Learning to Extract Attribute Values from a Search Engine with Few Examples*

Xingxing Zhang, Tao Ge, and Zhifang Sui

Key Laboratory of Computational Linguistics (Peking University), Ministry of Education
{zhangxingxing,getao,szf}@pku.edu.cn

Abstract. We propose an attribute value extraction method based on analysing snippets from a search engine. First, a pattern based detector is applied to locate the candidate attribute values in snippets. Then a classifier is used to predict whether a candidate value is correct. To train such a classifier, only very few annotated <entity, attribute, value> triples are needed, and sufficient training data can be generated automatically by matching these triples back to snippets and titles. Finally, as a correct value may appear in multiple snippets, to exploit such redundant information, all the individual predictions are assembled together by voting. Experiments on both Chinese and English corpora in the celebrity domain demonstrate the effectiveness of our method: with only 15 annotated <entity, attribute, value> triples, 7 of 12 attributes' precisions are over 85%; Compared to a state-of-the-art method, 11 of 12 attributes have improvements.

1 Introduction

One of the most important goals of building knowledge bases is to build a big table of *entities* with *attributes* and the corresponding *attribute values*. For example, for a celebrity, has the *attributes*: "height", "weight", and "education" and their corresponding *attribute values*: "1.75m", "65kg", and "xxx university", etc. Entity extraction ([9, 13]) and attribute extraction ([10, 12]) are relative mature technologies, and this paper focuses on the more challenging task, attribute value extraction. The problem setting is: given a list of named entities of a domain (e.g. celebrities), and the attribute names (e.g. birthdate, height, etc.), an algorithm is expected to output triples of <entity name, attribute name, attribute value> for all the entities and attributes. In this paper, when we mention "value", if not specified, it means "attribute value".

An important observation is that many entity attribute values can be found in free text, e.g., there are many sentences like "Chris Pine received his bachelor's degree from University of California, Berkeley" from which we can extract the "education" attribute value. Hence it is feasible to extract attribute values from the web text. However, the challenges are obvious:

(A). It is a huge computational obstacle to parse all the free text in the entire web;

(B). How to make certain that a piece of text is indeed description of the target attribute values other than some irrelevant information.

* This paper is supported by NSFC Project 61075067 and National Key Technology R&D Program (No: 2011BAH10B04-03).

M. Sun et al. (Eds.): CCL and NLP-NABD 2013, LNAI 8202, pp. 154–165, 2013.

In our approach, for a target entity and the attribute name, we take the following steps to get the attribute value: **Step1**: we formulate queries according to the entity and the attribute, and submit them to a search engine and harvest the returned snippets, which often contain candidate attribute values. **Step2**: a pattern based detector is applied to locate the candidate attribute values in each snippet. **Step3**: a statistical classifier is used to predict whether a candidate value in **Step2** is a correct one. **Step4**: as a correct value often appears in multiple snippets, to exploit such redundant information, all the individual predictions of **Step3** are assembled together by voting and then output the final answer of the attribute value.

In the above steps, the key issue is how to train the classifiers in **Step3** and determine the voting weights in **Step4**. Directly labelling the candidate slots in the snippets to train the classifiers for **Step3** requires tremendous human effort. Instead, we require few accurate (manually labelled or from some trustful knowledge bases) <entity, attribute, value> triples, and use again the search engine to find large amount of the pieces of texts containing these correct attribute values. Thus they can be used as pseudo labelled data to train the classifier of **Step3**.

In the proposed approach, challenge **(A)** is conquered by leveraging a search engine to help us find the candidate pieces of text about the entity attributes from the entire web. The underlying assumption is: if a powerful search engine cannot find the pages containing the correct attribute value for an entity, the entity/attribute value must be very rare and we give up. Also, considering the current commercial search engines have already indexed more than billions of web pages, this assumption is reasonable. For challenge **(B)**, our approach adopts a learning approach with very few labelled data. The underlying assumption is, for the same attribute, although different entities will have different attribute values, the expression ways in free text are similar. In addition, the assembling process in **Step4** well utilized the redundancy information from the web to depreciate noise: if many pages agree on an attribute value, it is more likely to be true.

