Concept Concordancer: A Visualization Tool for Corpus Analysis

Raymund Micoh Alvarez, Jonghyun Choi, Kimberly Anne Martinez, Randolph Nathaniel Yu, and Nathalie Rose Lim-Cheng*
De La Salle University
**Corresponding Author: Nathalie.lim@dlsu.edu.ph*

Abstract: Corpus analysis is performed by linguists manually to gather information about a language using large collections of texts. Among the tools that can be used for this is a concordancer. The concordance provides a convenient way of lining up all the instances of a certain keyword. This allows the linguist to study how a keyword was used in the given language. However, what is missing in current concordancers is the ability to search and list the instances of the keyword including the related words (like synonyms) within the text. Our research addresses this gap by designing and implementing a concept concordance that can find and show visualizations of related concepts, not just actual search keywords, within a corpus. Stanford CoreNLP was used to process the text and provide annotations while WordNet and ConceptNet were used as resources for concept finding. The Concept Concordancer is able to generate a list of concepts, serve as an annotating tool, show visualization of concordances and patterns, provide concept suggestions and store concepts in a dictionary. We performed tests on our concept concordance using a general corpus and found that ConceptNet and WordNet alone is not sufficient, as the concordance performs better only when additional concepts are added during customization.

Key Words: concordancer, computational linguistics, natural language processing

1. INTRODUCTION

Concordancers, which are one of the main tools that allow researchers in the field of language and linguistics to study text based on its usage, are available to create a more intuitive visualization of texts. According to McEnery and Hardieg (2014), it is undoubtedly the single most important tool available to a corpus linguist as it is able to search a corpus and retrieve from it a specific sequence of characters in any length. An example would be when a certain keyword is searched, the tool will display all sentences where the keyword exists. The keyword and the words following it will be highlighted and vertically aligned for the researchers to utilize in analyzing which words mostly collocate to the keyword. Most concordancers rely on implementing a search for part of the word, which makes it easier to find (Tribble and Glyn, 1997).

Examples would be words ending in -ing or words beginning with -dis. While corpus search tools such as concordancers are vital in expanding the range of research questions, it can also limit and define what can be done with a corpus: hence, there are many different concordancers with varying purposes. An example would be a syntactic concordancer that highlights the multi-word expressions (MWE's) then groups them into syntactically-homogenous classes ranked by the strength of association of the words (Seretan and Wehrli, 2010). Other concordancers can be used for semantic analysis which combines both a textual corpus and lexicon to link the text with its appropriate sense in a lexicon (Miller, et al., 1993).

Although there are many variations of concordancers, it is noticeable that there is a missing feature of concordancers that can search for concepts within a corpus. A concept is an umbrella term representing an idea that is depicted by a set of lexical terms. For a sample concept "book", some lexical terms include publication, tome, and reserve. Currently, concordancers can only return usage instances of the searched keyword but not terms representing a concept or idea (i.e., related words).

There is a large benefit in creating a Concept Concordancer that can be used by linguists to find words that represent concepts and not just specific words or synonyms. Moreover, having a concept concordance would allow developers of rule-based or pattern-based information extraction systems and ontology population systems in determining seed rules for retrieving relevant data from a corpus.

This paper discusses our Concept Concordancer tool. In section 2, we present the system architecture and the process of the system. In section 3, we discuss the test results. Lastly, we present our conclusion and future work in Section 4.

2. SYSTEM ARCHITECTURE

Our Concept Concordancer system is composed of four major modules. Shown in Figure 1 are the modules File Manager module, Annotations Module, Concept Concordancer API, Visualization Module, the Natural Language Processing (NLP) tool Stanford CoreNLP, and the resources WordNet, ConceptNet, and a concept dictionary.

The process of the system starts when the user uploads annotated or unannotated files to the system to be processed as a corpus. This is done in the File Manager Module. Unannotated texts go through data processing to be annotated then these files are merged into one corpus file. After the corpus is generated, the user may modify the annotation tags and create new annotations through the Annotations Module. The user can also choose to upload a concept dictionary from previous sessions to be used as a resource by the system.

