Statement Generation Based on Big Data for Keyword Search

Qingqing Liu and Zhengyou Xia(✉)

College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
zhengyou_xia@nuaa.edu.cn

Abstract. Natural language generation (NLG) is the process of automatically generating a high-quality natural language text through a planning process based on some key information. Regular NLG generates sentences by analyzing grammatical and semantics, generating rules, and then organizing elements based on rules and heuristics. However, sentences generated by such methods are too strict, poorly scalable and difficult to adapt to the changing language style of human beings nowadays. Our goal is to generate smooth, personal, multi-sentence text for end users. This paper introduces a new NLG system, which can generate distinctive statements, and discard the knowledge of semantics, syntax etc., which are required by the original rule-based generation statements. This system turns out to be simple and efficient. We obtain required corpus from the network, and then use the idea of the search engine to find sentences from a large amount of data that matches the meaning of the keyword provided by users. Such generated sentences are more consistent with people's daily life. Finally, we apply our system in the web commentary domain, evaluating our system based on three criteria. The result shows that our system works well in this field and can continue to deepen.

Keywords: NLG · Web crawler · Lucene structure

1 Introduction

Natural language processing includes natural language understanding and natural language generation. Natural language generation is a branch of artificial intelligence and computational linguistics. The corresponding language generation system is a computer model based on language information processing. Contrary to natural language analysis, it starts from the abstract concept level by selecting and executing certain semantics and grammar rules to generate text [1]. Natural language generation is the process of deliberately constructing a natural language text in order to meet specified communicative goals. Most of the previous statement generation is based on rules or statistics. For people's language tends to be more colloquial now, generator based on these algorithms are too rigid and can not express emotion, so it is difficult to generate statements that meet the public's requirements. So, after research, we introduced a simple, keyword-based statement generation system. The system requires a large amount of corpus resources, using the search mechanism, based on the keyword

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 477–488, 2019.
https://doi.org/10.1007/978-3-030-32388-2_41

to perform a matching search in the corpus, and then feedback to the user's desired statement.

Review previous statement generation methods, usually use semantic grammar analysis. Through discovering the rules, generate some templates, and then use rules and some heuristic methods, putting words on the certain template's location to generate statements. There are also some better methods based on statistic like language model. In our network public opinion control, in the hope that the generated network reply content is in line with the spoken language of the Internet users, not stiff and with personal emotional colors. The statements generated by the previous methods are mechanical, without theme emotions, which can give people a clear feeling that it is a machine-generated statement that does not conform to the network terminology and our aesthetic orientation, so it is difficult to use it in some filed such as network public opinion control. Since the previous methods are difficult to fulfill our requirements for generated statements, in order to avoid generating mechanized statements, our idea is to use the Lucene framework and existing network statements to generate statements that conform to personal orientation.

In order to generate statements, we must first have corpus. We use web crawler technology to grab a large number of comments from popular websites such as Weibo and Zhihu. Then we build a keyword-based search mechanism using Lucene architecture. More details are illustrated in Sect. 3. After that, we conduct experiments to verify the effect of the system. Finally, we make a conclusion about our work, and based on that, we also make a plan for the future work.

2 Related Work

The generation of natural language requires some key information. Sometimes, the input is incomplete or inaccurate, which will affect the operation of the generator. In order to resist this problem, Knight and Hatzivassiloglou [2] suggested overcoming the knowledge acquisition bottleneck in generation by tapping the information inherent in textual corpora. Their experiments show that automatically acquired corpus-based knowledge greatly reduces the need for deep handcrafted knowledge. In their approach, lattice is used to instead a large number of phrases, which saving a lot of space. Although the lattice-based method is promising, it has some disadvantages. So Langkilde [3] presents a new statistical approach to sentence generation in which alternative phrases are represented as packed sets of trees, or forests, which are more compact than a grid. With the development of NLG, many scholars have proposed new NLG methods. Galanis and Androutsopoulos [4] present a natural language generation system—Naturalowl, which produces texts describing individuals or classes of owl ontologies in three steps: document planning, micro-planning, and surface realization. It can generate fluent and coherent multi-sentence texts for end-users, but Naturalowl was used mostly to describe cultural heritage objects. It has domain dependencies, so it has limited practicality. Samely, Cimiano et al. [5] develop a Natural Language Generation (NLG) system that converts RDF data into natural language text based on an ontology and an associated ontology lexicon. They apply this system to the field of cooking, and evaluating the fluency and sufficiency of this system, which works well.