We evaluated our algorithm in celebrity's domain (on both Chinese and English corpus), and remarkable performance is achieved: 7 of the 12 attributes' precision are over 85%. We also compared our algorithm with a state-of-the-art system and find that 11 of the 12 attributes have promising improvements.

The contributions of this paper are: (1) We proposed an approach to leverage a search engine to retrieve the candidate pieces of free texts describing a target entity and attributes from the entire web, rather than very limited websites like Wikipedia as in the previous approaches ([11, 14]). Experiments show that this can significantly improve recall. (2) We proposed a learning approach which learns the ways of describing attribute values of entities in free text. By utilizing the search engine again, the learning approach only requires very few labelled ground truth triples of <entity, attribute, value>.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 presents the proposed algorithm. The experimental setup is presented in Section 4, and the paper concludes in Section 5.

2 Related Work

The existing *attribute value extraction* methods can be roughly categorized into two types: rule-based and machine-learning-based algorithms. Rule based algorithms mainly rely on attribute specific rules. [6] is the champion team of the attribute value extraction task in *WePS* 2009. First, they classify the pages by checking keywords in page title; then for each type of page, keywords and rules for an attribute is employed to extract attribute values. [7] extracts numerical values by sending queries such as "Obj is * [height unit] tall" to extract attribute values. These algorithms above need too many complex attribute specific rules, and they usually have high precisions but low recalls. However, the rules in our method are much simpler; thus, recall is guaranteed. Further, a robust statistical classifier is employed to confirm the extractions.

Machine learning based algorithms usually need some training data. [11] employs co-EM to extract <attribute, value> pairs from product descriptions. As an attribute may have several names (e.g. weight, strung weight), to build a knowledge base, we must disambiguate attribute names, which is not an easy task. [2] mainly extract numerical values; it is viewed as a decision variable that whether a candidate value should be assigned to an attribute. The final value of the decision variables are assigned by solving a constrained optimization problem and the ground facts in training data are constrains of the problem. To further increase the performance, some attribute specific common sense (e.g. unstrung weight is smaller than strung weight) is additionally introduced as constrains. In addition, the computational cost is great. While our algorithm can extract other values besides numerical values, and additional common sense constrains are not needed. The work in [14] is most related to ours. They present *Kylin*, which fills the empty values in Wikipedia's infobox. *Kylin* matches the existing values in infobox back to wiki articles and train a sentence classifier to predict whether a sentence contains certain type of value. Then extracting attribute values from these selected sentences is viewed as a sequential labelling problem, in which CRF is employed. The training data is acquired again by leveraging the existing values in infobox. But they only extract attribute values from a single Wikipedia page, and do not take advantage of the information from other sites. In [16], they also noticed that information on a single page is inadequate, so they employ the ontology in [15] and then articles in hypernym and hyponym classes are used as additional training data. To increase recall, they also use a search engine to get some relevant pages and extract attribute values from these pages. But if models are trained in Wikipedia pages and it runs on general pages, the extraction results may suffer. Besides, for some multi-word values, CRF may cause boundary detection errors. While our method extracts the candidate values as a whole, and boundary detection errors will not happen.

The tasks of *Relation Extraction* and that of *Attribute Value Extraction* are similar. Some relations such as *bornOnDate* and *graduatedFrom* in *relation extraction* are just the attribute values of a person's *birthdate* and *education* attributes, while some relations such as *producesProduct* and *publicationWritesAbout* cannot be viewed as attribute values of certain entity. Pattern-based relation extraction (e.g. [8], [17], [3]) usually bootstraps with some seed relations (facts), and in every iteration new patterns

and facts are extracted and then evaluated by statistical measures such as PMI. Usually, recalls of these methods are high, but these methods often produce noisy patterns and may drift away from target relations.

[5] proposed a multi-stage bootstrapping ontology construction algorithm. In each iteration, they integrated CPL and CSEAL ([4]), which are all pattern-based relation extractor, to fetch candidate facts, and then a statistical classifier is employed to further refine the meta-extractions. However, after each iteration of rule learning, bad extraction rules must be removed manually.

