sentence [22]. Parsing is used to find the syntax structure of a sentence [23][24]. We do parsing for subject identification of sentence. Co-reference resolution helps to find corresponding entity when a subject is not a named entity [25]. Using head rules defined in Bikel [21] operating on parsing result to get the subject of a sentence. Sentiment analysis supports extra information about entity emotional trend. In particular, we construct domain-dependent corpora such as movies, music and celebrities, and use general-purpose semantic knowledge bases for extracting and detecting sentimental information.

### 3.3.2 Extraction and Post-processing

Extraction realizes the goal of obtaining relation information between subject entity and object entity in the same sentence.

Extraction is executed for each sentence, and to figure out the relation between the subject entity S and other recognized entities E= $\{E_1, E_2 \ldots E_n\}$ in the sentence through patterns matching. Assuming all the relations are R= $\{r_1, r_2 \ldots r_m\}$, and the patterns are P= $\{p_1, p_2 \ldots p_m\}$, here every $p_i$ is a group of patterns representing the same relation $r_i$. What we do here is to find out $r_j$ for each S and $E_i$ pairs, and the detail steps are as below:

frag(S, $E_i$) gets the text fragment between the occurrences of S and $E_i$ from the original sentence. pattern_match(text, pattern) checks if the text conforms to the pattern.

Post-processing mainly focuses on normalization and transformation. Normalization is to turn data with the same type into unique format such as date type information "May 11 2001"; "1586.01.02" need changed into "5/11/2001" and "1/2/1586". Transformation is to transform data into triple format refer to the designed ontology.

Table 3 Extraction Algorithm

```
for (E_i in E):
  for (r_j in R):
    if (type(S) == type(r_j's subject) &&
      type (E_i) == type(r_j's object)){
      for (p_j^k in p_j):
        text_fragment=frag(S, E_i);
        match = pattern_match (text_fragment, p_j^k);
        if (match is true):
          add_triple (S, r_j, E_i);
```

### 3.3.3 Optimization

Throughout the entire processes for text extraction, NER, parsing and co-reference resolution modules cost almost 90~95% processing time (see Figure 4). Thus, we focus on our optimization for these tasks. We employ some optimization strategies compared with traditional relation extraction systems.

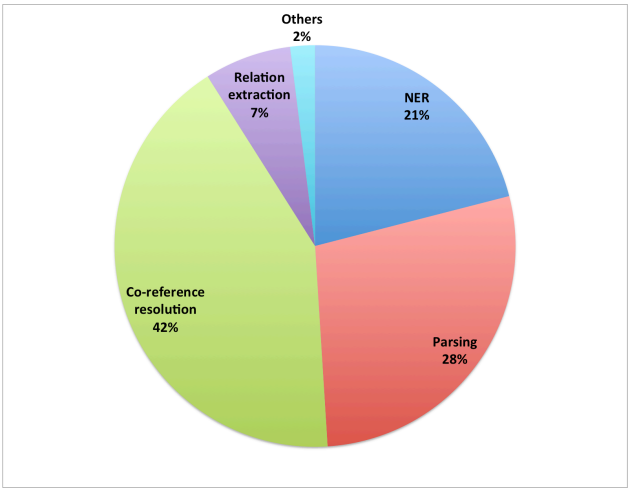


**Figure 3 A processing time of major tasks for text extraction**

As the original generated patterns may include varieties of entities such as date and country, pattern abstraction is necessary for gaining final patterns through the post-processing task. For example, a set of named entities is replaced by wildcard character (e.g. "1898" to "*"), and also some patterns are replaced by pattern stemming (e.g. "was" to "be").

We use Shift/Reduce mode instead of PCFG model. As shown in Table 4, Shift/Reduce reaches a better F1 measure while with much less time needed on parsing.

Co-reference resolution helps to find corresponding entity when a subject is not a named entity. Considering more than 88% co-reference cases occur in same paragraph [26], for pronoun, even more than 95%, our co-reference resolution unit focus on paragraph rather than whole article.

**Table 4 PCFG vs. Shift/Reduce**

| Model | Parsing (s) | F1 |
|---|---|---|
| PCFG | 426 | 85.54 |
| Shift/Reduce | 14 | 85.99 |

# 4. Experiments

## 4.1 Table extraction

To evaluate our proposed web table extraction method, we test it on 1200 Wikipedia HTML pages containing 2097 tables on local Linux PC (i7-3770 CPU and 6G memory). The following tables describe consumed extraction time based on web page and web table:

**Table 5 Extraction time per a web page**

| Stage | Average Consumed Time (ms) |
|---|---|
| Table Detection | 11.62 |
| Table Triplification | 3046.68 |
| Total | 3058.48 |

Table 5 shows each webpage's table extraction situation. Extraction task contains table detection and triplification. Since each page may have more than one table, we analyse each table extraction performance and the result is listed in Table 6.

