SCRIBE – An adaptive natural language generation engine

Vandita Shah
Student
Rajiv Gandhi Institute of Technology, Mumbai University

Sumer Shende
Student
Rajiv Gandhi Institute of Technology, Mumbai University

Vineet Trivedi
Student
Rajiv Gandhi Institute of Technology, Mumbai University

Rishabh Vig
Student
Rajiv Gandhi Institute of Technology, Mumbai University

Dnyaneshwar Dhangar
Professor
Rajiv Gandhi Institute of Technology, Mumbai University

***Abstract*—With the advent of electronic technology the modes of communication have expanded rapidly and the time frame to convey messages via the eclectic modes has reduced considerably. These two factors have significantly contributed to the coinage of the term 'Global Village'. An event occurring in one part of the world needs to be effectively communicated across the globe as it affects millions of people. The biggest hurdle under such circumstances is human delay. In this paper we propose a system called 'SCRIBE' which will substantially reduce human delay in conveyance of events and news, by automating the process of writing articles. Scribe is an Artificial Intelligence driven system which uses principles of Natural Language Processing for generating articles. Scribe studies articles pertaining to its domain to understand the language and format of articles that Scribe is expected to produce. Scribe then uses this knowledge base along with minimalistic input to generate articles. The goal is to design and build a system that will analyze, understand, and generate articles in a language that humans use naturally.**

I. Introduction

It is a repetitive and dynamically difficult errand for humans to process substantial volumes of raw data being perceived continuously from different domains, understand and interpret it while creating large amounts of text compositions which are syntactically correct and convey the logic preserving the original meaning.

The aim of scribe is to automate the task of article writing by building a system that learns the grammatical constructs of English language and uses this knowledge to formulate sentences and articles that are coherent and logically sound.

And in order to implement Scribe we are using the concepts of Natural Language Processing (NLP)[1]. Natural Language Processing[2] is a field of artificial intelligence that deals with how computers understand and manipulate text or speech. Some of the better known applications of NLP include Text-to-speech synthesizers which read text aloud for users and Machine translation systems which automatically render a document such as a web page in another language.

NLP primarily consists of two modules; Natural Language Understanding (NLU)[3] and Natural Language Generation (NLG)[4]. NLU helps the system understand Basic English language syntaxes and the structures of sentences by analyzing various text documents and finds links between commonly used words. Common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, parsing and semantic analysis are all phases of NLU.

Stanford CoreNLP is used for the purpose of Natural Language Understanding. The reasons to choose Stanford CoreNLP over other natural language processing libraries were:

- Stanford CoreNLP has a minimal conceptual footprint
- It is fast and easy to obtain linguistic annotations for any text using Stanford CoreNLP
- Stanford CoreNLP uses plain Java objects and provides a lightweight framework
- Minimal command line invocation
- Easy to set up
- Minimal code for analysis pipeline
- The annotation pipeline is extremely flexible and is not restricted to only a single sentence. It can be applied to multiple sentences or paragraphs.

Natural Language Generation involves the process of generating natural language from a machine representation system such as a knowledge base.

SimpleNLG is used for the purpose of NLG because it performs the tasks of orthography, morphology and simple grammar with high accuracy.

Orthography:

- Inserting appropriate whitespace in sentences and paragraphs.
- Absorbing punctuation. For example, generating the sentence "He lives in Washington D.C." instead of "He lives in Washington D.C.." (i.e., the sentence ends with a single period rather than two).
- Pouring – inserting line breaks between words rather than in the middle of a word, in order to fit text into rows.
- Formatting lists such as, "apples, pears, peaches and oranges."

Morphology:

- Handling inflected forms – that is, modifying/marking a word/lexeme to reflect grammatical information such as gender, tense, number or person.

Simple Grammar:

- Ensuring grammatical correctness.
- Creating well-formed verb groups (i.e., verb plus auxiliaries) such as "does not like".
- Allowing the user to define parts of a sentence or phrase and having SimpleNLG gather those parts together into an appropriate syntactic structure.

Most of the systems today have been developed in the field of either NLU or NLG. Scribe takes a step further by incorporating both these processes to truly understand and generate the English language.

For the implementation of this system, we require a large database that is used to store the entire word POS relations and linguistic annotations obtained from scanning diverse documents during the NLU phase. This data is stored in an unstructured relational form to make the data fetch more efficient.

The three key advantages of Scribe-produced text are scalability, tailor ability and consistency.