Our approach has a rule based component to detect the candidate attribute values in a piece of free text, getting high recall but low precision candidates. And we also have a successive learning component, a statistical classifier, to further confirm whether a candidate value is indeed a correct one considering the features from its context.

3 Methodology

Suppose that in the target domain, there are N unique named entities and A attributes, and our system is expected to output $N \times A$ attribute values. For each entity e and an attribute name a, we will get one attribute value v. The workflow of our approach is described in Figure 1. The system has four main successive components: *Corpus Fetcher*, *Candidate Value Detector*, *Attribute Value Classifier* and *Voter*. First, in the component of *Corpus Fetcher*, we formulate a query by concatenating the entity e and the attribute name a. For example, for an entity e = "Michael Jackson", and the attribute a = "Birthday", the formulated query is "Michael Jackson Birthday". The query is sent to a search engine, and we fetch the titles and snippets of the top K (=25 in our experiments) returned results. Then, entering into the component *Candidate Value Detector*, for an attribute, we define some patterns to filter the obviously wrong slots in a snippet, for example, some attribute values like *nationality* must belong to a finite enumerable list. Such a pattern based detector is used to roughly locate candidate values in a snippet. Notice that this detector will have high recall but low precision. Next is the task of the component of *Attribute Value Classifier*, a binary statistical classifier that is used to predict whether a candidate value is confident. The prediction is based on features extracted from snippets containing the candidate value

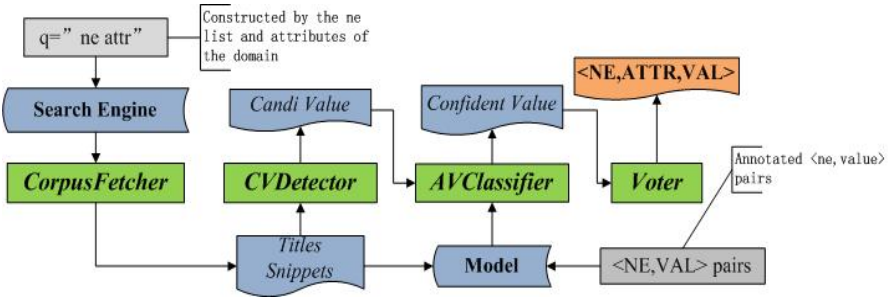

Fig. 1. System architecture. The grey parts are the inputs of the system.

and the snippet's corresponding title. Then all the non-confident candidate values are discarded. Finally, notice that the previous step may produce the same candidate value in different snippets, and it is the often case that the true attribute value appears in multiple snippets. To utilize this redundancy information and make the final decision, we adopt a component of *Voter* to assemble the predictions in the previous step by voting. The voter assigns a weight to each confident value (the refined candidate value in the last step) and accumulates the weights of the same attribute value as its voting score. The candidate values are ranked by their voting score and the extracted attribute value is the one with the highest score. For instance, we have four confident values (three unique ones), "v1 v2 v1 v3". After weighting, each value gets a weight, e.g. "v1:0.98 v2:0.64 v1:0.72 v3:0.99". After accumulation, the voting scores are "v1:1.7 v2:0.64 v3:0.99". So, the final extracted attribute value is "v1".

In the following parts of the section, we will introduce the main components of the system in detail.

3.1 Candidate Value Detector

For an attribute *a*, there should be a *validity checker* to judge whether a candidate value is a valid one. For example, for the attribute *birthday*, a candidate value should be of a valid date format. This paper considers the following two broad cases where the validity checker is easy to obtain: (1) the range of an attribute value is a finite enumerable set, e.g., a valid nationality value must belongs to the set of names of all the countries in the world (there are overall 192 countries); (2) the range of an attribute value can be described by nontrivial regular expressions. For example, birthday values have such formats as "0000-00-00" (e.g. 1986-10-12) or "<Month> 00, 0000", etc. The tested attributes in the paper are all of the two cases, whose valid formats (validity checkers) are shown in Table 1. Actually, for other cases where a value validity checker is provided, the method of this paper can also apply. For example, for the attribute of *spouse*, whose value should be of the type of *person*, we can define a validity checker based on some NER algorithms.

