Knowledge Based System for Composing Sentences to Summarize Documents

Andrey Timofeyev(✉) and Ben Choi

Computer Science, Louisiana Tech University, Ruston, USA
andtimo@latech.edu, pro@benchoi.org

Abstract. This chapter provides the details on how to build a knowledge-based system that is capable of composing new sentences to summarize multiple documents. The system is also capable of identifying the main topics of the given documents and is able to derive new concepts based on the given text data. In order to process the documents conceptually to create abstractive summaries, the system makes use of the Cyc development platform that consists of the world's largest knowledge base and one of the most powerful inference engines. The resultant knowledge based system first uses natural language processing techniques to extracts syntactic structure of the documents and then maps the words of the sentences into related concepts in the knowledge base. It then uses the inference engine to generalize and fuse concepts to form more abstract concepts. Since a word can be mapped into multiple concepts, the system also includes new techniques to handle word-sense disambiguation by using concept weights. After the generalization, the system is able to identify the main topics and the key concepts of the documents. The system then composes new sentences based on the key concepts by linking subject concepts with their related predicate concepts. The syntactic structure of the newly created sentences extends beyond simple subject-predicate-object triplets by incorporating adjective and adverb modifiers. The final stage is then to map the linked concepts back to words to form the abstractive sentences. The system has been implemented and tested. The implementation encodes a process that consists of seven stages: syntactic analysis, words mapping, concept propagation, concept weights and relations accumulation, topic derivation, subject identification, and new sentence generation. The implementation has been tested on various documents and webpages. The test results showed that the system is capable of creating new sentences that include abstracted concepts not explicitly mentioned in the original documents and that contain information synthesized from different parts of the documents to compose a summary.

Keywords: Text summarization · Knowledge-based system · Natural language processing · Data mining · Artificial intelligence

1 Introduction

In this chapter, we describe in details how to build a knowledge based automatic summarization system [1] that is capable of creating abstractive summaries of the given documents by generalizing new concepts, deriving main topics, and composing new

A. Fred et al. (Eds.): IC3K 2017, CCIS 976, pp. 164–183, 2019.
https://doi.org/10.1007/978-3-030-15640-4_9

sentences. The system processes text data on documents and webpages and utilizes knowledge base and inference engine to produce an abstractive summary. It generates summaries by composing new sentences based on the semantics derived from the text.

The system uses both the syntactic structure provided by the given documents and the commonsense knowledge provided by a knowledge base. It performs deep syntactic analysis by using capabilities of advanced natural language processing techniques. It uses Cyc development platform as a source of background knowledge. The Cyc development platform consists of the world's largest ontology of commonsense knowledge and a reasoning engine that allows information comprehension and abstraction [2]. In addition, Cyc ontology serves as a backbone for semantic analysis, knowledge generalization, and natural language generation functionality of the system. The system is domain independent and unsupervised, being limited only by the commonsense ontology provided by the Cyc development platform.

Research in automatic text summarization has been conducted mostly by using extractive methods that extract parts of the given text as the summary. However, abstractive summarization that creates an abstract as the summary is considered more desirable. The task to create an abstract based on reading a document is considered complex from human reader and is more so for machine. When human readers perform document summarization they tend to use their commonsense and domain expertise about subject matter to merge information from various parts of the document and synthesize novel information that was not explicitly mentioned in the text [3]. Moreover, it is not a simple task for human readers to compose new sentences for the summary.

We attempt to develop a system that is capable of composing new sentences for the summary. Our system generalizes new abstract concepts based on the knowledge derived from the text. It automatically detects main topics described in the text. Moreover, it composes new English sentences for some of the most significant concepts. The created sentences form an abstractive summary, combining concepts from different parts of the input text.

The system conducts summarization process in three principal stages: knowledge acquisition, knowledge discovery, and knowledge representation. The knowledge acquisition stage derives syntactic structure of each sentence of the input document and maps words and their relations into Cyc knowledge base. Next, the knowledge discovery stage generalizes concepts upward in the Cyc ontology and detects main topics covered in the text. Finally, the knowledge representation stage composes new sentences for some of the most significant concepts defined in main topics. The syntactic structure of the newly created sentences follows an enhanced subject-predicate-object model, where adjective and adverb modifiers are used to produce more complex and informative sentences.