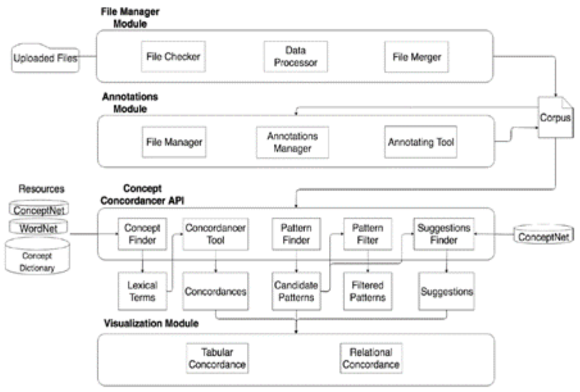


Fig. 1. Concept Concordancer System Architecture

When a concept search occurs, the Concept Finder within the Concept Concordancer API utilizes the resources, WordNet, ConceptNet, and the concept dictionary, to find related lexical terms to the keyword. These lexical terms are passed on to the Concordancer Tool which then searches the corpus for instances of the lexical terms. The resulting subcorpus containing the concordances is forwarded to the Pattern Finder. The Pattern Finder determines the candidate patterns based on the concordances. Using these candidate patterns and the "relatedto" relation from ConceptNet, concept suggestions are generated for the user to have an option to augment the concept list.

The Visualization Module displays the results of the Concordancer Tool and the Pattern Finder tool.

The following subsections provide more details on each of the modules.

2.1 File Manager Module

The File Manager Module processes input files from the users, annotates raw text files using Stanford CoreNLP and converts them to <tiger2/> files, and merges all the uploaded files into one corpus file for use in other modules. The File Manager Module consists of the File Checker, the Data Processor, and the File Merger.

When the user uploads a file to the system, the File Checker checks the uploaded file and verifies that it is in a text or <tiger2/> XML format. Unannotated files go through the Data Processor to be annotated while annotated files are directly passed to the File Merger. Figure 2 shows a sample text file containing an excerpt of the novel The History of Tom Jones, a Foundling by Henry Fielding named historyoftom.txt. Since the text file is unannotated, it is passed to the Data Processor.

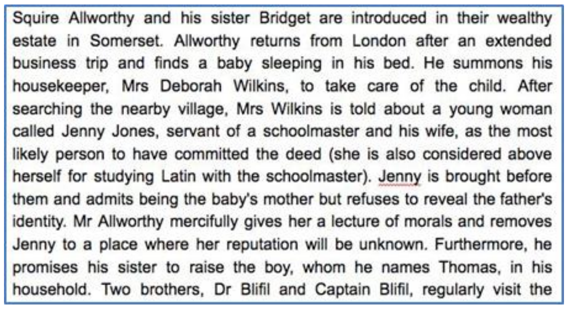

Squire Allworthy and his sister Bridget are introduced in their wealthy estate in Somerset. Allworthy returns from London after an extended business trip and finds a baby sleeping in his bed. He summons his housekeeper, Mrs Deborah Wilkins, to take care of the child. After searching the nearby village, Mrs Wilkins is told about a young woman called Jenny Jones, servant of a schoolmaster and his wife, as the most likely person to have committed the deed (she is also considered above herself for studying Latin with the schoolmaster). Jenny is brought before them and admits being the baby's mother but refuses to reveal the father's identity. Mr Allworthy mercifully gives her a lecture of morals and removes Jenny to a place where her reputation will be unknown. Furthermore, he promises his sister to raise the boy, whom he names Thomas, in his household. Two brothers, Dr Blifil and Captain Blifil, regularly visit the

