ICE: Idiom and Collocation Extractor for Research and Education

Vasanthi Vuppuluri
Verizon Labs
375 W Trimble Rd,
San Jose, CA 95131, USA
vuppulurivasanthi@gmail.com

Shahryar Baki, An Nguyen, Rakesh Verma
Computer Science Dept.
University of Houston
Houston, TX 77204, USA
shahryar@cs.uh.com
anqnguyen@outlook.com
rverma@uh.edu

Abstract

Collocation and idiom extraction are well-known challenges with many potential applications in Natural Language Processing (NLP). Our experimental, open-source software system, called ICE, is a python package for flexibly extracting collocations and idioms, currently in English. It also has a competitive POS tagger that can be used alone or as part of collocation/idiom extraction. ICE is available free of cost for research and educational uses in two user-friendly formats. This paper gives an overview of ICE and its performance, and briefly describes the research underlying the extraction algorithms.

1 Introduction

Idioms and collocations are special types of phrases in many languages. An *idiom* is a phrase whose meaning cannot be obtained compositionally, i.e., by combining the meanings of the words that compose it. Collocations are phrases in which there is a semantic association between the component words and some restrictions on which words can be replaced and which cannot. In short, *collocations* are arbitrarily restricted lexeme combinations such as *look into* and *fully aware*.

Many scientists from diverse fields have worked on the challenging tasks of automated collocation and idiom extraction, e.g., see (Garg and Goyal, 2014; Seretan, 2013; Verma and Vuppuluri, 2015; Verma et al., 2016) and the references contained therein, yet there is no multi-purpose, ready-to-use, and flexible system for extracting these phrases. Collocation and its special forms, such as idioms, can be useful in many important tasks, e.g., summarization (Barrera and Verma, 2012), question-answering (Barrera et al., 2011), language translation, topic segmentation, authorial style, and so on. As a result, a tool for these tasks would be very handy.

To tackle this void, we introduce a feature-rich system called ICE (short for Idiom and Collocation Extractor), which has two versions: one is flexible and pipelined seamlessly for research purposes as a component of a larger system such as a question answering system, and the second as a web-based tool for educational purposes. ICE has a modular architecture and also includes a POS tagger, which can be used alone or as part of collocation or idiom extraction. An experiment with the CoNLL dataset shows that ICE's POS tagger is competitive against the Stanford POS tagger. For ease of use in research, we provide ICE as a python package.

For collocation extraction, ICE uses the IR models and techniques introduced by (Verma et al., 2016). These methods include: dictionary search, online dictionaries, a substitution method that compares the Bing hit counts of a phrase against the Bing hit counts of new phrases obtained by substituting the component words of the phrase one at a time to determine the "adherence factor" of the component words in a collocation, and two methods that try to measure the probability of association of the component words again using hit counts. In (Verma et al., 2016), the authors created a gold-standard dataset of collocations by taking 100 sentences at random from the Wiki50 dataset and manually annotating them for collocations (including idioms) using eight volunteers, who used the Oxford Dictionary of Collocations and Oxford Dictionary of Idioms. Each sentence was given to two annotators, who were given 25 sentences each for annotation, and their work was checked and corrected afterwards by two other people. In creating this dataset, even with the assistance of dictionaries, human performance

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 108–111

varied from an F1-score of about 39% to 70% for the collocation task. A comparison showed that their combination schemes outperformed existing techniques, such as MWEToolkit (Ramisch et al., 2010) and Text-NSP (Banerjee and Pedersen, 2003), with the best method achieving an F1-score of around 40% on the gold-standard dataset, which is within the range of human performance.

For idiom extraction, ICE uses the semantics-based methods introduced by (Verma and Vuppuluri, 2015). The salient feature of these methods is that they use Bing search for the definition of a given phrase and then check the compositionality of the phrase definition against combinations of the words obtained when a define word query is issued, where the word belongs to the phrase. If there is a difference in the meaning, that phrase is considered an idiom. In (Verma and Vuppuluri, 2015), authors showed that their method outperforms AMALGr (Schneider et al., 2014). Their best method achieved an F1-score of about 95% on the VNC tokens dataset.

Thus, ICE includes extraction methods for idioms and collocations that are state-of-the-art. Other tools exist for collocation extraction, e.g., see (Anagnostou and Weir, 2006), in which four methods including Text-NSP are compared.

2 ICE - Architecture and Algorithms

As ICE's algorithms are based on Bing search, users must provide a valid user id for the Bing API. ICE receives a list of sentences as an input and outputs a list of all collocations and idioms. It first splits the input sentences using NLTK sentence tokenizer, then generates n-grams and part of speech tags. ICE's n-gram generator takes care of punctuation marks and has been shown to be better than NSP's n-gram generator. Finally, the output n-grams are given to the collocation and idiom detection algorithms. Collocation and idiom extraction has been done by the algorithm given by (Verma et al., 2016)[1] and (Verma and Vuppuluri, 2015). For part of speech tagging we combined NLTK's regex tagger with NLTK's N-Gram Tagger to have a better performance on POS tagging. We compared our tagger with Stanford POS tagger (Manning et al., 2014) on the CoNLL dataset.[2] The accuracy of our tagger is 92.11%, which is slightly higher than 91.19%, the accuracy of the Stanford tagger on the same corpus.