We have implemented our proposed system using modular and pipelined design framework. Modularity provides the ability to conveniently maintain parts of the system and to add new functionality as needed. The pipelined design framework allows comprehensible data flow between different modules. The system was tested on various documents and webpages. The test results show that the system is capable of identifying key concepts and discovering main topics comprised in the original text, generalizing new concept not explicitly mentioned in the text, and creating new sentences that contain information synthesized from various parts of the text. The newly created sentences have complex syntactic structures that enhance subject-predicate-object triplets with adjective and adverb modifiers. For example, the sentence "Colored grapefruit being sweet edible fruit" was automatically generated by the system analyzing encyclopedia articles describing grapefruits. Here, the subject concept "grapefruit" is modified by the adjective concept "colored" that was not explicitly mentioned in the text and the object concept "edible fruit" is modified by the adjective concept "sweet".

The sentence was created as the result of linked key concepts. The linked concepts is then mapped back to words to form the sentence. As we can see from the above example, the created sentence sound like made by a machine than by a human. Human reader might use the word "is" instead of "being" in the example sentence. In short, although our system is able to generate new abstractive sentences, there is much more research potential to further develop such a knowledge based system to compose new sentences as summary.

The rest of the paper is organized as follows. Section 2 outlines related work undertaken for automatic text summarization. Section 3 describes the summarization process workflow. Section 4 covers the system implementation in details. Section 5 provides detailed description of the system modules. Section 6 presents testing results. Section 7 provides the computational performance of the system. Section 8 discusses conclusions and directions of future work.

2 Related Work

Automatic text summarization seeks to compose a concise and coherent version of the original text preserving the most important information. Computational community has studied automatic text summarization problem since late 1950 s [4]. Studies in this area are generally divided into two main approaches – extractive and abstractive. Extractive text summarization aims to select the most important sentences from original text to form a summary. Such methods vary by different intermediate representations of the candidate sentences and different sentence scoring schemes [5]. Summaries created by extractive approach are highly relevant to the original text, but do not convey any new information. Most prominent methods in extractive text summarization use term frequency versus inverse document frequency (TF-IDF) metric [6, 7] and lexical chains for sentence representation [8, 9]. Statistical methods based on Latent Semantic Analysis (LSA), Bayesian topic modelling, Hidden Markov Model (HMM) and Conditional random field (CRF) derive underlying topics and use them as features for sentence selection [10, 11]. Despite significant advancements in the extractive text summarization, such approaches are not capable of semantic understanding and limited to the shallow knowledge contained in the text.

In contrast, abstractive text summarization aims to incorporate the meaning of the words and phrases and generalize knowledge not explicitly mentioned in the original text to form a summary. Phrase selection and merging methods in abstractive summarization aim to solve the problem of combining information from multiple sentences. Such methods construct clusters of phrases and then merge only informative ones to form summary sentences [12]. Graph transformation approaches convert original text into a form of sematic graph representation and then combine or reduce such representation with an aim of creating an abstractive summary [13, 14]. Summaries constructed by described methods consist of sentences not used in the original text, combining information from different parts, but such sentences do not convey new knowledge.

Several approaches attempt to incorporate semantic knowledge base into automatic text summarization by using WordNet lexical database [8, 15, 16]. Major drawback of WordNet system is the lack of domain-specific and common sense knowledge. Unlike Cyc, WordNet does not have reasoning engine and natural language generation capabilities.

Recent rapid development of deep learning contributes to the automatic text summarization, improving state-of-the-art performance. Deep learning methods applied to both extractive [17] and abstractive [18] summarization show promising results, but such approaches require vast amount of training data and powerful computational resources.

Our system extends to the one proposed in [19]. In the work, the structure of created sentences has simple subject-predicate-object pattern and new sentences are only created for clusters of compatible sentences found in the original text.

3 Knowledge-Based Abstractive Summarization Process

Our abstractive knowledge-based summarization system attempts to bring the machines one-step closer to the comprehension of the knowledge comprised in the text. The system performs text summarization in three principal stages: the knowledge acquisition, the knowledge discovery, and the knowledge representation. The overview of the summarization process is illustrated in Fig. 1.