**Table 6 Extraction time per a web table**

| Stage | Average Consumed Time (ms) |
|---|---|
| Table Detection | 6.65 |
| Table Triplification | 1743.45 |
| Total | 1750.20 |

According to the above tables, it is observed that most of time is spent on the table triplification (about 99%), since lots of retrieval for candidate entities and potential relations are included. Figure 4 compares the performance of table extraction speed among our approach, DRETa and annotating approach, which is proposed by Limaye, G. et al. [26]. Our approach uses 1.75s, which shows a better performance than DRETa using 7.28s, while slower than the annotating approach (0.7s), which is lack of table detection procedure.

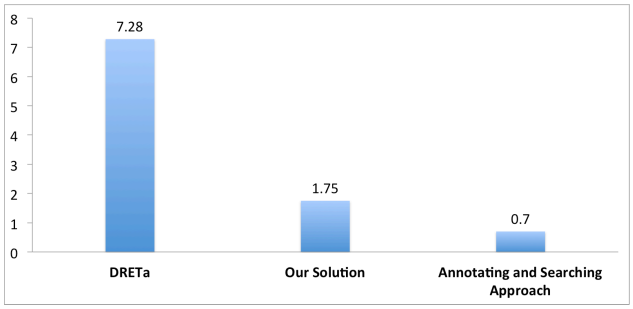


**Figure 4 Comparisons of table extraction methods**

## 4.2 Unstructured data extraction

In our experiment, we apply text analysis on plain text from HTML pages and Wikipedia pages, focusing on special domains, such as football (player and team), movie, etc. Table 7 shows the statistic of patterns for each domain, 6,269 patterns in total for 70 relations defined. For example, the Person domain has 26 relations such as name, date of birth and parents. In particular, the parents relation can be generated several patterns like ("$ARG1 be the child of $ARG2", $ARG1 and $ARG2 are entities and they are child and parent).

**Table 7 Pattern statistics**

| Domains | | Relations | Patterns |
|---|---|---|---|
| Movie | | 14 | 4609 |
| Football | Player | 5 | 130 |

**Table 9 Performance vs. Optimized baseline**

| | Avg. Len. | Avg. Cost | Max. Len. | Max. Cost | Min. Len. | Min. Cost |
|---|---|---|---|---|---|---|
| Baseline | 53.3 words | 4186 ms | 292 words | 44614 ms | 4 words | 100 ms |
| Baseline +Optimized | 53.3 words | 110 ms | 292 words | 669 ms | 4 words | 13 ms |

**Table 10 Precision Evaluations**

| Word Number | Extracted Entity Number | Correct Entity Number | Precision For Entity Extraction | Extracted Relation Number | Correct Relation Number | Precision For Relation Extraction |
|---|---|---|---|---|---|---|
| 150~300 | 229 | 224 | 97.8% | 543 | 531 | 97.8% |
| 50~150 | 105 | 96 | 91.4% | 284 | 275 | 96.8% |
| 20~50 | 236 | 226 | 95.8% | 682 | 667 | 97.8% |
| 1~20 | 102 | 95 | 93.1% | 348 | 336 | 96.6% |
| Overall | - | - | **94.7%** | - | - | **97.3%** |

| | | | |
|---|---|---|---|
| | Team | 7 | 201 |
| Person | | 26 | 682 |
| Organization | | 18 | 647 |
| Sum | | 70 | 6269 |

In our experiment, 106 sentences with different lengths are executed, and the length distribution of the test cases is shown in Table 8. There are totally 11 cases with length between 150 words and 300 words, 11 cases between 50 and 150, 51 cases between 20 and 50 words, 33 cases length larger than 1 word and less than 20 words.

**Table 8 Statistics of Test cases (length)**

| Word Number | 150~300 | 50~150 | 20~50 | 1~20 |
|---|---|---|---|---|
| Sample Number | 11 | 11 | 51 | 33 |

Table 8 shows the performance comparison between baseline and optimized solution. Based on the experiment result, we can see the optimized one average cost is 110ms while the baseline is 4186ms, which is optimized almost 40 times. For long sentence cases, optimization method plays a more significant role because parsing and co-reference spend much more time for long sentences against shorter ones.

Table 9 summarizes the precision evaluation of the results on the test cases. Precision evaluation has two features: entity extraction and relation extraction. Entity extraction mainly refers to extracted entity triple number, correct triple number and the precision. The relation extraction is the same as entity one. There are four categories based on based on ranges of a sentence length. For example, 1-20 region has total 33 cases as shown in Table 8. We extract 102 entity triples and 348 relation triples. Finally, we obtain 94.7% and 97.3% for the precision score of entity extraction and relation extraction.

## 5. Conclusions

In this paper, we presented the knowledge extraction framework, which is to extract knowledge entities and its relations from large-scale data sources. We described a conceptual model, its implementation and evaluation of this framework. This approach has a novel method for extracting various data formats in a single process, and ultimately reduces a processing time of overall text extraction.

For processing various unstructured data sources, specific extraction engines are developed for various formats, such as metadata and web tables from HTML web pages and unstructured and plain text. Metadata extraction is developed to detect the semantic tags in the page source files, and generate triples referring to the ontologies used by these HTML pages. Web tables are also a valuable part for extracting knowledge through cells and rows and columns annotating referring to a knowledge base. The text extraction framework is applied the distant supervision method to train patterns based on seed data and training corpus and then apply these patterns to text for extracting entities and its relations. Finally, we transform these extracted data to knowledge referring to our defined ontology depending on the mappings from named entity types to ontology class, and from patterns to ontology properties.