Table 1. Attributes and their formats

Attribute	Format	Attribute	Format
体重 (weight)	(?i)(\\d+(\\.\\d+)?)\\s?(kg|千克|公斤|磅)	出生日期 (birthdate)	(\\d+)年(\\d+)月(\\d+)日, (\\d+)-(\\d+)-(\\d+), …
国籍 (nationality)	Entities in Country list	毕业院校 (education)	Entities in School list
民族(Ethnic Group)	Entities in Ethnic Group list	英文名 (English name)	[A-Z][A-Za-z]+(\\s[A-Z][A-Za-z]+(\\s[A-Z][A-Za-z]+)?)?
血型(blood type)	(A|B|AB|O|0)\\s?型	身高 (height)	(?i)(\\d+)\\s?(cm|厘米), (\\d+\\.\\d+)\\s?米
birthdate	(\\d+)\\s+(January|Jan|February|Feb|...)\\s+(\\d+), (\\d+)-(\\d+)-(\\d+), …	height	(?i)(\\d+)\\s+?cm, (?i)(\\d+)\\s*ft\\s*(\\d+)\\s*in, (?i)(\\d+\\.\\d+)\\s+m, …
nationality	Entities in nationality list	weight	(?is)(\\d+(\\.\\d+)?)\\s+?kg, (?is)(\\d+(\\.\\d+)?)\\s+?lb

The validity checker for an attribute plays the role of a candidate value detector, which detects the valid values in the snippets as candidates. Notice that in this step, some valid but incorrect values may also be extracted as candidates. For example, when we detect valid birthdate values, some irrelevant dates such as the report dates and the page's dates may also be extracted. Actually in this step, we care more about recalls than precisions. In the next steps (Section 3.2, and Section 3.3), from different aspects, we will further filter the candidate values produced in this step to promise high precision. Section 3.2 will filter the candidate values by a classifier considering the context of a candidate value in a snippet. And Section 3.3 will utilize the fact that a correct value often appears in multiple snippets to design a voting mechanism, so that the correct value agreed by multiple snippets is picked up while many incorrect candidates are filtered.

3.2 Attribute Value Classifier

The candidates *Candidate Value Detector* output may be incorrect for the target entity. It may be the case that one snippet may describe several named entities (e.g. celebrities), and the candidate may be other entity's value. In addition, a candidate may be some noise in snippets. For example, the candidate "birthdate" may be just the report date of a piece of news. Thus, we introduce a statistical classifier, the *Attribute Value Classifier*, which aims at refining these candidates by utilizing the features in the snippet containing the candidate and the corresponding title of the snippet.

We train one binary classifier for each attribute, which tries to predict whether a candidate value is confident. The classification model we used is Maximum Entropy Model, which can provide the probabilities of predictions. And in the next section, these probabilities will be used to improve voting.

- **Features**

The prediction is based on features extracted from the snippet containing the candidate value and the corresponding title. We use two types of features: title features describing topics of search results, and snippet features encoding local information of search results. Feature (1)-(3) are title features, and (4)-(7) are snippet features. All these features except (3) and (7) are binary.

(1) Whether the title contains the current named entity. This is a strong indication that the search result is describing the current named entity.
(2) Whether the title contains other named entities of the same class. For example, when extracting Michael Jackson's birthdate, we will see if other celebrities' names are in the title. This is a strong indication that the search result is describing other named entities or the current and other named entities at the same time.
(3) Other words with their POS tags in the title. For example, the title is "Michael Jackson - Wikipedia, the free encyclopedia" and the current named entity is "Michael Jackson". Then feature (3) is "-/: Wikipedia/NNP ,/, the/DT free/JJ encyclopedia/NN".
(4) In the sentence that the candidate value appears, whether the current attribute name appears. This is a strong indication that the candidate value is the value of the current attribute.

