World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:11, No:6, 2017

Evolving Knowledge Extraction from Online Resources

Zhibo Xiao, Tharini Nayanika de Silva, Kezhi Mao

Open Science Index, Computer and Systems Engineering Vol:11, No:6, 2017 waset.org/Publication/10007303

***Abstract*—In this paper, we present an evolving knowledge extraction system named AKEOS (Automatic Knowledge Extraction from Online Sources). AKEOS consists of two modules, including a one-time learning module and an evolving learning module. The one-time learning module takes in user input query, and automatically harvests knowledge from online unstructured resources in an unsupervised way. The output of the one-time learning is a structured vector representing the harvested knowledge. The evolving learning module automatically schedules and performs repeated one-time learning to extract the newest information and track the development of an event. In addition, the evolving learning module summarizes the knowledge learned at different time points to produce a final knowledge vector about the event. With the evolving learning, we are able to visualize the key information of the event, discover the trends, and track the development of an event.

***Keywords*—Evolving learning, knowledge extraction, knowledge graph, text mining.

I. Introduction

THE information on the Internet is overwhelming us. Even for the result of a single query on a search engine, it is hard to quickly grasp the key information underlying the returned search results. Traditional media and social media are also shortening our attentions and shifting our focuses with the news flash that we usually may not be able to keep up with. In order to address this problem, automatic knowledge extraction has been proposed.

A few open information extraction (OIE) systems have been developed. As stated in [1], an OIE system takes in a corpus of texts, without any a priori knowledge or specification of the relations of interest, and outputs a set of extracted relations. One example work is KnowItAll [2], which is an information extraction system that addresses the challenge of automated information extraction by learning to label its own training examples. KnowItAll sets a foundation of modern OIE system but it requires a large amount of data from search engine to train and learn relations. The more recent TextRunner [3], however, learns a general model of how relations are expressed in a particular language using CRF paradigm, in this way, it not only reduces the error rate of KnowItAll, but also save the time when learning new relations. Recently, SemIE is proposed in [4], which extends TextRunner and exploits the predicate-argument structure of a text to handle complex sentences.

In recent years, evolving learning has received more and more attentions. Different aspects of evolving learning in information retrieval and information extraction have been investigated. reference [5] proposed a summarization method to track the development of tweet stream. This work extracts key phrases but does not discover the development of an event. Reference [6] proposed a method to track the emerging topics of streaming documents using dictionary learning. References [7]-[9] investigated a specific time-related information retrieval task, chronological citation recommendation, which utilized different topic models to discover time-related features of academic citations. In spite of the success of the aforementioned work, each work only touches on one single aspect of evolving learning. In this study, we present our AKEOS (Automatic Knowledge Extraction from Online Sources) which is an end-to-end system.

AKEOS consists of two main modules, including a one-time learning module and an evolving learning module. The one-time learning aims to distil the valuable information from the overwhelmingly large volume of unstructured text that search engine returns for one query, and output a structured knowledge vector to summarize the query result. The one-time learning extracts knowledge for one query session and the knowledge extracted just reflects the event until a particular time. As the event develops, new information will be generated. The objective of the evolving learning is to extract the knowledge of the new development of the event. The user only needs to submit a query once, the evolving learning mechanism in AKEOS automatically schedules and performs the one-time knowledge extraction, and summarize the knowledge extracted at a different time. Here, the knowledge summarization is done through knowledge vector merging, including attribute merging and value merging. The end product is a knowledge vector. The final knowledge vector summarizes the overall information of an event, while the individual knowledge vectors generate at different time points reflect how the event evolves over time.

In summary, AKEOS has the following merits:

1) It harvests unstructured text data from online resources, distils the important information from the data to produce structured knowledge vectors.
2) It is an end-to-end system, with query words as input, knowledge vector as the end product.
3) It tracks the development of events and discovers the underlying patterns.

II. System Architecture

AKEOS consists of two modules: one-time learning and evolving learning. The one-time learning module is a

Zhibo Xiao and Tharini Nayanika de Silva are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.
Kezhi Mao is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: ekzmao@ntu.edu.sg).