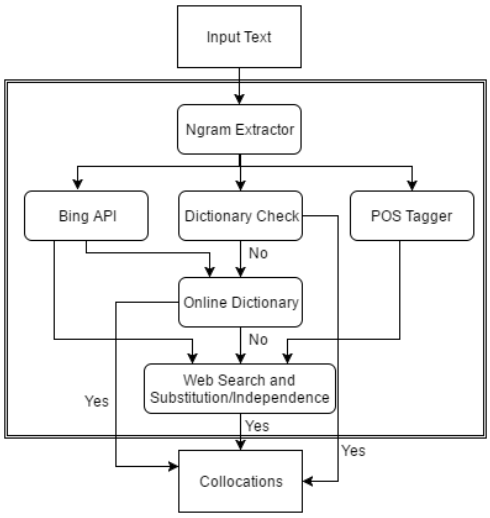

Figure 1: Collocation extractor diagram

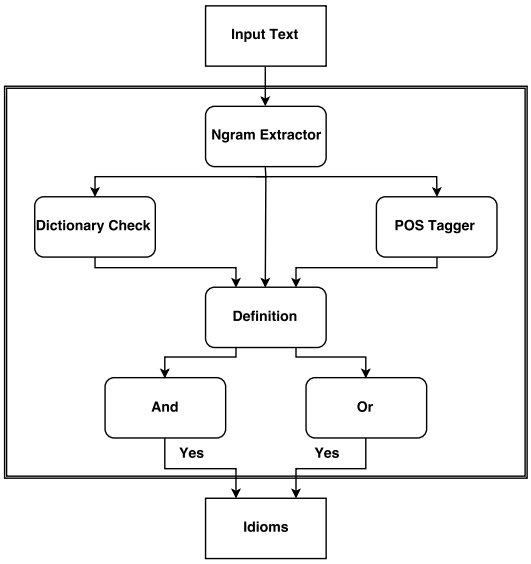

Figure 2: Idiom extractor diagram

Collocation/Idiom Extractor. The collocation extraction technique combines different methods in a pipeline in order to increase precision. Figures 1 and 2 show the idiom and collocation extraction system architectures separately. As shown in the diagrams, there are two methods for identifying idioms (called And and Or) and four different methods for identifying collocations including: offline dictionary search, online dictionary search, web search and substitution, and web search and independence.

For collocations, ICE pipelines the first and second methods, then pipelines them with the third or the fourth method (both options are available

[1] http://www2.cs.uh.edu/~rmverma/paper_216.pdf

[2] Available at http://www.cnts.ua.ac.be/conll2003/ner/

in the code). These methods are connected sequentially. This means that if something is considered as a collocation in one component, it will be added to the list of collocations and will not be given to the next component (yes/no arrows in the diagram). Table 1 shows the description of each component in collocation extractor diagram.

The *Ngram Extractor* receives all sentences and generates n-grams ranging from bigrams up to 8-grams. It uses NLTK sentence and word tokenizers for generating tokens. Then, it combines the generated tokens together taking care of punctuation to generate the n-grams.

Dictionary Check uses WordNet (Miller, 1995) as a dictionary and looks up each n-gram to see if it exists in WordNet or not (a collocation should exist in the dictionary). All n-grams that are considered as non-collocations are given to the next component as input.

The next component is *Online Dictionary*. It searches online dictionaries to see if the n-gram exists in any of them or not. It uses Bing Search API [3] to search for n-grams in the web.

Web Search and Substitution is the next component in the pipeline. This method uses Bing Search API to obtain hit counts for a phrase query. Then each word in the n-gram will be replaced by 5 random words (one at the time), and the hit counts are obtained. At the end, we will have a list of hit counts. These values will be used to differentiate between collocations and non-collocations.

The last component in the pipeline of collocation extraction is *Web Search and Independence*. The idea of this method is to check whether the probability of a phrase exceeds the probability that we would expect if the words are independent. It uses hit counts in order to estimate the probabilities. These probabilities are used to differentiate between collocations and non-collocations.

When running the collocation extraction function, one of the components should be selected out of the third and fourth ones.