(5) In the sentence that the candidate value appears, whether the current named entity appears. This is a strong indication that the candidate value is related to the current named entity.
(6) In the sentence that the candidate value appears, whether the other named entities (of the same class) appear. This indicates that the snippet is describing other named entities, and thus the candidate value may not be a confident value.
(7) Other words with their POS tags and distance to the candidate value in the sentence. For example, in the sentence "Michael Jackson was born on August 29, 1958", "Michael Jackson" is the current named entity and "August 29, 1958" is the candidate value. Then feature (7) is "was/VBD/-3 born/VBN/-2 on/IN/-1". Note that distances of the words on the candidate value's left are negative and distances of the words on the candidate value's right are positive.

The intuitions under the feature design are: (1) if a search result is describing the current named entity, then it is likely that the candidate value in the snippet is correct; (2) if the current named entity or the current attribute appears in the same sentence with the candidate value, then it is also likely that the value is correct.

- **Generating Training Data**

Training the classifier needs labelled data. It is not practical to manually annotate the correct attribute values in each snippet. We propose a method to reduce the labelling effort. Rather than relying on direct annotations on each snippet, we only require a few correct <entity, attribute, value> triples, which are matched back to the search results (title and snippets) to generate training data for the classifier. An advantage of this method is that it is easy to get a few correct <entity, attribute, value> triples, either by human labelling or from some structured sites such as Wikipedia.

Table 2. Numbers of training examples generated by 15 <entity, attribute, value> triples for each attribute. "pos" means the proportion of positive training examples.

Attribute	N	pos	Attribute	N	pos	Attribute	N	pos
体重 (weight)	245	53%	出生日期 (birthdate)	1181	18%	国籍 (nationality)	419	74%
毕业院校 (education)	307	68%	民族 (Ethnic Group)	82	89%	英文名 (English name)	74	24%
血型 (blood type)	188	68%	身高 (height)	207	60%	birthdate	393	34%
height	77	41%	nationality	338	62%	weight	77	74%

Step 1: Annotate some <entity, attribute, value> triples (only 15 triples in the experiment). We can also get these triples from some structured sites (e.g. Wikipedia, Bio27, etc.).

Step 2: Submit queries to a search engine and match these <entity, attribute, value> triples back to search results. For example, we are extracting celebrities' birthdate, and we have a labelled triple of <'Michale Jackson', 'Birthdate', '19580829'>. We send the query "Michael Jackson birthdate" to a search engine and get the top *K*

(*K*=25 in our experiments) search results (titles and snippets). Then the *Candidate Value Detector* extracts all the candidate values and converts them to standard formats ('yyyymmdd'). If a candidate value equals to the true value ("19580829"), then we annotate a positive label to the value in the snippet and get a positive training example; otherwise, we get a negative training example. In this way, each candidate value in a snippet will produce a training example. The number of training examples generated by the 15 <entity, attribute, value> triples for each attribute is shown in Table 2. The number of positive training examples is not necessarily less than the number of negative training examples. Proportions of positive training examples are in the 'pos' columns.

3.3 Voter

After the classification, there may be several candidate values for an entity's attribute and the correct value often appears in multiple snippets. Intuitively, the most frequent candidate value is most likely to be the correct value. Therefore, a simple strategy is to count how many times a candidate value is classified as confident value by the classifier. However, this may cause a problem when several candidate values get the same highest score. To alleviate the problem, we leverage the classification probabilities provided by the *Attribute Value Classifier* and use the probability as each vote's weight. Experiments show that this strategy can improve precisions and recalls by about 1%.

4 Experiment

We use Baidu[1] (Chinese) and Google (English) to test our algorithm. For our *Attribute Value Classifier*, we employ a Maximum Entropy model implemented by Le Zhang [18]. We employed L-BFGS and the Gaussian prior is 1.0.