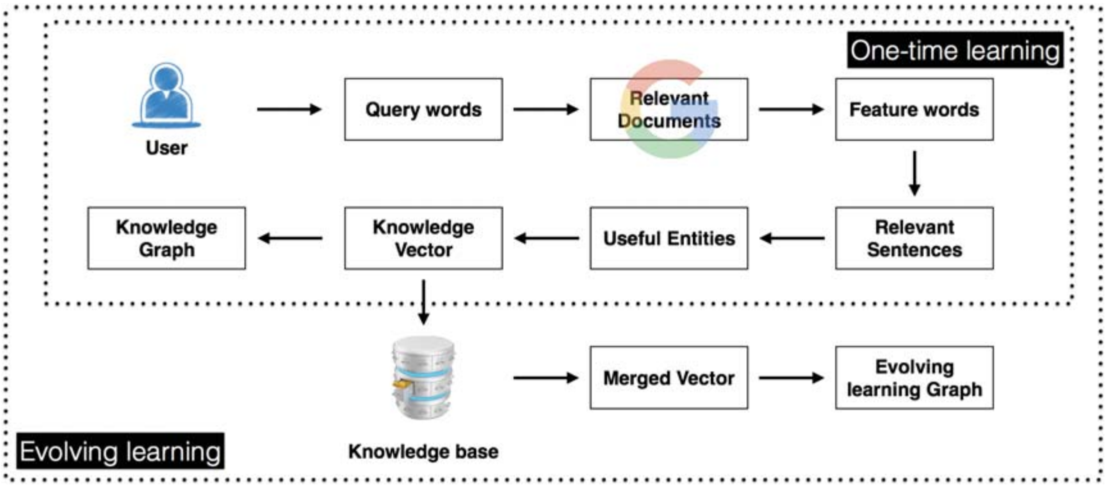


Fig. 1 The AKEOS architecture

stand-alone system, while the evolving learning module is built upon the one-time learning module. The system architecture is shown in Fig. 1. The inputs to the one-time learning system are query words from a user. The query words are then used to crawl related documents from online resources. Since Google ranks search results by relevance to the query, top documents, say top 100 documents, are deemed “relevant documents”. The one-time learning system first extracts the main paragraph text from each document. Feature words are then selected from the relevant documents. These feature words are associated with key information of an event and are used as a guide to relevant sentence selection. From these relevant sentences, useful entities are extracted. Through this procedure of knowledge extraction, the unstructured text is transformed into a structured knowledge vector. Each time the one-time learning is performed, the structured knowledge vector will be stored in the knowledge base for further use.

The **one-time learning** module only extract the knowledge of an event until a particular time point. The **evolving learning** module schedules and performs repeated one-time learning at different stages of an event to ensure the knowledge extracted is up-to-date. The newly extracted knowledge will be merged with previously extracted knowledge to form a new vector. For example, after the first day of an event, the feature word list for day 1 is learned, along with relevant sentences, extracted useful entities and the knowledge vector v_1 of the event for day 1. After the second day, the feature word list based on the newly published articles on day 2 are generated, along with the corresponding relevant sentences, useful entities and knowledge vector of v_2. At the end of day 2, the knowledge vectors of day 1 and day 2 are merged into a new vector, denoted by u_1. After n days, u_{n-1} is learned and is treated as final vector $v = u_{n-1}$. User can use the final vector v to check the summarized knowledge of the event, and use $v_i, i = 1, 2, \ldots, n$ to review the development of the event. With $u_i,\ i \in \{1, \ldots, n-1\}$, user can also check how one particular attribute changes over time in the corresponding evolving learning graph.

III. One-Time Knowledge Extraction

One-time learning module takes in user query to crawl the Internet for relevant articles and extract relevant information from these article. One-time learning is the foundation of the evolving learning module.