Recently, Recurrent Neural Networks (RNNs) based approaches have shown promising performance in tackling the NLG problems. Mikolov [6] proposed a language model based on the recurrent neural network (RNNLM), which used for speech recognition. Zhang and Lapata [7] used RNN to create interesting Chinese poems; Wen et al. [8] recently proposed a recurrent neural network (RNN) for end-to-end learning generation based on Mikolov's RNNLM, then uses CNN and a backward RNNLM as rerankers to generate utterances. Because of the variety of human language styles, each NLG system has a targeted field. Here, we introduce a different statement generation system.

3 Automatic Generation of Statements Based on Keywords

We use web crawler technology to get comments from popular websites such as Weibo and Zhihu. Web crawler (also known as a Web spider) is a program that can autonomously collect web page content [9]. According to system structure and implementation technology, Web Crawler can be roughly divided into the following types: General Purpose Web Crawler, Focused Web Crawler, Incremental Web Crawler and Deep Web Crawler [10]. Based on our needs, we choose to use the simpler focused web crawler.

3.1 Focused Web Crawler

Focused Web Crawler, also known as Topical Crawler, is a web crawler that selectively crawls pages related to pre-defined topics. Compared with the general web crawler, the focus crawler only needs to crawl the page related to the theme, which greatly saves the hardware and network resources. The saved pages are also updated quickly due to the small number, and it also satisfies the needs of specific people for specific domain information [10].

Chen [11] improved the crawling strategy of the focused crawler system based on the Fish-Search algorithm and the Shark-Search algorithm. Although the accuracy of the crawling was slightly decreased, it could obtain a higher rate of discovery of the subject resources; Hailiang et al. [12] introduced a set of efficient crawler algorithm with less resources and customizable themes, which met the personalized needs of users; Zhang [13] combined the search engine supported by the web crawler and RSS information to learn from each other's strengths, and researched the RSS-based focused web crawler, which achieved good results in the university website group; Fang [14] and others used the link navigation technology based on the site page link structure to achieve efficient crawling of the topic information, and constructed a lightweight focused web crawler with low resource consumption, high data collection accuracy and control ability. A hybrid update mechanism is proposed to reduce the implementation complexity of incremental updates. Topic description, page filtering and link filtering are the focus of research on web crawlers. Compared with various types of improved focused web crawlers, we find that Fang Qiming et al.'s research is more suitable for crawling data in our research, so we study it as a reference for our crawling.

Since we crawled the data in order to reserve a large number of statements, did not need to analyze the data or for other uses. Therefore, we partially borrowed the ideas of the customizable focused web crawler for P2P search by Fang Qiming and others, and abandoned its theme customization features and other additional highlights. We make some simplifies on the basis of it, so that it can be more concise and efficient in our research. The general structural framework of our web crawling is shown in Fig. 1. Based on the consideration of the user's future use, we crawled the comment and grabbed the like number of the comment together. It turns out that the more comments' like number, the more influential of the comments. We have crawled 100,000 pieces of data from the Internet.

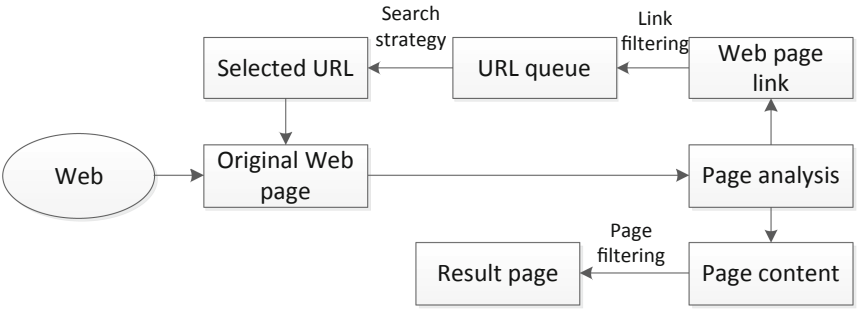

Fig. 1. The general system structure of the focused web crawler

3.2 Data Storage

When people express their opinions on Weibo, Zhihu and Tianya etc., they usually enhance their emotions by attaching some related expressions or pictures. When we crawl statements, these expressions and pictures are also being crawled, but their forms will be changed in this process. Irregular symbols will affect the normal expression of the statement. When the user uses it later, it will cause confusion. Therefore, when we grab the comment statement, we use the regularization expression to filter out these irregular expressions, only retain the text expression in the comment statement, and store it in the database. The specific process is shown in Fig. 2.