Since we have multiple extractors extracting data from different sources, we should provide a fusion mechanism to merge all extracted knowledge to our knowledge base. And to improve our framework, we take into account several topics, including extraction performance and precision, more language support and incremental data updates and data fusion.

## References


[1] Mobileappcost.com, Google now, Siri, Cortana, Facebook M, Alexa - A closer look towards Intelligent Virtual Assistants, 2015, available at: http://mobileappcost.com/google-now-siri-cortana-facebook-m-alexa-a-closer-look-towards-intelligent-virtual-assistants/

[2] Jiang, J.; Awadallah, A. H.; Jones, R.; Ozertem, U.; Zitouni, I.; Kulkarni, R. G. & Khan, O. Z. (2015), Automatic Online Evaluation of Intelligent Assistants., in Aldo Gangemi; Stefano Leonardi & Alessandro Panconesi, ed., 'WWW' , ACM, , pp. 506-516 .

[3] Gattani, Abhishek, Lamba, Digvijay S., Garera, Nikesh, Tiwari, Mitul, Chai, Xiaoyong, Das, Sanjib, Subramaniam, Sri, Rajaraman, Anand, Harinarayan, Venky and Doan, AnHai. Entity Extraction, Linking, Classification, and Tagging for Social Media: A Wikipedia-based Approach. Proc. VLDB Endow. 6 , no. 11 (2013): 1126--1137.

[4] Shen, Wei ; Wang, Jianyong ; Luo, Ping ; Wang, Min ; wen Chen, Xue (Bearb.) ; Lebanon, Guy (Bearb.) ; Wang, Haixun (Bearb.) ; Zaki, Mohammed J. (Bearb.): A graph-basesd approach for ontology population with named entities.. In: CIKM : ACM, 2012. - ISBN 978-1-4503-1156-4, S. 345-354

[5] Vrandecic, D. The Rise of Wikidata. IEEE Intelligent Systems 28 (4) , 90-95 . 2013.

[6] Vrandecic, D. & Krötzsch, M. Wikidata: a free collaborative knowledge base. Communication of ACM 57 (10). 78-85. 2014.

[7] Jens Lehmann, Robert Isele, etc. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web 1 (2012) 1–5 1 IOS Press.

[8] Suchanek, Fabian M., Kasneci, Gjergji, Weikum, Gerhard, YAGO: A Core of Semantic Knowledge. In: WWW. New York, NY, USA, ACM Press, 2007

[9] Farzaneh Mahdisoltani, Joanna Biega, Fabian M. Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. Conference on Innovative Data Systems Research (CIDR) 2015.

[10] K. Patel, S. Drucker, J. Fogarty, A. Kapoor, D. Tan. Open Information Extraction: the Second Generation. International Joint Conference on Artificial Intelligence (IJCAI). 2011.

[11] Mausam, M. Schmitz, R. Bart, S. Soderland, O. Etzioni. Open Language Learning for Information Extraction. Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL), 2012.

[12] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphyy, Thomas Strohmann, Shaohua Sun, Wei Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. KDD, 2014.

[13] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08). ACM, New York, NY, USA, 1247-1250. 2008.

[14] Zhongyuan Wang, Jiuming Huang, Hongsong Li, etc. Probase: a Universal Knowledge Base for Semantic Search. Microsoft Research Asia, 2010.

[15] Munoz Emir, Aidan Hogan, and Alessandra Mileo. DRETa: Extracting RDF from Wikitables. International Semantic Web Conference (Posters & Demos). 2013.

[16] Limaye Girija, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. Proceedings of the VLDB Endowment 3.1-2 (2010): 1338-1347.

[17] Adida, Ben, and Mark Birbeck. RDFa primer: Bridging the human and data webs." Retrieved June 20 (2008): 2008.

[18] Ronallo, Jason. HTML5 Microdata and Schema.org. Code4Lib Journal 16 (2012).

[19] Kopecky, Jacek, Karthik Gomadam, and Tomas Vitvar. hrests: An html microformat for describing restful web services. Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on. Vol. 1. IEEE, 2008.

[20] Wang Yalin, and Jianying Hu. A machine learning based approach for table detection on the web. Proceedings of the 11th international conference on World Wide Web. ACM, 2002.

[21] Daniel M. Bikel. On the Parameter Space of Generative Lexicalized Statistical Parsing Models. Ph.D dissertation 2004.

[22] Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the 18th ICML, pages 282–289. Morgan Kaufmann, San Francisco, CA.

[23] Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430.

[24] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang and Jingbo Zhu. Fast and Accurate Shift-Reduce Constituent Parsing. In Proceedings of ACL2013.

[25] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics 39(4): 885-916, 2013.

[26] Mira Ariel. Accessing Noun-Phrase Antecedents. London: Routledge. 1990.