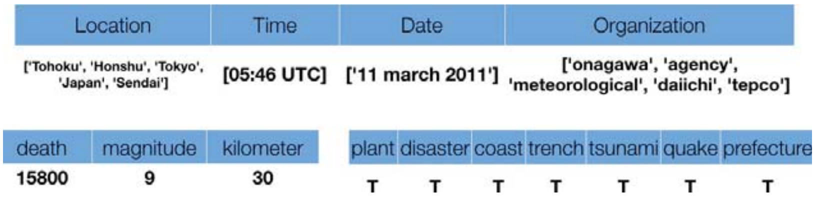
The module first takes in user query and use Google search engine to search relevant articles on the Internet. We treat the top 100 query results from Google as relevant articles. After extracting the text from these webpages, we adopted an enhanced TF-IDF method which combines the TF-IDF measure with word frequencies from Google’s N-gram dataset to measure the importance of the word in the context of the user query in relation to a generic context. Extracted feature words will be used in the following tasks, including one-time learning and evolving learning.

Till now, the relevant documents concerning the query words are collected, but not all the sentences are relevant to the query words. Hence, feature words are used to select the relevant sentences. For a sentence containing M words in the corpus, each word is transformed into a vector, then convoluted with each feature word vector. The sentence is transformed into a vector with the length of M. A further max pooling operation will select the most distinct feature in the sentence. In this manner, a sentence with variable length is transformed into a scalar. With a predefined threshold, the relevant sentences are selected.

After relevant sentences are selected, the next step is to extract entities containing useful information. In AKEOS, the following are extracted as entities:

- Quantitative words, which are words associated with a number, such as ‘deaths’, ‘people killed’, ‘missing’, ‘km’, ‘magnitude’.
- Geographic locations
- NER tags, such as date, time, location, person, organization

After the entities are extracted, NER tagged properties, quantitative properties, and other feature words are represented as a knowledge vector. The knowledge vectors for the same



Location	Time	Date	Organization
['Tohoku', 'Honshu', 'Tokyo', 'Japan', 'Sendai']	[05:46 UTC]	['11 march 2011']	['onagawa', 'agency', 'meteorological', 'daiichi', 'tepco']

death	magnitude	kilometer
15800	9	30

plant	disaster	coast	trench	tsunami	quake	prefecture
T	T	T	T	T	T	T

Fig. 2 Example of knowledge vector of event “Tohoku earthquake”

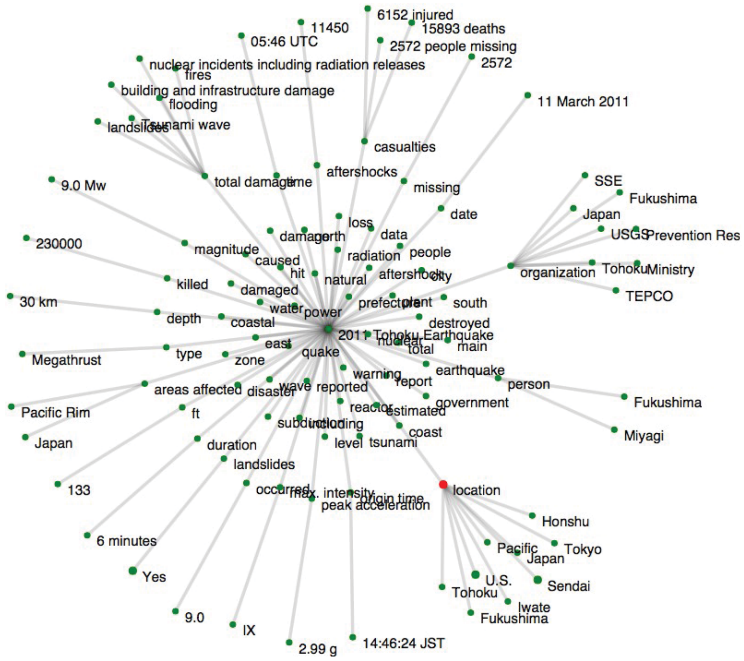


Fig. 3 Example of knowledge graph of “Tohoku earthquake”

event on different days are then used as input for evolving learning system. An example of the knowledge vector for the query “Tohoku earthquake” is shown in Fig. 2. The knowledge vector consists of three parts, including the NER features, the quantitative features, and other feature words. The knowledge vector can be visualized using D3.js [10] as shown in Fig. 3. The center is the query word submitted by user, the feature words are scattered around the query word in the level, and the value of each feature words constitute the second level.