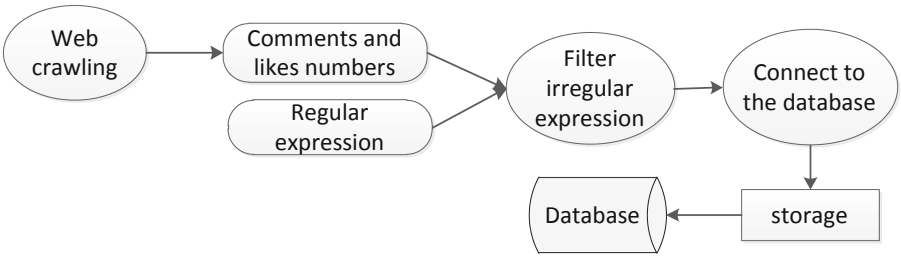

Fig. 2. Data filtering and storage

3.3 Search Framework Construction - The Use of Lucene

The search in this study is based on keyword search and borrowing the idea of search engine. For search engines, the most important thing is how to respond quickly and effectively to a large number of user retrieval needs. Therefore, when designing a search engine, the large amount of computation should be done as much as possible when the index is created. In the design of the search program, we should strive to improve the efficiency. Therefore, it is necessary to establish an efficient index library for documents. Lucene as an open source project, has triggered a huge response from the open source community since its inception. Programmers not only use it to build specific full-text search applications, but also integrate it into various system software to build Web applications, and even some commercial software use Lucene as the core of their internal full-text search subsystem.

Lucene [15] is a sub-project of the apache software foundation jakarta project. It has the characteristics of complete structure, clear hierarchy and good scalability, and provides relatively complete API documentation. At the same time, Lucene also has object-oriented and platform-independent—the core part of the system is encapsulated into an abstract class or interface. The specific platform implementation is designed as an abstract class or interface. It is a full-text retrieval engine architecture with low coupling, high efficiency and easy secondary development [16]. As an excellent full-text search engine architecture, Lucene adopts a highly optimized inverted index structure [17], which greatly improves the retrieval efficiency. Lucene implements full-text search in roughly two processes, index creation and index query. Figure 3 shows the entire search process.

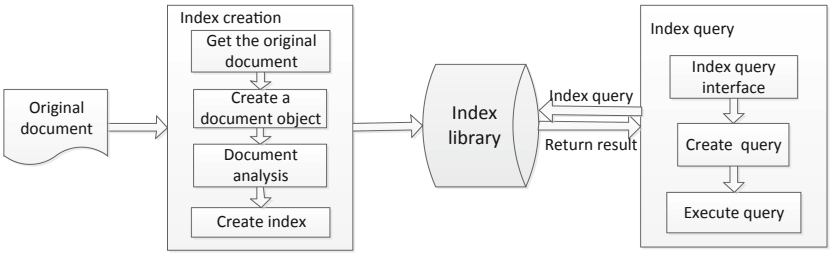

Fig. 3. Lucene full-text indexing process

Index Creation. *Get the Original Document.* The original document refers to the content to be indexed and searched, including web pages on the Internet, data in the database, files on the disk, and so on. Here, it refers to the data we crawled from the web page and saved in the database. We get the comment content and corresponding like number from the database as the original document.

Create Document Object. A document object represents a collection of fields. The domain of the document represents some metadata related to the document or the document. Based on our needs, we created two fields, which are stored separately and indexed as different domains of the document.

Document Analysis. After the original content is created as a document containing the domain, the content of the domain is parsed. Specifically, the text is split into a series of independent atomic elements called vocabulary units. Each vocabulary unit roughly corresponds to a "word" in the language. Because the language we want to parse is Chinese, and Lucene's own Chinese parser segmentation performance is not ideal, compare several different third-party Chinese word segmentation devices, taking into account the factors of the segmentation speed and the correct rate, we choose mmseg4j as our tokenizer.