Fig. 2. Excerpt of History of Tom Jones stored in a text file

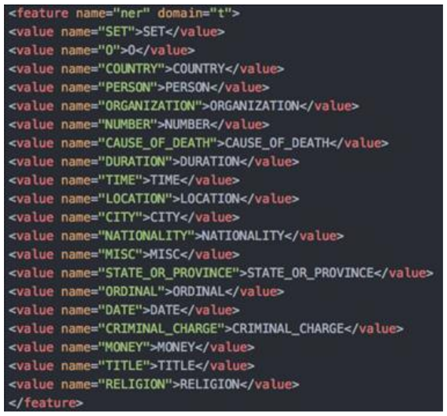

```xml
<feature name="ner" domain="t">
<value name="SET">SET</value>
<value name="O">O</value>
<value name="COUNTRY">COUNTRY</value>
<value name="PERSON">PERSON</value>
<value name="ORGANIZATION">ORGANIZATION</value>
<value name="NUMBER">NUMBER</value>
<value name="CAUSE_OF_DEATH">CAUSE_OF_DEATH</value>
<value name="DURATION">DURATION</value>
<value name="TIME">TIME</value>
<value name="LOCATION">LOCATION</value>
<value name="CITY">CITY</value>
<value name="NATIONALITY">NATIONALITY</value>
<value name="MISC">MISC</value>
<value name="STATE_OR_PROVINCE">STATE_OR_PROVINCE</value>
<value name="ORDINAL">ORDINAL</value>
<value name="DATE">DATE</value>
<value name="CRIMINAL_CHARGE">CRIMINAL_CHARGE</value>
<value name="MONEY">MONEY</value>
<value name="TITLE">TITLE</value>
<value name="RELIGION">RELIGION</value>
</feature>
```

Fig. 3. Features from historyoftom.tiger2

The Data Processor transforms unannotated text/s into annotated text/s. Stanford CoreNLP functions are invoked and the resulting annotations are stored using the <tiger2/> XML format. Figures 3 and 4 shows the resulting <tiger2/> XML file when the historyoftom text file is processed. Figure 3 shows the section for the named-entity recognition (NER) annotation set. By default, this section of the corpus contains the parts-of-speech (POS) and NER annotation sets from Stanford CoreNLP. Figure 4 shows the excerpt of the corpus that contains the sentences and their respective annotations.

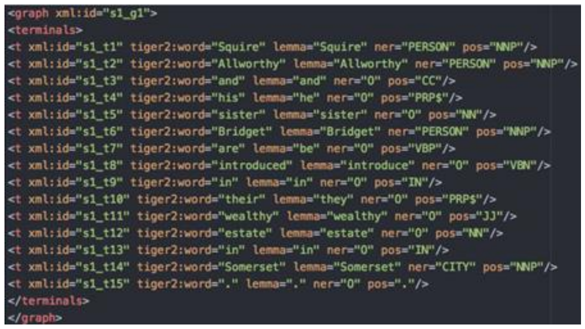

```xml
<graph xml:id="s1_g1">
<terminals>
<t xml:id="s1_t1" tiger2:word="Squire" lemma="Squire" ner="PERSON" pos="NNP"/>
<t xml:id="s1_t2" tiger2:word="Allworthy" lemma="Allworthy" ner="PERSON" pos="NNP"/>
<t xml:id="s1_t3" tiger2:word="and" lemma="and" ner="O" pos="CC"/>
<t xml:id="s1_t4" tiger2:word="his" lemma="he" ner="O" pos="PRP$"/>
<t xml:id="s1_t5" tiger2:word="sister" lemma="sister" ner="O" pos="NN"/>
<t xml:id="s1_t6" tiger2:word="Bridget" lemma="Bridget" ner="PERSON" pos="NNP"/>
<t xml:id="s1_t7" tiger2:word="are" lemma="be" ner="O" pos="VBP"/>
<t xml:id="s1_t8" tiger2:word="introduced" lemma="introduce" ner="O" pos="VBN"/>
<t xml:id="s1_t9" tiger2:word="in" lemma="in" ner="O" pos="IN"/>
<t xml:id="s1_t10" tiger2:word="their" lemma="they" ner="O" pos="PRP$"/>
<t xml:id="s1_t11" tiger2:word="wealthy" lemma="wealthy" ner="O" pos="JJ"/>
<t xml:id="s1_t12" tiger2:word="estate" lemma="estate" ner="O" pos="NN"/>
<t xml:id="s1_t13" tiger2:word="in" lemma="in" ner="O" pos="IN"/>
<t xml:id="s1_t14" tiger2:word="Somerset" lemma="Somerset" ner="CITY" pos="NNP"/>
<t xml:id="s1_t15" tiger2:word="." lemma="." ner="O" pos="."/>
</terminals>
</graph>
```

Fig. 4. Sentence from historyoftom.tiger2

When more than one file is uploaded, the File Merger merges them into one <tiger2/> XML file.