IV. Evolving Learning

The main motivation to develop evolving learning in AKEOS is to introduce a time dimension to the query result of an event. In one-time learning, after keying in the query words, the system returns a knowledge graph as the result. The Knowledge graph reveals key information about an event based on the information extracted at the time. But for a developing event, it is observed that

1) Facts update over time;
2) Media/Public attention shifts over time.

The above two characteristics raise the requirement to automatically keep track of new information in a developing event. For example, when *2015 Paris attack* happened on November 13th, the media mainly reported the locations of the attack and a rough estimate of casualties on Day 1. On Day 3, the news reports all gave an accurate casualty number. In the meantime, the media started to report the possible perpetrator. On Day 5, after identifying the perpetrator, the media started a deeper investigation to find possible linking between the Syria refugee and this attack, and discuss the refugee policy that Belgium takes. This example exhibits how an event unfolds through time, and how people and media attention changes over time.

In our evolving learning system, the user only needs to submit an event query, the system automatically runs the whole knowledge learning process each day until no new information is learned. After the process is done, the system automatically summarizes the knowledge learned from the starting date of the event while keeping the knowledge learned in each day. The whole process is illustrated in Fig. 4. In the following, we will explain the evolving learning in details.

A. *Updating Event Vector*

Facts about an event update every day, especially in the first few days of an event. The one-time learning system guarantees that the system extracts the correct information from the documents collected in each day. On the other hand, to reflect the changes and evolving trend, the latest knowledge

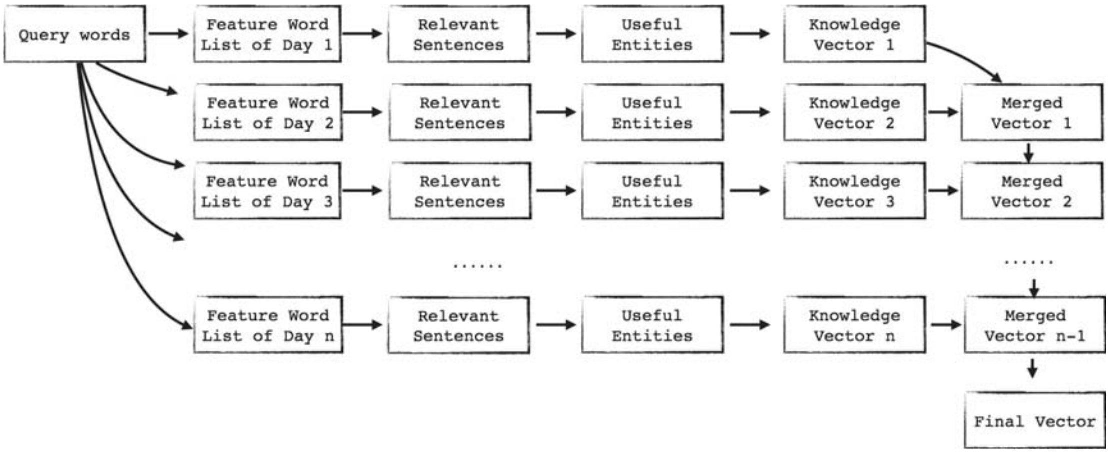


Fig. 4 The evolving learning module architecture

vector must satisfy the following criteria:

- If two features are semantically similar, they should be combined as one.
- If a feature word appears in every day's knowledge vector, the merged vector should keep its latest value.
- If a feature word does not appear every day, the merged vector should keep its latest value.
- The merged vector on every day should be kept.

Automatically merging knowledge vector requires that the old vector must be updated using the newly extracted knowledge vector each day. This requires the system to perform feature merging and value merging.

1) Feature Merging: In evolving learning system, after several days' of learning the development of the certain event, the corresponding learned vector is accumulated, in order to show the final knowledge graph, these vectors need to be merged. If a new feature word w_i has never been learned in previous days' feature word list, it will be automatically inserted into the final feature word list; if a feature word w_j which has already been in previous day's feature word list, the system should merge the value of w_j in the previous vector and the value of w_j in the current vector.