Create Index. The traditional indexing method is find the content of the file according to the file and match the search keyword in the file content. This method is a sequential scanning method, data volume is large and search is slow. The inverted index structure creates an index that is indexed on the vocabulary unit, and the document is found by words. In order to achieve fast retrieval, Lucene uses an inverted index data structure. The specific steps to create an index are as follows: 1. Create an IndexWriter object.2. Create the document object. 3. Create a field object and add the field to the document object. 4. Write the document object to the index library using the indexwriter object. The index is created in this process. Index and document objects are written to the index library. 5. Close the IndexWriter object.

Index Query. *Index Query Interface.* The index query interface is an interface for the user to input keywords, complete the search and display the results.

Create Query. Before the user enters the keyword to perform the search, a query object need to be created. The query object can specify the document field and query keyword to be searched, and the query object generates a specific query syntax. There are two ways to create query object: use Query subclass provided by Lucene and use QueryParser to parse the query expression. Since QueryParser can be flexibly combined, including Boolean logic expressions, fuzzy matching, etc., so we choose use QueryParser to parse query expressions.

Execute Query. Lucene performs the search. Here we use Lucene's near real-time search. Near-real-time search can search the content that IndexWriter has not yet committed. The introduction of near real-time search enables the contents in the system to be indexed and searched faster, which reduces the overhead incurred when the system submits the index operation. Lucene implements near real-time search through the NRTManager class. Synchronization users after index update can get the latest index (IndexSearcher) in two ways: 1. Call the NRTManagerReopenThread object, which is responsible for tracking the changes of the index memory in real time, calling the maybeReopen method every time it changes, keep the latest index and open a new IndexSearcher object. IndexSearcher object which user need is NRTManager to obtain the SearcherManager object by calling the getSearcherManager method, then get the IndexSearcher object through the SearcherManager object and return it to user. After using it, call the release method of the SearcherManager to release the IndexSearcher object and close NRTManagerReopenThread; 2. Instead of using the NRTManagerReopenThread object, directly call NRTManager's maybeReopen method to get the latest IndexSearcher object to get the latest index. Here we use the second method.

4 Experiment

Natural language generation is domain dependent. NLG system invented by Sugiyama et al. [18] is oriented to the dialogue system, so the sentences they generate are used to simulate human dialogue, requiring sentences to be coherent and consistent with the topic of dialogue. Finally, they check their effectiveness by chatting with real users. The effect pursued by our system is to generate coherent sentences with personal opinions and emotional expressions and we made our experiment on the web comments filed. According to the needs of the field, we set three standards for the statements generated: smooth expression, clear theme and emotion. The following experiments are based on these three criteria to determine the effect of the generated statement. In order to better and more impartially evaluate the effectiveness of the system, we invited 7 netizens with online comment experience to participate in our experiment, based on there own inputs, making judgment on the three criteria of generation statements, and the opinion of the majority is taken as the final result.

4.1 Algorithm Evaluation

In the past, natural language generation was mostly based on rules and statistics. In order to verify the effectiveness of the proposed method, we performed a comparison experiment between different statement generation algorithms. Since our experimental field is web comments, the content format of the comment varies from person to person, and the syntax of each comment is different. The rule-based algorithm is not very good in our experiments, so we chose the statistics -based algorithms, two very popular language generation model—N-gram language model [19] and hidden Markov model.

N-Gram Language Model. N-gram is an algorithm based on statistical language models. The basic idea is to slide the contents of the text into a n-sized slide window, forming a sequence of n-length fragments of bytes. The occurrence frequency of all gram is counted and filtered according to the preset threshold to form a vector feature space of the text. Each gram in the list is a feature vector dimension. The model is based on the Markov assumption that in a piece of text, the appearance of the Nth word is only related to the first n-1 words, and is not related to any other words. Based on such a hypothesis, the probability of occurrence of each word in the text can be assessed, and the probability of the entire sentence is the product of the probability of each word.