2.2 Annotations Module

The Annotations Module consists of the Annotations Manager and the Annotating Tool.

The Annotations Manager allows the user to manage custom annotation types and annotations that can be used in the Annotating Tool. It is an editor for updates or creation of custom annotations categories and values for the current corpus. If the user chooses to create a new annotation type, a feature name and values will be required by the module. The feature name will be used as the name of the domain of the values. Multiple values or annotations can be added per feature. Once the user successfully fills up the information required in creating a new annotation type, the module will update the current corpus file and will add the newly created annotation as a new feature. The newly created annotation can then be used in the Annotating Tool to annotate the tokens in the corpus.

The Annotating Tool serves as an editor for the annotated text within the corpus. It presents an organized list of sentences within the corpus along with their respective annotation tags that can be manually edited or updated by the user. The user can modify the annotations of a token. The user may edit the tags generated by the system (POS and NER) or they may use custom annotations created in the Annotations Manager.

2.3 Concept Concordancer API

We implemented our Concept Concordancer as a set of reusable functions to be used as an Applications Programming Interface (API). This is to allow flexibility in its usage, as well as to promote reusability in other applications, like as a component of an information extraction system.

The Concept Concordancer API accepts any serialized <tiger>XML corpus and outputs the corresponding result per function. The Concept Concordancer API consists of the Concept Finder, Concordancer Tool, Pattern Finder, Pattern Filter, and Concept Suggestion Finder.

The Concept Finder retrieves the related lexical terms of the input concept with the use of ConceptNet, WordNet, and a concept dictionary of previous searches (if any). It accepts an input concept string and POS value (noun or verb) from the user. It uses the input concept string by passing it to both ConceptNet and WordNet to retrieve the related lexical terms. For nouns, the relations retrieved from WordNet are its synonyms, hyponyms, and hypernyms, whereas the other word forms are retrieved from ConceptNet. For verbs, the relations retrieved from WordNet are synonyms and troponyms, whereas those retrieved from ConceptNet are the other word forms. It then uses the input concept string to retrieve all listed related words in the concept dictionary for that specific concept. If the word “show” as a verb is searched, the Concept Finder will return a list of related lexical terms. Sample results include “convey”, “usher”, and “testify” from synonyms in WordNet. “Peep” is a result from retrieving the troponym (manner) of “show”. “Shows”, “showing”, and “shown” are other word forms of show retrieved from ConceptNet.

The Concordancer Tool determines the concordances in a corpus based on a list of lexical terms. It accepts the set of lexical terms, a POS value string, and a corpus <tiger2/> XML file. For every lexical term on the list, the Concordancer Tool will go through the entire corpus to find instances of the term. For every instance the tool finds, it checks whether the POS value of the instance matches the POS value string (noun == noun, verb == verb). If the instance has the expected annotations, the sentence it belongs to will be added to the output concordance list, which is essentially a subcorpus that contains only the sentences that include instances of the lexical terms. The system will then pass this subcorpus file to the Pattern Finder to determine the candidate patterns.