In some cases, a few different but semantically similar words appear in the feature word list, for example, "injure" and "wound", "plant" and "building". Keeping them all in the final feature list has some short-comings:

- The final vector would be too long
- Values of the same entity is scattered under different features

The first shortcoming is self-explanatory. The second shortcoming will create redundant nodes in knowledge graph of one-time learning. In evolving learning, it will create several similar outcomes with missing information. For example, in the "2015 Paris attack", on Day 2, the word *wound* is one of the feature words, and its value is 99, on Day 3, the word *injure* is in the feature word list, and its value is 352. Clearly, both *wound* and *injure* talks about the number of people injured in the attack, the number increases as time goes by. As stated above, the system should merge these two words together, and save their corresponding value into knowledge vector. In order to deal with this problem, following steps are adopted in AKEOS:

- For any given feature word w_i, check whether it is in previous day's feature list, if it already exists in the feature word list, then merge the values of this feature with previous days' feature values(see Section IV-A2).
- If it is not in previous days' feature word list, then select 5 sentences containing this word. Perform word sense disambiguation to decide which meaning in the WordNet this word adopted in this sentence, since a word may have multiple meanings.
- After deciding the meaning of this word in the context, we adopted *lch similarity* to calculate the differences between this word to the existing feature word list.

We set the threshold of 2.15 of lch similarity as the default value. Although there are other similarity measures similar to lch, after experimenting the performance of these measures, we found out that the key factor is the WordNet's categories playing the crucial role in this task, hence we continue to use lch similarity to perform the feature merging task.

2) Value Merging:

```
Suicide bomb, Car bomb, car bomb, bomb,
Suicide car bomb, School bomb, suicide
bomb, Truck bomb, Bicycle bomb, suicide
bombing, Bomb, bomb vest, petrol bomb,
backpack bomb
```

The above paragraph shows an example of the values of attack types in the database. To a human being, the attack types are bombing, we need to merge them together as a single item. For example, bomb or bombing contain the string bomb. To merge them, we need to solve two problems:

1) Can they be merged together? Do they contain the same string?
2) Which one should be selected to represent them?

To solve the first problem, we adopted fuzzy string matching method to calculate the similarity of strings. We adopted pywsd [11] to calculate the similarity and experimented several thresholds to determine which parameter servers best to find the similar strings. We found that 2.15 is a suitable value since it gives the best result.

To solve the second problem, we resort to Levenshtein distance. In information theory and computer science, the Levenshtein distance is a metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. For example, the Levenshtein distance between ”kitten” and ”sitting” is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

1) **k**itten → **s**itten (substitution of ”s” for ”k”)
2) sitt**e**n → sitt**i**n (substitution of ”i” for ”e”)
3) sittin → sittin**g** (insertion of ”g” at the end).

We use fuzzywuzzy Python package[1] to calculate the Levenshtein distance between strings and use 70 as the threshold to select the similar string. After the processing above, the remaining strings can be seen as similar strings with the same meaning, and the shortest string is selected to represent the whole group.

B. Stopping Criteria

One of the key issues in evolving learning is the stopping criteria for knowledge learning. In AKEOS, particularly the evolving learning system, the stopping point is the time when the latest published documents concerning the query event does not have or have very little new **relevant** information. The key criteria here is to determine the degree of relevancy of new information to the query. Take “2015 Paris attack” as an example, if this query term is put into Google, new documents will still be returned but few of them will be about the event, instead, most of them will be about subject remotely relevant to this event. In order to find the stopping criteria, we monitored three events:

- Jakarta bombing
- Istanbul bombing
- Burkina Faso attack

In these three events, *Istanbul bombing* is a big event which attracted a lot of media coverage; *Jakarta bombing* is a relatively small event compared with Istanbul bombing; *Burkina Faso attack* is an even smaller event which did not get a wide news coverage. We picked out these three different scaled events with the aim to compare and find the stopping criteria of evolving learning. Concerning the stopping criteria, we compared two criteria:

- The number of features of Day i compared with the number of features of Day $i-1$, which is called **daily difference**
- The number of features of Day i compared with the number of features of all previous days, which is called **accumulated difference**

Figs. 5-7 show the **daily difference** values for *Jakarta bombing*, *Istanbul bombing* and *Burkina Faso attack* respectively. The x-axis shows the crawling day, starting from day 0; the y-axis shows the **daily difference** in percentage. For example, in Fig. 5, the first bar’s y value reads 88%, which means that the 2nd days’ feature list contains 88% of the word in 1st days’ feature list.