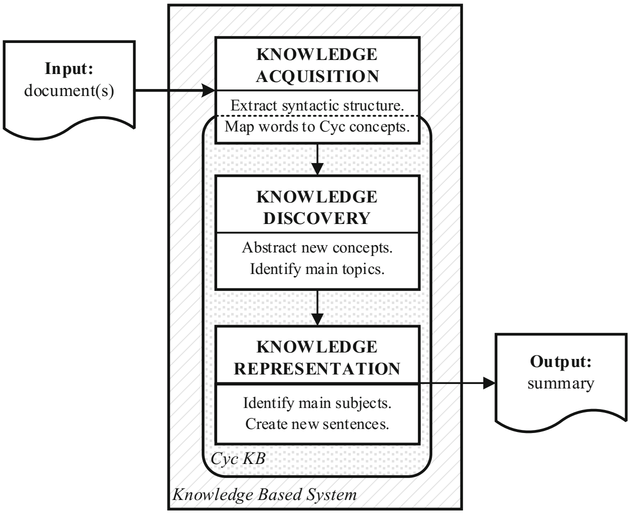

Fig. 1. System's workflow diagram [1].

During the knowledge acquisition stage, the algorithm receives text documents as an input, performs deep syntactic analysis, and maps the words with their syntactic relationships into the Cyc knowledge base. During the knowledge discovery stage, the system performs a generalization of new concepts by propagating the concepts that were mapped into Cyc knowledge base by the knowledge acquisition step. It also performs the task of the identification of the main topics of the text based on the mapped and generalized concepts. Finally, during the knowledge representation stage, the system generates new sentences using knowledge derived from the input text documents and the capabilities of the Cyc inference engine.

3.1 Knowledge Acquisition

The knowledge acquisition stage consists of two sub-processes. The first sub-process extracts the syntactic structures from the given documents. This sub-process serves as a data preprocessing and transformation step. It normalizes raw text data and transforms it into syntactic representation. The workflow diagram of the sub-process is outlined in Fig. 2.

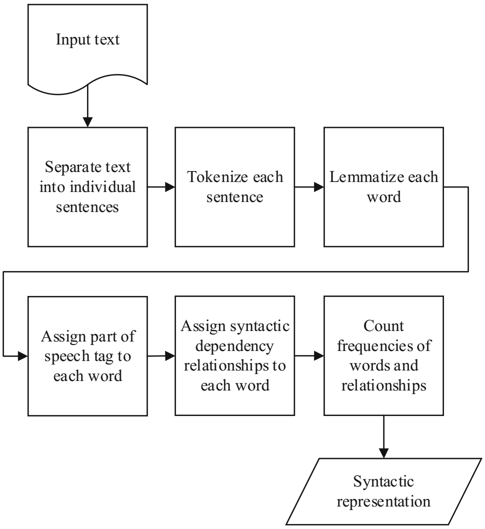

Fig. 2. Syntactic structure extraction sub-process workflow diagram.

The second sub-process maps words from syntactic representation of the text to Cyc concepts. Mapped Cyc concepts are utilized for reasoning during subsequent steps of the algorithm. The workflow diagram of the sub-process is illustrated in Fig. 3.

3.2 Knowledge Discovery

The knowledge discovery stage performs two sub-processes: it abstracts new concepts and identifies main topics described in the input text.

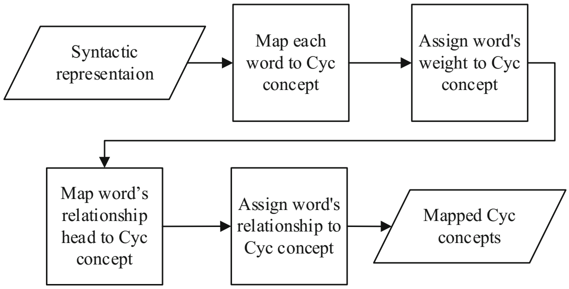

Fig. 3. Mapping words to Cyc concepts sub-process workflow diagram.

New concepts abstraction sub-process generalizes the information derived from the text. It finds the ancestors of mapped Cyc concepts and assigns the descendants' propagated weight and syntactic dependency relationships to the ancestors. It is an important part of the abstractive summarization process as it allows deriving concepts that are not explicitly mentioned in the input text. For example, concepts like "cat," "tiger," "jaguar," and "lion" are generalized into more abstract "feline" concept. The workflow diagram of the new concepts abstraction sub-process is illustrated in Fig. 4.