Table 3. Numbers of nonempty values for each attribute

Attribute	N	Attribute	N	Attribute	N	Attribute	N
体重 (weight)	667	出生日期 (birthdate)	3004	国籍 (nationality)	3726	毕业院校 (education)	1162
民族 (Ethnic Group)	861	英文名 (English name)	2759	血型 (blood type)	1652	身高 (height)	2985
birthdate	1532	height	1543	nationality	1295	weight	1522

[1] http://www.baidu.com
[2] http://ent.qq.com/c/all_star.shtml
[3] http://app.ent.ifeng.com/star/
[4] http://baike.baidu.com
[5] http://www.hudong.com
[6] http://www.wikipedia.org
[7] http://www.bio27.com

4.1 Evaluation Dataset

The experiments were conducted in the celebrity domain (on both Chinese and English corpus). We collected the 4476 celebrities in "qq entertainment[2]" as our Chinese named entity list. And we crawled celebrities' data in "qq entertainment", "ifeng entertainment[3]", "baidu baike[4]", "hudong baike[5]" and "Wikipedia[6]" as our Chinese standard evaluation dataset. Similarly, we collected 1600 celebrities in "bio27[7]" as our English named entity list. And we crawled celebrities' data in "bio27" as our English standard evaluation dataset. Some named entities' values cannot be found in all these sites, and their values are empty in our dataset. Numbers of nonempty values for each attribute are in Table 3.

4.2 Experimental Results

We tested 14 attribute values, and 8 of them are on Chinese corpora (alias Cx), while 4 of them are on English corpora (alias Ex). In addition, we use 15 <entity, attribute, value> triples for each attribute to generate training data and train the *Attribute Value Classifier*. We evaluated their Precisions and Recalls. As some named entities' certain attribute may not exist (e.g. M. Jackson is an American, and does not have the attribute "民族(Ethnic Group)"), we only evaluate the attribute values in the our dataset. The results in Table 4 show that 7 of the 12 attributes' precision are over 85%.

Table 4. Results with 15 <entity, attribute, value> triples. In graphs, attributes are replaced by their "Alias". "Cx" donates a Chinese attribute, while "Ex" donates an English attribute.

Attribute	Alias	P	R	Attribute	Alias	P	R
体重 (weight)	CW	0.7711	0.6312	出生日期 (birthdate)	CB	**0.8582**	0.7397
国籍 (nationality)	CN	**0.9239**	0.8341	毕业院校 (education)	CE	**0.8512**	0.7040
民族(Ethnic Group)	CEG	0.8357	0.6202	英文名 (English name)	CEN	0.6320	0.4937
血型(blood type)	CB	**0.9322**	0.7488	身高 (height)	CH	**0.8676**	0.6938
birthdate	EB	**0.9139**	0.9073	height	EH	0.7120	0.6137
nationality	EN	**0.9451**	0.9444	weight	EW	0.7533	0.6419

4.3 Single Site *vs.* Multiple Sites

To increase recall, we leverage a search engine to extract attribute values from the entire web, rather than very limited websites. In this section, we provide evidences for this claim. We studied the best recalls an algorithm can achieve on two sites, namely, Wikipedia and Baidu-baike (Chinese version Wikipedia), and compared them with that of the proposed algorithm. Specifically, for a target entity and one of its attributes, if its correct attribute value for the current attribute can be found on the entity's page, it does count for a correct extraction. The comparison in Figure 2 shows

that the recall of the proposed algorithm is better than the best recalls on Wikipedia and Baidu-baike. That is to say that no algorithm targeting at these two sites can have a better recall than the proposed algorithm. Therefore, it is necessary to leverage the information from multiple sites.

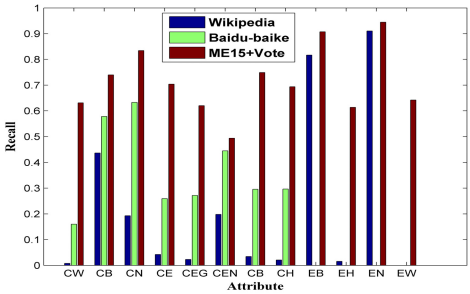

Fig. 2. Best recalls an algorithm can achieve on Wikipedia and Baidu-baike and the recall of the proposed algorithm. The attributes on x-axis are all their aliases (details in Table 4).

4.4 Comparison to a Previous System

In this section, we compare the performance of different methods on algorithm level (on the same corpora).