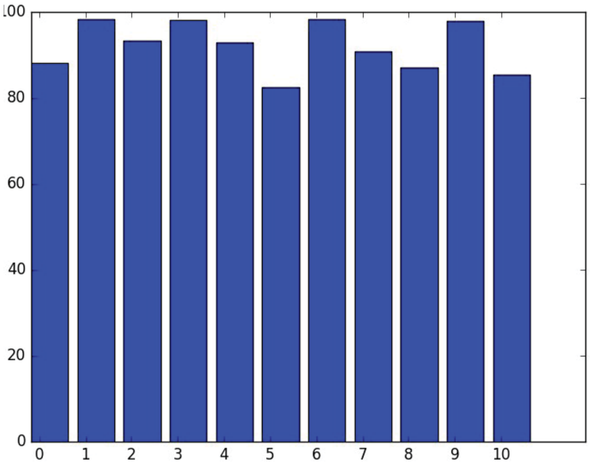


Fig. 5 Daily difference of the query “Jakarta bombing”

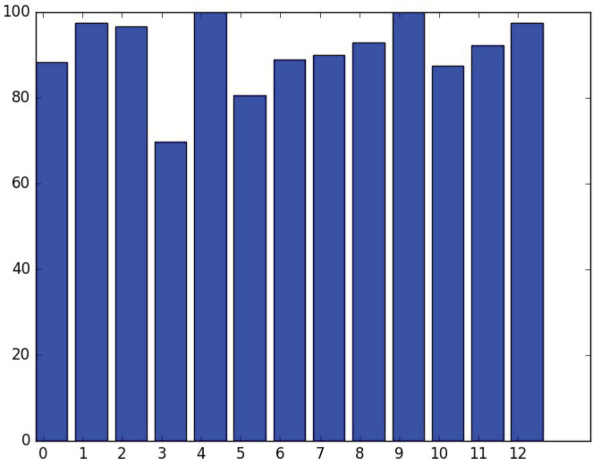


Fig. 6 Daily difference of the query “Istanbul bombing”

From Figs. 5-7, we can see that in both small events with little coverage or big event with wide coverage, the number of feature words changes day to day. There is no clear indication on how these feature words changes, for some days, feature word number decrease compared with the previous day, while for some days, feature word number increase compared with the previous day. Next, we turn to analyze the **accumulated difference** factor.

Figs. 8-10 show the **accumulated difference** of each day’s feature word number to accumulated feature word list. The x-axis indicates the day, and the y-axis shows the percentage of accumulated difference. The positive values in the figure show that this day’s feature word list contain new feature word which has never occurred in previous days’ feature word list, the negative values indicate that this days’ feature words’ number decreased. The **accumulated difference** figures also give a mixed result on evaluating the new information based

[1] https://github.com/seatgeek/fuzzywuzzy

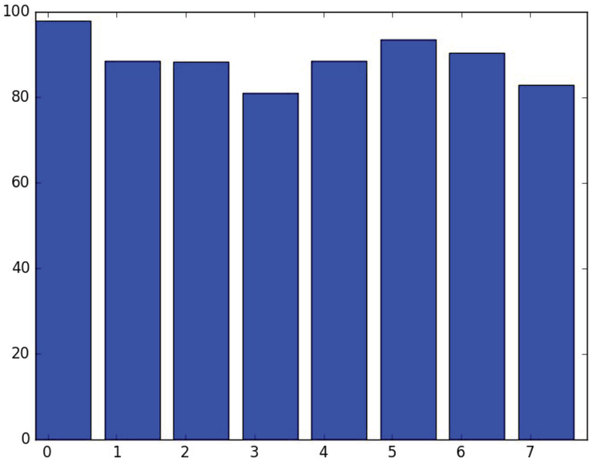


Fig. 7 Daily difference of the query "Burkina Faso attack"