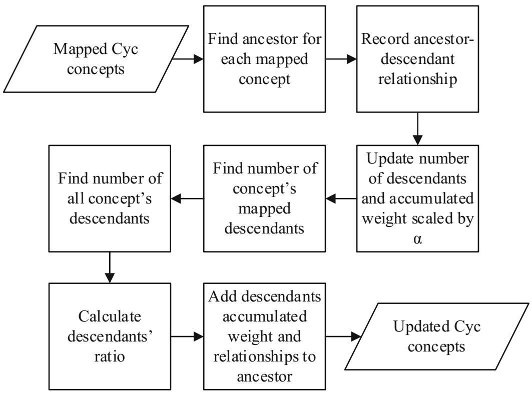

Fig. 4. New concepts abstraction sub-process workflow diagram.

The main topics identification sub-process detects topics described in the text with an assumption that they are represented by the most frequently used micro theories. Micro theories form the basis of the knowledge organization in Cyc ontology being the clusters of Cyc concepts and facts, typically representing one specific domain of knowledge. For example, #$BiologyMt is a micro theory containing biological knowledge, and #$MathMt is a micro theory containing concepts and facts describing the field of mathematics. Each Cyc concept is defined within a micro theory. The workflow diagram of the main topics identification sub-process is illustrated in Fig. 5.

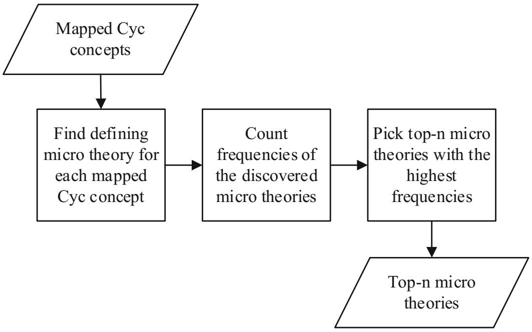

Fig. 5. Main topics identification sub-process workflow diagram.

3.3 Knowledge Representation

The knowledge representation stage utilizes powerful capabilities of the Cyc inference engine to generate new sentences based on the information discovered during knowledge acquisition and knowledge discovery steps. This stage uses mapped and generalized Cyc concepts, their syntactic dependency relationships, and the most frequent micro theories as inputs. The knowledge representation stage consists of two sub-processes – candidate subjects' discovery and new sentences generation.

The candidate subjects' discovery sub-process identifies significant subject concepts out of all the mapped and generalized Cyc concepts. The workflow diagram of the sub-process is outlined in Fig. 6.

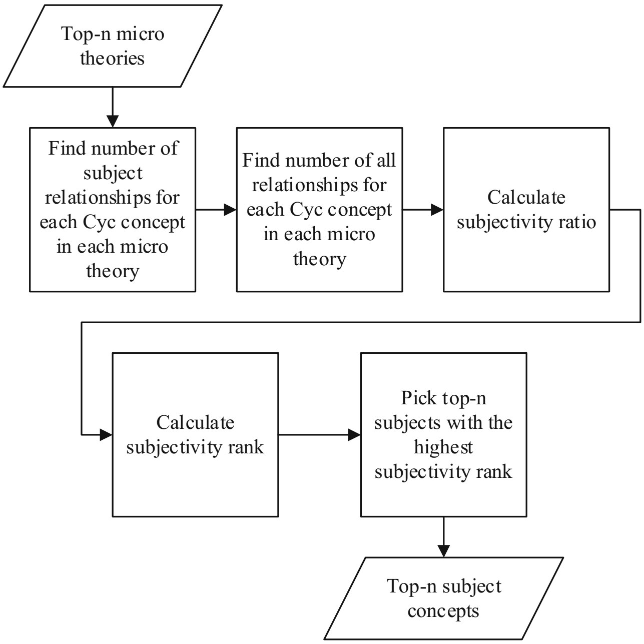

Fig. 6. Candidate subjects discovery sub-process workflow diagram.

The new sentences generation sub-process composes new sentences for each of the identified candidate subject concepts. The generated sentences serve as a final summary of the input text. The workflow diagram of the sub-process is outlined in Fig. 7.