Hidden Markov Model. The hidden Markov model is a statistical model for dealing with sequence problems. It describes a sequence of unobservable states randomly generated by a hidden Markov chain, and then randomly generating an observation from each state to produce an observed sequence. Hidden Markov model is a generation model that has been used by many scholars in natural language processing domain and has achieved good results.

Before the start of the experiment, some uncertain parameters of these two algorithms will affect the generation of the results. In order to carry out the experiment more efficiently, we select and determine the important parameters of the two models in

advance. Since we only select parameters that perform better, in order to carry out more efficiently, we only extract 10,000 data from our corpus for experimentation of parameter selection. For the N-gram model, the value of n will greatly affect the generation results, so we have chosen different n values for experimentation. For the hidden markov model, we selected the number of hidden states as the important parameters for the experiment. The results of the two experiments are shown in Figs. 4 and 5.

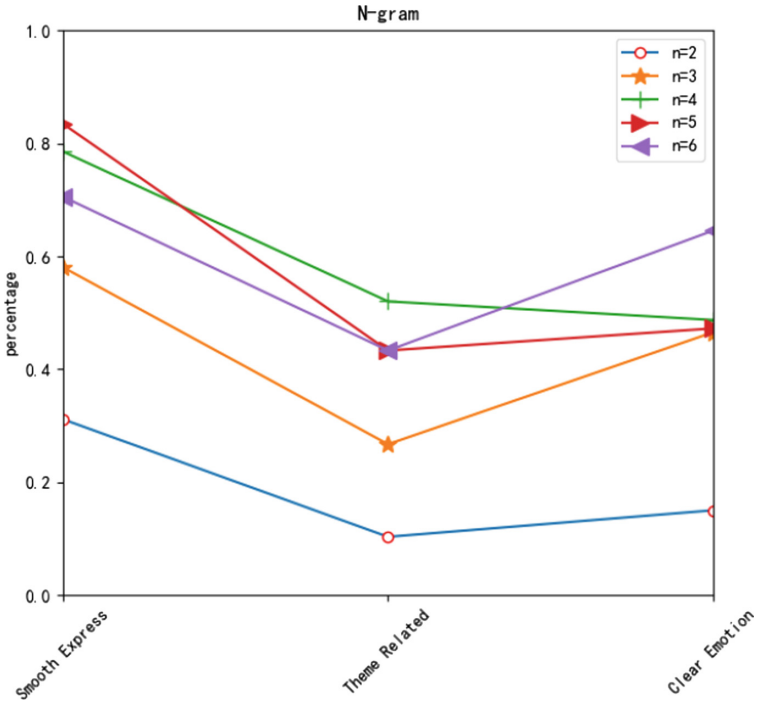

Fig. 4. Result for N-gram

Figures 4 and 5 show the experimental results of the two model parameter selections. For the N-gram model, we can see that when n is 5 the comprehensive effect is best, but in the experiment we find that the larger n is, the more repeated statements are generated, and when n is 5, more repeated statements are generated. But when n = 4, the repeated statements are relatively small and the model performs well on each standard, so we take n to 4 for the experiment. For the HMM model, we randomly took 5 numbers. The experimental results showed that the sentence coherence performed well, and other standards' performance were similar. After combining various aspects, we decided to set the number of hidden states to 10.

Table 1 is the result of our comparative experiment. It can be found that the 4-gram and HMM models perform well in the sentence coherence standard, but compared with the proposed algorithm, there is still a certain gap. Logically speaking, the statement we