II. Methodology

Stanford CoreNLP

The first part of any artificial intelligence system is learning. Hence we feed learning data to the system. As the domain is restricted to cricket for experimental purpose the data fed to the system consists of cricket articles all stored in a .txt format. By feeding training data to Scribe we want Scribe to:

- Understand the English language, the structure of sentences and characteristics of each word,
- Understand the essence of a cricket article, its structure and flow,
- Develop a correct grammatical sense,
- Understand and establish links between words,
- Know the frequency, with which a particular word is used in a given context,
- Know the frequency, with which a particular word is used in association with another word,
- Be able to use the knowledge base in order to predict the most probable word given a particular scenario,

This is done using Stanford CoreNLP libraries. The execution flow of Stanford CoreNLP is as follows:

- Tokenization (tokenize)
- Sentence Splitting (ssplit)
- Part of Speech Tagging (pos)
- Morphological Analysis (lemma)
- Named Entity Recognition (ner)
- Syntactic Parsing (parse)
- Coreference Resolution (dcoref)
- Other Annotators (gender, sentiment)

Tokenization (tokenize): It is the process of breaking up some text or a sequence of characters while eliminating certain characters like punctuation marks and whitespace in a defined document unit, into pieces called tokens. For example:

Input: Sachin, Sehwag, Ganguly and Dravid are all great cricketers.

Output: Sachin | Sehwag | Ganguly | and | Dravid | are | all | great | cricketers

Sentence Splitting (ssplit): Splits a sequence of tokens into sentences

Part of Speech Tagging (pos): Labels the words and tokens with their respective part of speech. The labelling of the words and tokens with their respective part of speech is complex process. While labelling the definition of the word and the usage or context of the word in that particular sentence or text, needs to be considered. The part of speech is more detailed than just simple 'verb', 'noun', 'adverb' etc. for example 'noun-plural'

Morphological analysis (lemma): Lemmatisation is the process of deriving the lemma of a given word from its various inflected forms. The lemma of a word is the base form of the word. For example the verb 'to score' has many inflected forms like score, scores, scoring and scored. Lemmatisation is different from stemming in the crucial aspect, that lemmatisation takes into account the context and the meaning of a given word. Stemming is the process of removing the ends of words optimistically assuming that the end product will be the lemma of the given word.

For example the words 'better' and 'best' have 'good' as its lemma. Stemming operates on the basic principle of chopping without knowledge or context of the words and hence will never be able to obtain the lemma of the words ‘better’ and ‘best’. However lemmatisation incorporates context and meaning and hence can obtain the lemma for ‘better’ and ‘best’.

Named Entity Recognition (ner): Recognizes named (PERSON, LOCATION, ORGANIZATION, MISC) and numerical (MONEY, NUMBER, DATE, TIME, DURATION, SET) entities.

Syntactic Parsing (parse): Provides full syntactic analysis, including both constituent and dependency representation, based on a probabilistic parser.

Coreference Resolution (dcoref): It is the process of identifying all expressions that refer to the same entity. The output of Coreference Resolution is used in several higher level NLP tasks like document summarization, question answering, and information extraction.
For example:

“I think Sachin played well and he scored a lot of runs”, she said.

Gender Identification: Adds gender information to names

Sentiment Analysis: Sentiment analysis is used to determine the sentiment of a given text i.e. to determine whether a piece of writing is positive, negative or neutral. It uses deep learning. The nodes of the binarized tree of each sentence are given a sentiment score.

Once the training data is fed to Scribe, we obtain the required linguistic annotations using the Stanford CoreNLP[5] libraries. We then move on to our next step – storing the obtained linguistic annotations.

MongoDB

The linguistic annotations need to be stored in a database. The database will work as a data repository of analyzed cricket articles. The database shall aid Scribe in its task of writing articles. For the purpose of storing data, we analyzed RDBMS and NoSQL databases and finally decided to use MongoDB[6].

MongoDB is a document database, wherein a database consists of several collections and each collection contains the various documents.

The primary reason for using MongoDB is that it is schema less. Each document can have its own structure. Each article is unique and hence the number of fields, content and size associated with each article will be different. Thus a standard format like that of a RDBMS will either fail or will be very inefficient for the proposed system. However with the flexibility offered by the schema less MongoDB[7] aligns perfectly with the needs of Scribe. MongoDB does not have the complex concept of JOINs and hence querying and retrieving information becomes much simpler. MongoDB Uses internal memory for storing the working set, enabling faster access of data. This improves the overall efficiency of the proposed system.

MongoDB is highly scalable and the data is stored in the form of JSON style documents. MongoDB retains certain properties of SQL like Ad hoc queries and indexing which are favorable and familiar thus easy to work with.

MongoDB focusses on high availability. It has auto-sharding and performs replication. Text search is integrated and queries are JavaScript expressions. MapReduce can be used for batch processing of data and aggregation operations.

For example if the sentence to be processed by Scribe using the Stanford CoreNLP libraries is:

‘Sachin scored a spectacular 117 runs. He was declared the man of the match.’

Its POS Tagging is:

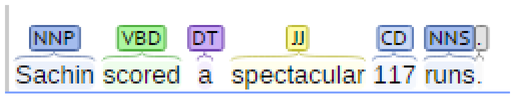

Figure1: Part of Speech Tagging of the given sentence.