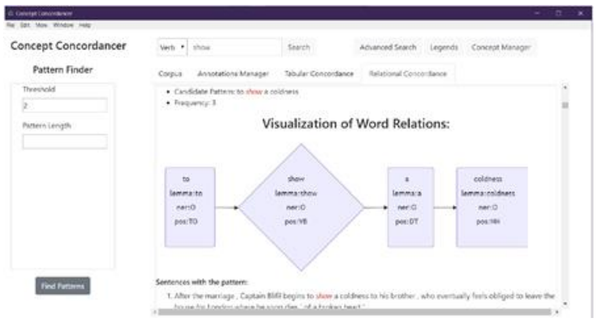
The Pattern Finder determines the candidate patterns based on the resulting concordances. It accepts the concordance list from the Concordancer Tool or from any data source if used in other applications. The candidate patterns are determined by getting word group patterns that occur more than once in the corpus. These patterns are the patterns of the words surrounding the lexical term. The number of times a pattern must occur in the corpus to be considered as a pattern can be adjusted, but the default value is two times. The candidate patterns are generated and are passed by the system to the Concept Suggestions Finder, Pattern Filter, and Visualization Module for display.
The Pattern Filter allows the user to filter the results by searching for a pattern from the concordances in the results file from the concept search or by using the patterns check filter. It accepts an input pattern string, and the concordances. The input pattern string is any combination of possible values of the chosen annotation type (e.g. DT/IN for POS). It goes through the whole file to find instances of the pattern string. It outputs a subset of the concordances list containing only the sentences with the pattern string. The result is forwarded to the Visualization Module to update the concordances shown.

The Concept Suggestion Finder generates a list of concept suggestions or terms related to the searched concept. It uses the patterns generated by the Pattern Finder and uses the “relatedto” relation from ConceptNet as a resource. For every pattern, it goes through the corpus to find words that are used in the same pattern.

Before it adds a word to the suggestions, it verifies that the word is not already part of the concept list. Then, for every word from the "relatedto" relation, it only adds the word to the list of suggestions if it is present in the corpus.

2.4 Visualization Module

This Visualization Module consists of Tabular Concordance and Relational Concordance. The Tabular Concordance submodule uses the concordance list from the Concordancer Tool in presenting the concordances that will be shown in the front-end. Figure 5 shows the resulting tabular view for the search word "show" as a verb.

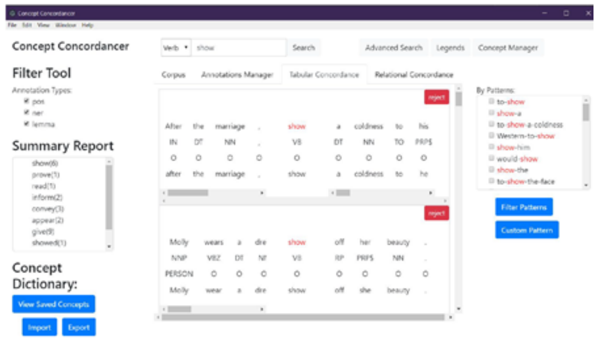


- Candidate Pattern: to show a coldness
- Frequency: 3

Sentences with the pattern:

1. After the marriage , Captain Blifil begins to show a coldness to his brother , who eventually feels obliged to leave the house for London where he soon dies , "of a broken heart."

Fig. 5. Relational Graph



show(6)
prove(1)
read(1)
inform(2)
convey(3)
appear(2)
give(9)
showed(1)

After	the	marriage	,	show	a	coldness	to	his
IN	DT	NN	,	VB	DT	NN	TO	PRP$
O	O	O	O	O	O	O	O	O
after	the	marriage	,	show	a	coldness	to	he

Molly	wears	a	dre	show	off	her	beauty	.
NNP	VBZ	DT	NI	VB	RP	PRP$	NN	.
PERSON	O	O	O	O	O	O	O	O
Molly	wear	a	dre	show	off	she	beauty	.

- to-show
- show-a
- to-show-a-coldness
- Western-to-show
- show-him
- would-show
- show-the
- to-show-the-face

Fig. 6. Tabular Concordance

The Relational Concordance submodule uses the concordance list from the Concordancer Tool and the candidate patterns from the Pattern Finder in visualizing the relations in the patterns. The generated visualization will be shown in the front-end. Figure 6 shows the resulting relational view for the search word "show" as a verb.

3. RESULTS AND OBSERVATIONS

For testing, we used a corpus of general written documents, such as excerpts from novels. We used two excerpt documents: The History of Tom Jones, a Foundling by Henry Fielding and The Hound of Baskervilles by Arthur Conan Doyle. The general documents were retrieved from the dhworkshop website (DHWorkshop).