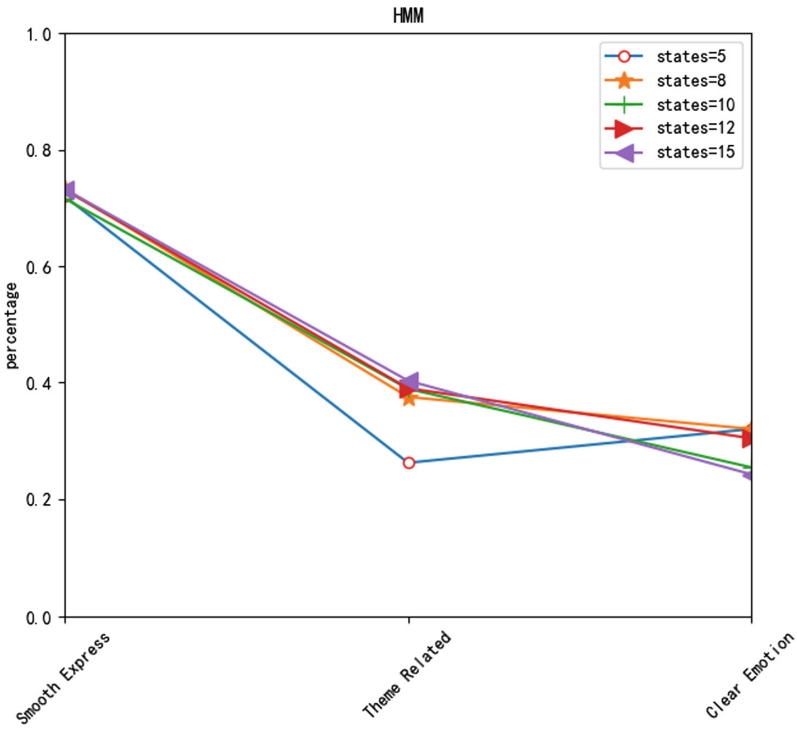

Fig. 5. Result for HMM

generated came from the reviewer, and the fluency should be better, but due to some network symbols and grammatical errors, our fluency did not reach 100%, but the effect is still very good. The performance of 4-gram and HMM model in terms of theme and emotion is not ideal, On the contrary, the model we presented is doing very well. Overall, our model performs well on all three standards. This means that our algorithm performs well in the generation of network statements, especially in capturing themes and expressing emotions. In order to more intuitively express the effects of our generator, we posted some sentences generated by our generator based on the keywords provided by the participants, as shown in Fig. 6.

Table 1. Comparison results of the three algorithms

Model	Smooth expression (%)	Theme related (%)	Clear emotion (%)
4-gram	70.73	39.02	36.10
HMM	72.50	38.33	33.33
Proposed	93.04	81.32	68.09

From the perspective of user usage, we also make experiments with the part-of-speech selection of keywords.

Fig. 6. Example for keyword-based statement generation

4.2 Part of Speech Experiment

For the part-of-speech experiment, we chose three important parts of speech: nouns, verbs and adjectives, and make 7 participants to randomly input 10 nouns, verbs and adjectives, then statistically calculate the different effects of the generated sentences with different part-of-speech.

As shown, the fluency of the generated statement is as good as it is in all parts of speech. In terms of the subject, the noun and verb performs well and noun performs better, if the user wants to generate a topic-related statement, we suggest that input is a noun. Finally, on the emotional side, adjective perform very well, so if the user tends to produce statements with some emotions, adjectives are recommended when entering

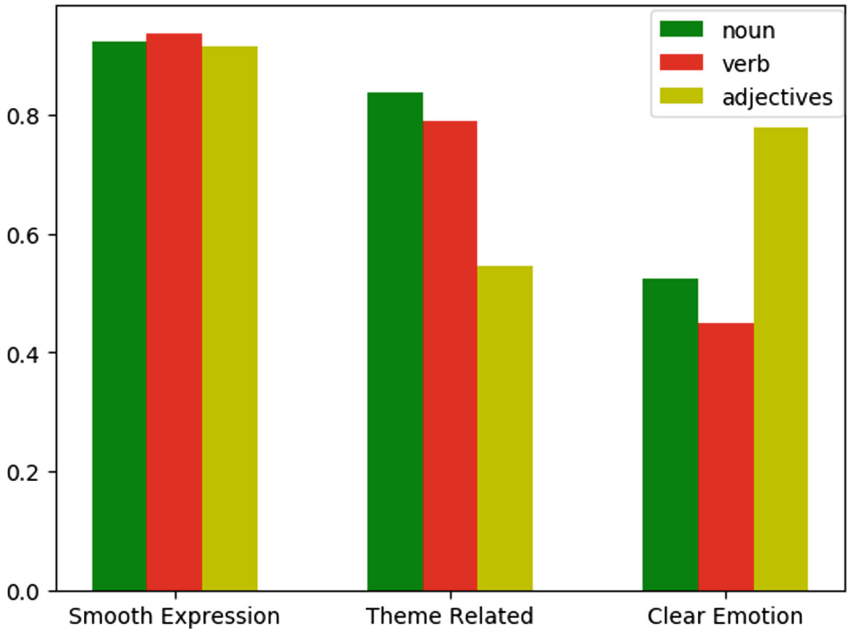

Fig. 7. Part of speech experiment results

keywords. But we can see form the figure, in general, nouns perform well in all aspects, so if the statements you want to generate cover a wider range, the noun is preferred (Fig. 7).