Then the POS tagging annotation will be stored in the database as:

```
{
  "sentences": [
    {
"parse": "(ROOT\n  (S\n    (NP (NNP Sachin))\n    (VP (VBD scored)\n      (NP (DT a) (JJ spectacular) (CD 117) (NNS runs)))\n    (. .)))"
    }
  ]
}
```

Once the linguistic annotations have been stored in MongoDB we move on to the next step – writing the article.

SimpleNLG

In order to write an article Scribe needs minimalistic input. The criterion for input is that it should be easily and rapidly available. For the purpose of cricket our input is the statistical sheet of the cricket match. It is available immediately after the match, which satisfies the rapid availability criterion. It can easily be obtained from the internet or any other source.

The structure for sports articles is well defined. It consists of a lead, bridge and a body. The lead answers crucial questions like the teams playing the match, where the match was played, which team won, what were the final scores of both the teams and it also mentions exemplary performance by any player. The bridge acts like a link between the lead and the body by providing the background of the match and its constituent teams. The body provides the remaining details of the match.

Scribe uses the information provided by the statistical sheet to obtain answers to these questions. Scribe reads the player names and their respective performance. It compares the performance of the players to threshold values and also to the performance of other players in that match. Other details like venue, teams, scores of both the teams, winning team and man of the match are also obtained from the statistical sheet. As sports articles predominantly follow the aforementioned fixed format, data is fed to a SimpleNLG based program in accordance with the format.

SimpleNLG[8] is a library and not an application. SimpleNLG library is used to create a program that will generate grammatically correct sentences in English. The library consists of classes which allow the user to define the subject, the object, the verb and additional compliments to the sentence. For example if want to build a sentence "Sachin scored 10 runs" we need to feed SimpleNLG with the subject: 'Sachin'; the object: '10 runs' and the verb: 'score'. The object and subject will be obtained from the statistical sheet and feeding it to SimpleNLG based program is a case of simple mapping. The verb is obtained by running statistical learning algorithms on the repository of articles and the linguistic annotations stored in the database.

Once the content of the sentence has been fed to the SimpleNLG based program, it will assemble the parts of the sentence into a grammatically correct form and output the result. Once all the sentences have been created SimpleNLG classes are used to organize them into a logically coherent article which is the final output of the Scribe.

III. Conclusion

Scribe can expand its data sources. It can be made capable enough to draw raw data from sources like RSS feeds of multiple website, social media plug roots, blogging stations, news tickers etc. By expanding its sources of data, the richness and quantity of data can be increased. For this, Scribe needs to be able to read text from different formats like XML, LateX along with grabbing the correct unformatted text from html documents.

Scribe has been implemented for producing articles for a cricket match. However its use is not bounded to only cricket. Scribe can easily learn how to write articles on other sports like football, hockey, baseball etc. For this Scribe needs to be fed training data in the form of articles from the respective sports and it will learn the structure and details of each sport from the training data itself. With minimal changes made to the Scribe it will be ready to use for any other sport.

However the most important use of Scribe is the coverage of natural disasters. As stated earlier one of the key problems that this system was designed to tackle and eliminate was human delay in communication. During natural disasters chances of human delay are high and the need to communicate the disaster and its consequences is of paramount importance. Abnormal readings from instruments like seismometers, Saffir-Simpson scale and stream gauges indicating a natural disaster like earthquake, hurricane or flood respectively should be communicated to Scribe. Scribe can then write a piece informing the people of a natural disaster. Scribe can be one of the fastest modes of communicating the occurrence of a natural disaster, the area it has occurred in and its possible impact.

Incorporating the above suggestions in scribe will make the proposed system a multifaceted all-purpose journalist capable of writing an article on any given topic.

References

[1] Abhimanyu Chopra, Abhinav Prashar, and Chandresh Sain, 2013, "Natural Language Processing", *International Journal of Technology Enhancements and Emerging Engineering Research*.

[2] E. Cambria and B. White, 2014, "Jumping NLP Curves: A Review of Natural Language Processing Research", *IEEE Computational Intelligence Magazine*.

[3] J. M. Huerta and D. Lubensky, 2003, "Graph-based representation and techniques for NLU application development", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03).*

[4] N. Ito and M. Hagiwara, 2011, "Natural language generation using automatically constructed lexical resources" *The 2011 International Joint Conference on Neural Networks (IJCNN).*

[5] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard and David McClosky, 2014, "The Stanford CoreNLP Natural Language Processing Toolkit", *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations.*

[6] Zachary Parker, Scott Poe and Susan V. Vrbsky, 2013, "Comparing NoSQL MongoDB to an SQL DB", *Proceedings of the 51st ACM Southeast Conference*.

[7] Veronika Abramova and Jorge Bernardino, 2013, "NoSQL databases: MongoDB vs cassandra", *Proceedings of the International C* Conference on Computer Science and Software Engineering*.

[8] Albert Gatt and Ehud Reiter, 2009, "SimpleNLG: A realisation engine for practical application", *Proceedings of the 12th European Workshop on Natural Language Generation.*