The sample general corpus has 11413 words and 529 sentences. There is an average number of 21.57 words per sentence. It contains 2279 unique words, counting the noise words. It uses the third person point of view and usage is not domain specific.

We conducted 10 concept searches using the general corpus. Our criterion in choosing the concepts used in the testing is that these concepts should have five or more related words in the corpus. For each concept search, we created a testing reference containing the expected output.

There were three levels of testing performed. First, the initial results were recorded by comparing the actual results of the system to the expected results from the testing references. Second, relevant concept suggestions generated by the system were added to the concept list and the updated results were recorded. Third, the remaining expected concepts were added to the concept list as custom concepts and the updated results were recorded.

One of the issues found in testing with the general corpus was a limitation of the Concept Finder. There are related terms that it is not able to find with the ConceptNet and WordNet resources. This is because WordNet does not contain all possible related terms to every concept. Another contributing factor is that only the first degree related terms are used by the system to avoid more unrelated terms. This can be seen in the relatively lower scores on initial tests when concept suggestions and custom concepts have not yet been utilized to improve the performance. In total, only 31 concepts out of the expected 73 or 42.47% concepts were found by the Concept Finder in the initial results.

Another issue is Stanford CoreNLP generating incorrect annotations. There are instances where expected concordances are not found by the system because it does not have the correct annotation. For example, in the concept search for “book”, the concordance "Book IV concludes with a conversation between Sophia and Mrs Honour..." was missed because "Book" was incorrectly tagged as a verb. This occurred in 3 out of the 5 verb concept searches or 60% and in 2 out of the 5 verb concept searches or 40%. There are 8 concepts in the expected concordances that have incorrect annotations.

It was found that the system excels when the concepts in the expected concordances are in the base form because the system does not automatically consider all word forms of the concepts in the concept list. Different word forms have to be added through the add concept functionality of the system. An example of this is in the concept search for “house”. “Apartment” and “apartments” had to be added separately because they do not have the same form. All word forms are not automatically considered because not all forms represent the same concept. This can be depicted in the sample word “show” as a noun. Its word forms such as “showing” and “showed” do not have the same concept it has.

4. CONCLUSIONS AND FUTURE WORK

Our concept concordancer can process uploaded documents, manage corpus annotations, find concept concordances, generate candidate patterns, generate concept suggestions, and visualize results.

The following are potential areas of growth for the system:

● Provide multi-word support for annotations. Currently, each annotation refers to a single word. Allowing a phrase to be referred to by the annotation would be beneficial.

● Augment Concept Finder resources by cosidering ConceptNet relations such as "TypeOf" and "PartOf". Currently, only other word forms are retrieved from ConceptNet.

● Store retrieved ConceptNet and WordNet data in server to improve processing time. This would prevent delays from connecting always connecting to ConceptNet and WordNet servers every time a search is done.

● Add entries to a local ConceptNet instead of having a separate concept dictionary. Adding of new concepts to the same repository would eliminate need to maintain a separate concept dictionary.

5. ACKNOWLEDGMENTS

This is an offshoot from a project funded by the University Research Coordination Office.

6. REFERENCES

DHWorkshop. http://dhworkshop.pbworks.com/w/page/87288400/FrontPage. Accessed: 2018-05-29.

Gupta, S., & Manning, C. (2014). Spied: Stanford pattern-based information extraction and diagnostics. In Proceedings of the ACL 2014 workshop on interactive language learning, visualization, and interfaces

McEnery, T., & Hardieg, A. (2014). Corpus linguistics: Method, theory and practice. Cambridge University Press

Miller, G. A., Leacock, C., Tengi, R., & Bunker, R. T. (1993). A semantic concordance. In Proceedings of the workshop on human language technology, 303–308 . Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dx.doi.org/10.3115/1075671.1075742 doi: 10.3115/1075671.1075742

Seretan, V., & Wehrli, E. (2010). Tools for syntactic concordancing. In Proceedings of the international multiconference on computer science and information technology, 493-500. doi: 10.1109/IMCSIT.2010.5679742

Tribble, C., & Glyn, J. (1997). Concordances in the Classroom: a resource book for teachers. Athelstan Columns