5 Conclusion and Future Work

We propose a keyword-based statement generation method that utilizes the idea of a search engine on big data. We uses the Lucene search framework to quickly generate a large number of statements related to keywords that user wants. Experiments show that sentences generated by our system are smooth, have clear themes and can express personal feelings, which is consistent with the effect of netizens' usual online reviews. At the same time, part-of-speech experiments show that users can get sentences that match individual inclinations by inputting keywords with different part of speech.

Although the statement generation of the system is simple and easy to use, but in our experiment, we also find that when the user uses the system to generate statements, statements that didn't match users' purpose will be generated, a lot of useless statements that are not related to the user's requirements are also generated. The reason for the phenomenon is that the corpus of the system is not perfect. The system relies heavily on the corpus, so the corpus is very important. And the content of the corpus needs to be updated continuously to generate statements that conform to the popular style, which is time consuming and labor intensive. So our future research will try to reduce the reliance on the corpus and generate statements in a more intelligent way.

References

1. Zhang, J., Chen, J.: Overview of natural language generation. Comput. Appl. Res. (08) (2006)
2. Knight, K., Hatzivassiloglou, V.: Two-level, many paths generation. In: Proceedings of ACL 1995 (1995)
3. Langkilde, I.: Forest-based statistical sentence generation. In: Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics, Seattle, WA (2000)
4. Galanis, D., Androutsopoulos, I.: Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system. In: Proceedings of 11th European Workshop on Natural Language Generation, pp. 143–146 (2007)
5. Cimiano, P., Nagel, D., Unger, C.: Exploiting ontology lexica for generating natural language texts from RDF data. In: Proceedings of 14th European Workshop on NLG, pp. 10–19 (2013)
6. Mikolov, T., Karafit, M., Burget, L., Cernocky, J., Khudanpur, S.: Recurrent neural network based language model. In: Proceedings on InterSpeech (2010)
7. Zhang, X., Lapata, M.: Chinese poetry generation with recurrent neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 670–680. Association for Computational Linguistics (2014)

8. Wen, T.-H., et al.: Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In: Proceedings of SIGdial. Association for Computational Linguistics (2015)
9. Chakrabarti, S., Van Den Berg, M., Dom, B.: Focused crawling: a new approach to topic specific web resource discovery. Comput. Netw. **31**(11), 1623–1640 (1999)
10. Sun, L., He, G., Wu, L.: Research on the Web Crawler. Comput. Knowl. Technol. (2010)
11. Chen, H.: Research and realization on focused crawler key technologies of vertical search engine. Master's thesis, Central China Normal University, Wuhan
12. Hailiang, Z., Li, S.: A customized focusing crawler. Electron. Technol. 51–54 (2009)
13. Zhang, R.: Research on RSS-based focused web crawler in college website group. Nanchang University (2012)
14. Fang, Q., Yang, G., Wu, Y.: Customized focused crawler for peer-to-peer Web search. J. Huazhong Univ. Sci. Technol. (Nat. Sci.) 153–157 (2007)
15. Hatcher, C.E., Gospodnetic, O., McCandless, M.: Lucene in Action, 2nd edn. Manning Publication, Stamford (2010)
16. Tang, H., He, Y., Xu, X.: Distributed parallel index based on Lucene. Comput. Technol. Dev. 123–126 (2011)
17. Liu, C., Guo, Q.: Analysis and research of web chinese retrieval system based Lucene, pp. 1051–1055. Computer Society (2009)
18. Sugiyama, H., Meguro, T., Higashinaka, R., Minami, Y.: Open-domain utterance generation for conversational dialogue systems using web-scale dependency structures. In: SIGDIAL, pp. 334–338 (2013)
19. Oh, A.H., Rudnicky, A.I.: Stochastic language generation for spoken dialogue systems. In: Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems, vol. 3, pp. 27–32. Association for Computational Linguistics (2000)