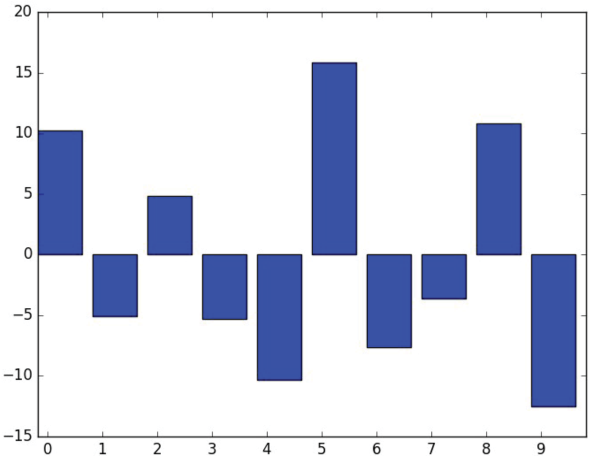


Fig. 8 Accumulated difference of the query "Jakarta bombing"

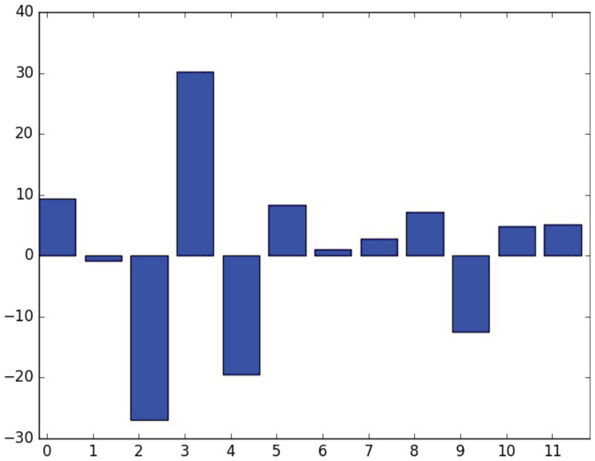


Fig. 9 Accumulated difference of the query "Istanbul bombing"

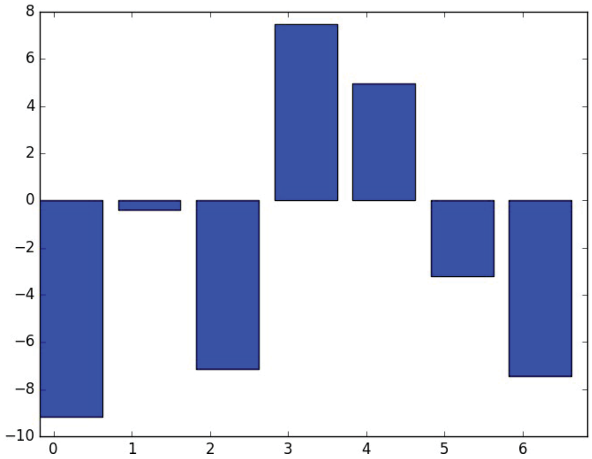


Fig. 10 Accumulated difference of the query "Burkina Faso attack"

on feature words. For big and small events, compared with previously accumulated feature words, there is no clear indication on which day the evolving learning should be stopped. In these experiments, we use the query stopping day as a parameter which can be set by the user to fit various needs.

V. Evolving Learning Examples

With aforementioned learning process, we present examples of evolving learning to validate the effectiveness of our system.

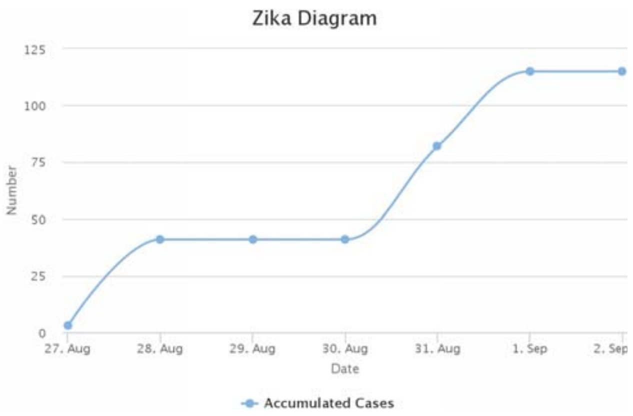


Fig. 11 Diagram showing the number changes of Zika infected case number of evolving learning result of query "Singapore Zika outbreak"

For query "Singapore Zika outbreak", Fig. 11 reveals the case number of first seven days of Singapore Zika virus outbreak query result. The x-axis is the date, and the y-axis shows the number of infected cases. The figure clearly shows the sharp increase numbers of the infected case number.