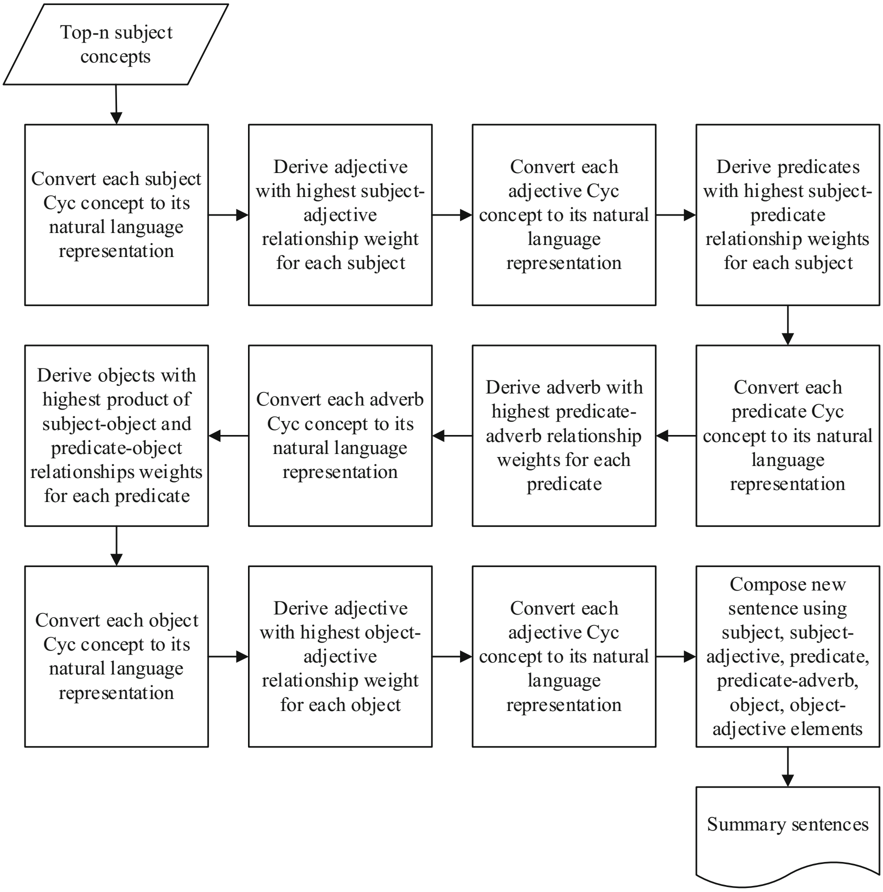

Fig. 7. New sentences generation sub-process workflow diagram.

4 Details of the System's Implementation

We chose Python as the implementation language to develop our system because of the advanced Natural Language Processing tools and libraries it supplies. Our system uses Cyc knowledge base and inference engine as a backbone for the semantic analysis. Cyc development platform supports communications with the knowledge base and utilization of the inference engine through the application programming interfaces (APIs) implemented in Java. We utilize Java-Python wrapper supported by JPype library to allow our system using Cyc Java API packages. JPype library is essentially an interface at a basic level of virtual machines [20]. It requires starting Java Virtual Machine with a path to the appropriate jar files before Java methods and classes can be accessible within Python code. Communication between our system and Cyc development platform is illustrated in Fig. 8. To the best of our knowledge, our developed system is the first Python-based system that allows communication with Cyc development platform.

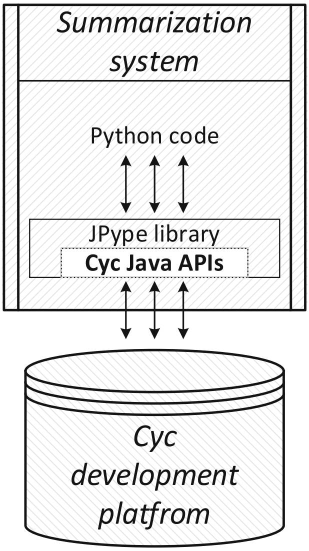

Fig. 8. Communication between summarization system and Cyc development platform.

We have designed our system as a modular and pipelined summarization framework. Modularity provides the ability to conveniently maintain parts of the system and to add new functionality as needed. Pipelined design allows comprehensible data flow between different modules.

The system consists of seven modules:

1. Syntactic analysis;
2. Mapping words to Cyc KB;
3. Concepts propagation;
4. Concepts' weights and relations accumulation;
5. Topics derivation;
6. Subjects identification;
7. New sentences generation.

Modules 1 and 2 together constitute the knowledge acquisition stage of the summarization process. Modules 3, 4 and 5 together make up the knowledge discovery stage of the summarization process. Modules 6 and 7 together form knowledge representation stage of the summarization process. System modules are illustrated in Fig. 9.