The *Idiom Extractor* diagram is relatively simpler. Given the input n-gram, it creates n + 1 sets. The first contains (stemmed) words in the meaning of the phrase. The next n sets contain stemmed word in the meaning of each word in the n-gram. Then it applies the set difference operator to n pairs containing the first set and each of the n sets. The Or subsystem considers a phrase as an idiom if at least one word survives one of the subtractions (union of difference sets should be non-empty). For the And, at least one word has to exist that survived every subtraction (intersection of difference sets should be non-empty)

Performance. ICE outperforms both Text-NSP and MWEToolkit. On the gold-standard dataset, ICE's F1-score was 40.40%, MWE-Toolkit's F1-score was 18.31%, and Text-NSP had 18%. We also compared our idiom extraction with AMALGr method (Schneider et al., 2014) on their dataset and the highest F1-score achieved by ICE was 95% compared to 67.42% for AMALGr. For detailed comparison of ICE's collocation and idiom extraction algorithm with existing tools, please refer to (Verma et al., 2016) and (Verma and Vuppuluri, 2015).

Sample Code. Below is the sample code for using ICE's collocation extraction as part of a bigger system. For idiom extraction you can use IdiomExtractor class instead of collocationExtractor.

```
>> input=["he and Chazz duel
 with all keys on the line."]
>>from ICE import
 CollocationExtractor
>>extractor =
 CollocationExtractor.
 with_collocation_pipeline(
 "T1" , bing_key = "Temp",
 pos_check = False)
>> print(extractor.
 get_collocations_of_length(
  input, length = 3))
>> ["on the line"]
```

Educational Uses. ICE also has a web-based interface for demonstration and educational purposes. A user can type in a sentence into an input field and get a list of the idioms or collocations in the sentence. A screen-shot of the web-based interface is shown in Figure 3.[4]

3 Conclusion

ICE is a tool for extracting idioms and collocations, but it also has functions for part of speech

[3] http://datamarket.azure.com/dataset/bing/search

[4] The web interface is accessible through https://shahryarbaki.ddns.net/collocation/

Table 1: Components of Collocation Extraction Subsystem of ICE

Component	Description
Ngram Extractor	Generates bigram up to 8gram
Dictionary Check	Look up ngram in dictionary (Wordnet)
Online Dictionary	Look up ngram in online dictionaries (Bing)
Web Search and Substitution	Hitcount for phrase query and 5 generated queries by randomly changing the words in the ngram
Web Search and Independence	Probability of a phrase exceeds the probability that we would expect if the words are independent

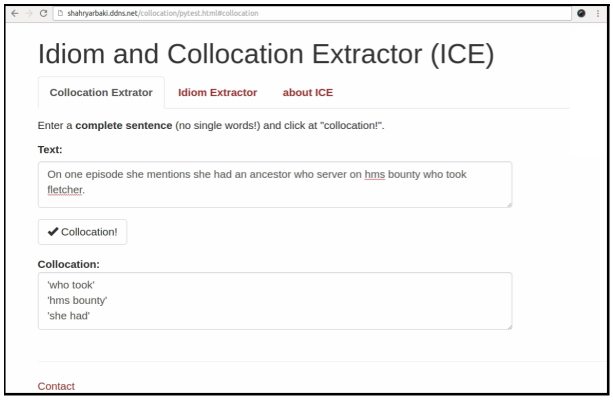

Figure 3: Screen-shot of the online version of ICE

tagging and n-gram extraction. All the components of the ICE are connected as a pipeline, hence every part of the system can be changed without affecting the other parts. The tool is available online at `https://github.com/shahryarabaki/ICE` as a python package and also at a website. The software is Licensed under the Apache License, Version 2.0.

References

Nikolaos K. Anagnostou and George R. S. Weir. 2006. Review of software applications for deriving collocations. In *ICT in the Analysis, Teaching and Learning of Languages, Preprints of the ICTATLL Workshop 2006*, pages 91–100.

Satanjeev Banerjee and Ted Pedersen. 2003. The design, implementation, and use of the ngram statistics package. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 370–381. Springer.

Araly Barrera and Rakesh Verma. 2012. Combining syntax and semantics for automatic extractive single-document summarization. In *CICLING*, volume LNCS 7182, pages 366–377.

Araly Barrera, Rakesh Verma, and Ryan Vincent. 2011. Semquest: University of houston's semantics-based question answering system. In *Proceedings of the Fourth Text Analysis Conference, TAC 2011, Gaithersburg, Maryland, USA, November 14-15, 2011*.

Chitra Garg and Lalit Goyal. 2014. Automatic extraction of idiom, proverb and its variations from text using statistical approach. *An International Journal of Engineering Sciences*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010. Multiword expressions in the wild?: the mwetoolkit comes in handy. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 57–60. Association for Computational Linguistics.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the mwe gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Violeta Seretan. 2013. A multilingual integrated framework for processing lexical collocations. In *Computational Linguistics - Applications*, pages 87–108. Springer - Netherlands.

Rakesh Verma and Vasanthi Vuppuluri. 2015. A new approach for idiom identification using meanings and the web. *Recent Advances in Natural Language Processing*, pages 681–687.

Rakesh Verma, Vasanthi Vuppuluri, An Nguyen, Arjun Mukherjee, Ghita Mammar, Shahryar Baki, and Reed Armstrong. 2016. Mining the web for collocations: Ir models of term associations. In *International Conference on Intelligent Text Processing and Computational Linguistics*.