It can be difficult to compare our results with other attribute value extraction systems. Unlike semantic role labelling, there are some public available datasets (e.g. PropBank and Pen TreeBank). The datasets used by previous systems are different from ours. One feature of our system is the leverage of multiple sites data, so we cannot only use Wikipedia as in [14]; besides numerical values, we can extract other kind of values, so we cannot use the dataset in [2]. Finally, we implemented *Kylin* [14].

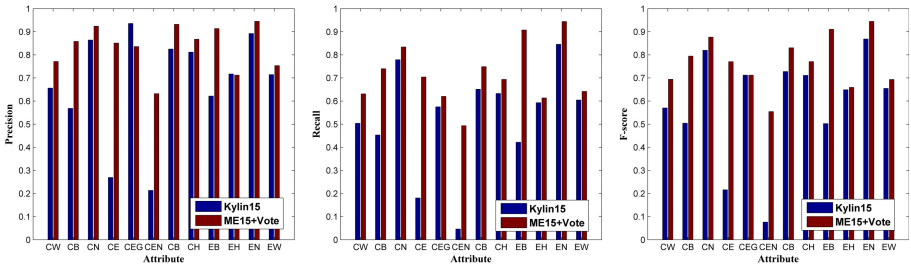

Fig. 3. The comparison between *Kylin* ([14]) and our method (ME15+Vote). Both systems use 15 <entity, attribute, value> triples. The attributes on x-axis are all their aliases (Table 4).

In experiments, we implemented *Kylin* in [14] to extract attribute values from snippets. We tested *Kylin* with 15 annotated <entity, attribute, value> triples and 100 annotated <entity, attribute, value> triples (the same amount of annotated triples with our method) respectively. And the comparisons are shown in Figure 3 and Figure 4 respectively.

In Figure 3, ME15+Vote have a better precision, recall and F-score on all attributes except the attribute “民族(Ethnic Group)”(CEG). Among these attributes, the improvements on “出生日期(birthdate)”(CB), “毕业院校(education)”(CE), “英文名(English name)”(CEN) and “birthdate”(CB) are obvious. In Figure 4, results of *Kylin* with 100 annotated pairs are better than their results with 15 annotated triples on most attributes. However, the results are similar with that of 15 annotated triples: still only the results of “民族(Ethnic Group)”(CEG) are better than the proposed method. The proposed method leverages a rule-based detector to locate the candidate values and during classification, the features in title, which reflects the topic information of the snippet, are used. We believe these factors above lead to a better performance.

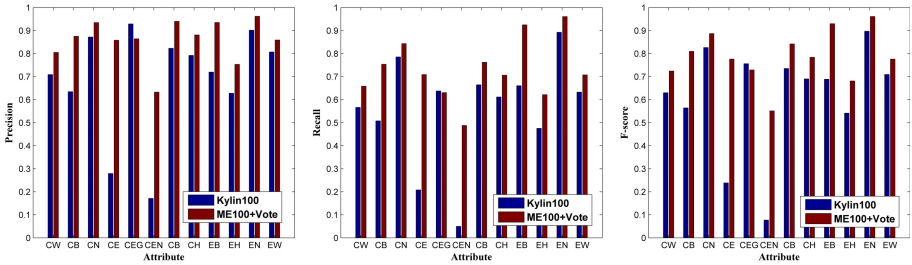

Fig. 4. The comparison between *Kylin* ([14]) and our method (ME100+Vote). Both systems use 100 <entity, attribute, value> triples. The attributes on x-axis are all their aliases (Table 4).

4.5 Impact of the Amount of Annotated Data

We also studied the effects of different amount annotated <entity, attribute, value> triples on F-scores. It is shown in Figure 5. We can see when the amount of annotated triples is between 5 and 15, some attributes’ (e.g. 体重(weight)) F-scores increase significantly; when the amount is between 15 and 100, F-scores do not increase much; when the amount is between 100 and 300, “英文名(English name)” and weight have about 5% increase. But for 5%’s improvement to annotate 20 times of data is not worthy. So we use 15 annotated triples, and they are sufficient for most attributes.