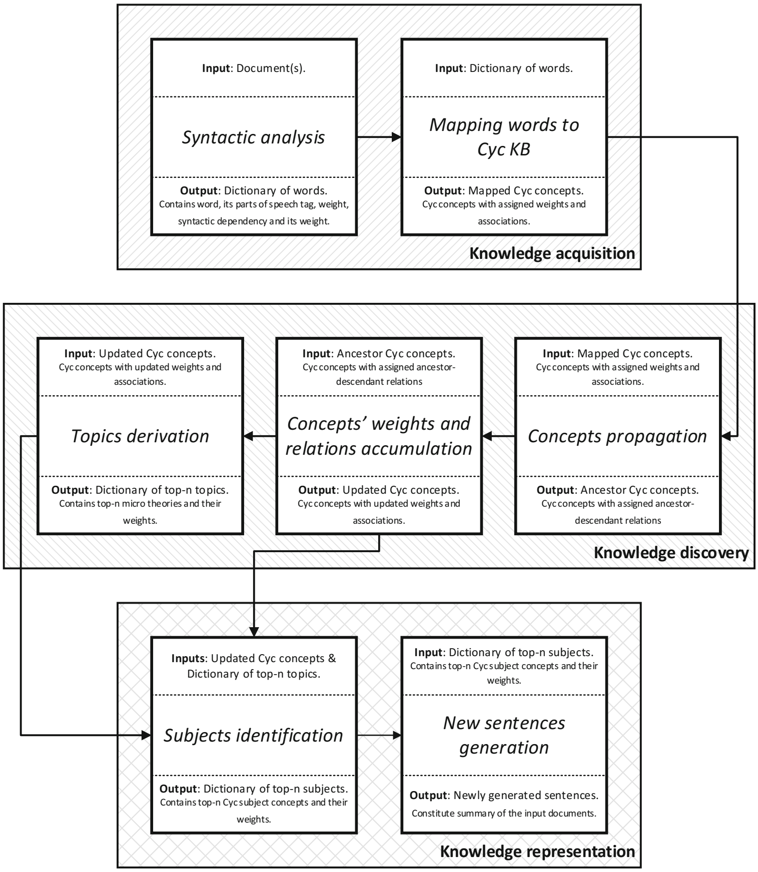

Fig. 9. Modular design of the system.

5 Description of the System's Modules

5.1 "Syntactic Analysis" Module

The first module in the system is the "Syntactic analysis" module. The role of this module is essentially a data preprocessing. The module takes documents as an input and transforms them into syntactic representations. It first performs text normalization by lemmatizing each word in each sentence. Then it derives part of speech tags, parses syntactic dependencies and counts word's weights. The syntactic dependencies are recorded in the following format: ("word" "type" "head"), where "word" is the dependent element, "type" is the type of the dependency, and "head" is the leading element. For example, applying syntactic parser on the following sentence: "John usually drinks strong coffee" produces the following syntactic dependencies between words: ("Steve" "nsubj" "drinks"), ("tea" "dobj" "drinks"), ("rarely" "advmod" "drinks"), ("black" "amod" "tea"). Syntactic dependencies of the example sentence are illustrated in Fig. 10.

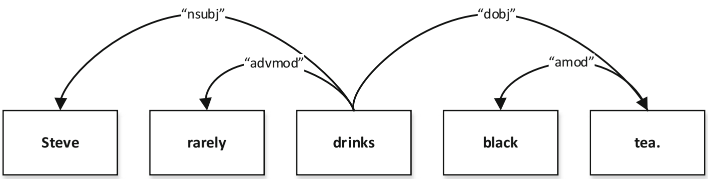

Fig. 10. Illustration of the syntactic dependencies of a sample sentence.

The "Syntactic analysis" module is implemented using SpaCy – Python library for advanced natural language processing. SpaCy library is the fastest in the world with the accuracy within one percent of the current state of the art systems for part of speech tagging and syntactic dependencies analysis [21]. The "Syntactic analysis" module operates outside of the Cyc development platform. The output of the module is a dictionary that contains words, their part of speech tags, weights and syntactic dependencies. This dictionary serves as an input for "Mapping words to Cyc KB" module.

5.2 "Mapping Words to Cyc KB" Module

The "Mapping words to Cyc KB" module takes dictionary of words, derived by the "Syntactic analysis" module, as an input. This module finds an appropriate Cyc concept for each word in the dictionary, and assigns word's weight and syntactic dependency associations to Cyc concept. It starts by mapping each word to the corresponding Cyc concept (1). Next, it assigns word's weight to Cyc concept (2). Then it maps the syntactic dependency head to the appropriate Cyc concept. Finally, it assigns the syntactic dependency association and its weight to the Cyc concept (3). Table 1 provides the description of Cyc commands used to implement each step.