For query "Pairs attack 2015", Fig. 12 shows the number of people killed in the first seven days of Paris attack. The x-axis is the date, and the y-axis is the number of people killed. As can be seen from the picture, the majority of the data points showed the number of the casualty of people, however, on the second day, the number of people killed is 2, this is not an error, this number reveals the number of the police officer

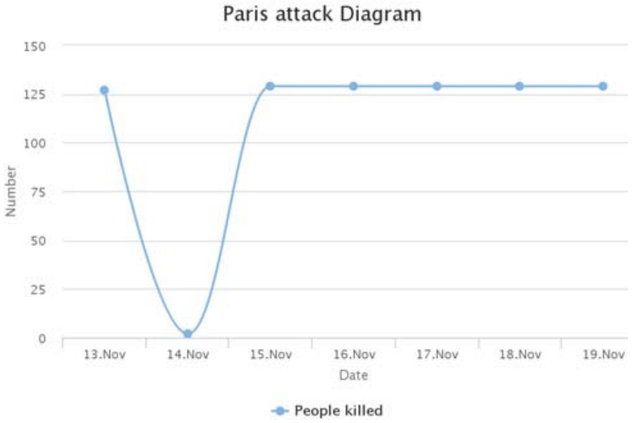


Fig. 12 Diagram showing the number changes of people killed of evolving learning result of query "Pairs attack 2015"

killed on that day. Since this is the focus of the media on the second day, the system automatically picks up this number.

VI. Conclusion

In this paper, we have presented an evolving information extraction system AKEOS that automatically harvests knowledge from on-line resources. AKEOS consists of two main modules, one-time learning module and evolving learning module. The one-time learning module extracts knowledge for one query, starting from feature word extraction, relevant sentences extraction and useful entities extraction. The whole process is performed in an unsupervised manner. The one-time learning transforms the unstructured text to structured knowledge vector and knowledge graph. The evolving learning module is built upon the one-time learning module, which extracts knowledge of an event in consecutive days to discover trends and patterns. With daily extracted knowledge vector, evolving learning module automatically merge the new vector with existing vectors to reflect the latest development of the event. When no new information appears, the evolving learning module generates the evolving learning graph for the user to view the trend of key information in the event. In the future, we will investigate how to use multiple sources to validate the accountability of extracted information.

References

[1] J. Piskorski and R. Yangarber, *Information Extraction: Past, Present and Future*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 23–49.

[2] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artificial intelligence*, vol. 165, no. 1, pp. 91–134, 2005.

[3] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[4] S. S. Tan, T. Y. Lim, L.-K. Soon, and E. K. Tang, "Learning to extract domain-specific relations from complex sentences," *Expert Systems with Applications*, vol. 60, pp. 107 – 117, 2016.

[5] L. Shou, Z. Wang, K. Chen, and G. Chen, "Sumblr: Continuous summarization of evolving tweet streams," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '13. New York, NY, USA: ACM, 2013, pp. 533–542.

[6] S. P. Kasiviswanathan, P. Melville, A. Banerjee, and V. Sindhwani, "Emerging topic detection using dictionary learning," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ser. CIKM '11. New York, NY, USA: ACM, 2011, pp. 745–754.

[7] Z. Jiang, X. Liu, and L. Gao, "Dynamic topic/citation influence modeling for chronological citation recommendation," in *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, ser. Web-KR '14. New York, NY, USA: ACM, 2014, pp. 15–18.

[8] Z. Jiang, "Chronological scientific information recommendation via supervised dynamic topic modeling," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ser. WSDM '15. New York, NY, USA: ACM, 2015, pp. 453–458.

[9] Z. Jiang, X. Liu, and L. Gao, "Chronological citation recommendation with information-need shifting," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 1291–1300.

[10] S. Teller, *Data Visualization with D3.Js*. Packt Publishing, 2013.

[11] L. Tan, "Pywsd: Python implementations of word sense disambiguation (wsd) technologies (software)," https://github.com/alvations/pywsd.