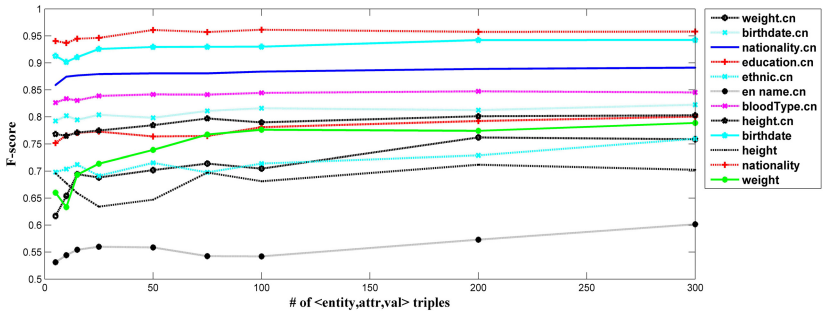

Fig. 5. The impact of amount of annotated <entity, attribute, value> triples on F-score

5 Conclusions

In this paper, we proposed an attribute extraction algorithm. The attribute values are extracted from snippets returned by a search engine. We use some strict (mainly rule based) methods to locate the possible values in the snippets. Then a classifier is used to predict whether the candidate value is a confident one. To train the classifier, we only need to annotate very little data, and sufficient training data will be generated automatically. A correct value may appear in multiple snippets, and we also have a strategy to vote and score the confident values.

References

1. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: IJCAI (2007)
2. Bakalov, A., Fuxman, A., Talukdar, P., Chakrabarti, S.: Scad: collective discovery of attribute values. In: Proceedings of WWW 2011, Hyderabad, India, pp. 447–456 (2011)
3. Cafarella, M.J.: Extracting and querying a comprehensive web database. In: CIDR (2009)
4. Carlson, A., Betteridge, J., Wang, R.C., Hruschka Jr., E.R., Mitchell, T.M.: Coupled semi-supervised learning for information extraction. In: Proc. of WSDM (2010a)
5. Carlson, A., et al.: Toward an architecture for never-ending language learning. In: Proceedings of AAAI 2010 (2010b)
6. Cimiano, P., Völker, J.: Text2Onto – a framework for ontology learning and data-driven change discovery. In: NLDB (2005)
7. Davidov, D., Rappoport, A.: Extraction and Approximation of Numerical Attributes from the Web. In: Proc. of ACL (2010)
8. Etzioni, O., et al.: Unsupervised named-entity extraction from the web: An experimental study. Artif. Intell. 165(1) (2005)
9. Kozareva, Z., Riloff, E., Hovy, E.: Semantic class learning from the web with hyponym pattern linkage graphs. In: Proceedings of ACL 2008: HLT (2008)
10. Pasca, M., Van Durme, B.: Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. In: Proceedings of ACL 2008, pp. 19–27 (2008)
11. Probst, K., Ghani, R., Krema, M., Fano, A., Liu, Y.: Semi-supervised learning of attribute-value pairs from product descriptions. In: IJCAI (2007)
12. Ravi, S., Pasca, M.: Using Structured Text for Large-Scale Attribute Extraction. In: Proceedings of CIKM 2008, pp. 1183–1192 (2008)
13. Wang, R.C., Cohen, W.W.: Language-independent set expansion of named entities using the web. In: ICDM, pp. 342–350. IEEE Computer Society (2007)
14. Wu, F., Weld, D.S.: Automatically semantifying Wikipedia. In: CIKM, pp. 41–50 (2007)
15. Wu, F., Weld, D.S.: Automatically refining the wikipedia infobox ontology. In: Proceedings of WWW 2008 (2008)
16. Wu, F., Hoffmann, R., Weld, D.S.: Information extraction from Wikipedia: Moving down the long tail. In: Proceedings of KDD (2008)
17. Xu, F., Uszkoreit, H., Li, H.: A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In: ACL (2007)
18. Zhang, L.: Maximum Entropy Modeling Toolkit for Python and C++ (2004), `http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html`