Table 1. Description of Cyc commands used by "Mapping words to Cyc KB" module.

Step	Cyc command	Description
1	(#$and (#$denotation ?Word ?POS ?Num ?Concept) (#$wordForms ?Word ?WordForm "word") (#$genls ?POS ?POSTag))	Command uses built-in "#$denotation" Cyc predicate to relate a "word", its part of speech tag (?POS), and a sense number (?Num) to concept (?Concept). It also uses "#$wordForms" and "#$genls" predicates to accommodate for all variations of word's lexical forms
2	(#$conceptWeight ?Concept ?Weight)	Command uses user-defined "#$conceptWeight" Cyc predicate that assigns the weight (?Weight) to the concept (?Concept)
3	(#$conceptAssociation ?Concept ?Type ?HeadConcept ?Weight)	Command uses user-defined "#$conceptAssociation" Cyc predicate that assigns a specific type (?Type) of a syntactic dependency association, the leading element (?HeadConcept) and the weight (?Weight) to the concept (?Concept)

This module communicates with Cyc development platform and updates weight and syntactic dependency relations of Cyc concepts. The output of the module are mapped Cyc concepts with assigned weights and syntactic dependency relations. The mapped Cyc concepts serve as an input for "Concepts propagation" module. "Syntactic analysis" and "Mapping words to Cyc KB" modules together constitute the knowledge acquisition step of the summarization process.

5.3 "Concepts Propagation" Module

The "Concepts propagation" module takes Cyc concepts, mapped by "Mapping words to Cyc KB" module, as an input and finds their closest ancestor concepts. This module performs generalization and abstraction of new concepts that have not been mentioned in the text explicitly. It starts by querying Cyc knowledge base for all the concepts that have assigned weight (1). Then it finds an ancestor concept for each concept derived by the query (2). Next, it records the number of ancestor's descendant concepts and their weight (3). Finally, it assigns ancestor-descendant relation between ancestor and descendant concepts (4). Table 2 provides the description of Cyc commands used to implement each step.

Table 2. Description of Cyc commands used by "Concepts propagation" module.

Step	Cyc command	Description
1	(#$conceptWeight ?Concept ?Weight)	Command uses user-defined "#$conceptWeight" Cyc predicate to retrieve concepts (?Concept) that have assigned weights (?Weight)
2	(#$min-genls ?Concept)	Command uses built-in "min-genls" Cyc predicate to retrieve the closest ancestor concept for the given concept (?Concept)
3	(#$conceptDescendants ?Concept ?Weight ?Count)	Command uses user-defined "#$conceptDescendants" Cyc predicate to record the number of descendants (?Count) and their weight (?Weight) to the ancestor concept (?Concept)
4	(#$conceptAncestorOf ?Concept ?Descendant)	Command uses user-defined "#$conceptAncestorOf" predicate to assign ancestor-descendant relation between the ancestor concept (?Concept) and the descendant concept (?Descendant)

This module communicates with Cyc development platform to derive all mapped Cyc concepts, find closest ancestor concepts and update ancestor concepts' relations. The output of the module are ancestor Cyc concepts with assigned descendant concepts' weights and counts and ancestor-descendant relations. The ancestor Cyc concepts are used by "Concepts' weights and relations accumulation" module.

5.4 "Concepts' Weights and Relations Accumulation" Module

The "Concepts' weights and relations accumulation" module takes ancestor Cyc concepts as an input and adds descendants' accumulated weight and relations to ancestor concepts if the calculated descendant-ratio is higher than the threshold. The descendant-ratio is the number of mapped descendant concepts divided by the number of all descendant concepts of an ancestor concept. This module starts by querying Cyc knowledge base for all ancestor concepts (1). Then it calculates the descendant ratio for each ancestor concept (2.1, 2.2). Next, it adds propagated descendants' weight (3) and descendants' associations with their propagated weights (4) to ancestor concepts if the descendant-ratio is higher than the defined threshold. Table 3 provides the description of Cyc commands used to implement each step.

Table 3. Description of Cyc commands used by "Concepts' weights and relations accumulation" module.

Step	Cyc command	Description
1	(#$conceptDescendants ?Concept ?Weight ?Count)	Command uses user-defined "#$conceptDescendants" Cyc predicate to retrieve all concepts (?Concept) that have descendants
2.1	(#$conceptAncestorOf ?AncConcept ?MappedDesc)	Command uses user-defined "#$conceptAncestorOf" predicate to retrieve mapped descendant concepts (?MappedDesc) of the given ancestor concept (?AncConcept)
2.2	(#$genls ?AncConcept ?DescConcept)	Command uses built-in "#$genls" Cyc predicate to retrieve all descendant concepts (?DescConcept) of the given ancestor concept (?AncConcept)
3	(#$conceptWeight ?AncConcept ?DescWeight)	Command uses user-defined "#$conceptWeight" Cyc predicate to assigns the descendant concepts' propagated weight (?DescWeight) to the ancestor concept
4	(and (#$conceptAncestorOf ?AncConcept ?DescConcept) (#$conceptAssociation ?DescConcept ?Type ?HeadConcept ?Weight))	Command uses user-defined "#$conceptAncestorOf" and "#$conceptAssociation" Cyc predicates to assign descendant's association (?DescConcept) and its propagated weight (?Weight) to the ancestor concept (?AncConcept)

This module communicates with Cyc development platform to derive all ancestor Cyc concepts, find the number of ancestor's mapped descendants, find the number of all ancestor's descendants and update ancestor's weight and relations. The output of the module are the Cyc concepts with updated weights and syntactic dependency associations. Updated Cyc concepts are used by the "Topics derivation" and the "Subjects identification" modules.

5.5 "Topics Derivation" Module

The "Topics derivation" module takes updated Cyc concepts as an input and derives defining micro theory for each concept. Micro theories with the highest weights represent the main topics of the document. This module first derives defining micro theory for each Cyc concept that have assigned weight (1). Then it counts the weights of derived micro theories based on their frequencies and picks up top-n with the highest weights. Table 4 provides the description of Cyc command used to implement defining micro theory derivation.

Table 4. Description of Cyc command used by "Topics derivation" module.

Step	Cyc command	Description
1	(#$and (#$conceptWeight ?Concept ?Weight) (#$definingMt ?Concept ?MicroTheory))	Command uses user-defined "#$conceptWeight" Cyc predicate and built-in "definingMt" Cyc predicate to derive defining micro theory (?MicroTheory) for each concept (?Concept) that have assigned weight (?Weight)

This module communicates with Cyc development platform to derive defining micro theory for each mapped Cyc concept. Calculation of the derived micro theories' weights is handled outside of the Cyc development platform. The output of the module is the micro theories dictionary that contains top-n micro theories with highest weights. This dictionary serves as an input for the "Subjects identification" module. The "Concepts propagation", the "Concepts' weights and relations accumulation" and the "Topics derivation" modules together constitute knowledge discovery step of the summarization process.

5.6 "Subjects Identification" Module

The "Subjects identification" module uses updated Cyc concepts and the dictionary of top-n micro theories as an input to derive most informative subject concepts based on a subjectivity rank. Subjectivity ranks is the product of the concept's weight and the concept's subjectivity ratio. Subjectivity ratio is the number of concept's syntactic dependency associations labelled as "subject" relations divided by the total number of concept's syntactic dependency associations. Subjectivity rank allows identifying concepts with the strongest subject roles in the documents. The module start by querying Cyc knowledge base for all mapped Cyc concepts for each micro theory in top-n micro theories dictionary (1). Then it calculates subjectivity ratio and subjectivity rank for each derived Cyc concept (2.1, 2.2). Finally, it picks top-n subject concepts with the highest subjectivity rank. Table 5 provides the description of Cyc commands used to implement each step.

Table 5. Description of Cyc commands used by "Subjects identification" module.

Step	Cyc command	Description
1	(#$and (#$definingMt ?Concept ?MicroTheory) (#$conceptWeight ?Concept ?Weight))	Command uses built-in "#$definingMt" Cyc predicate and user-defined "conceptWeight" Cyc predicate to derive concepts (?Concept) that have assigned weight (?Weight) for each micro theory (?MicroTheory) in micro theories dictionary
2.1	(#$conceptAssociation ?Concept "nsubj" ?HeadConcept ?Weight)	Command uses user-defined "#$conceptAssociation" Cyc predicate with "nsubj" parameter to derive the concept's (?Concept) syntactic dependency associations labelled as "subject" relations
2.2	(#$conceptAssociation ?Concept ?Type ?HeadConcept ?Weight)	Command uses user-defined "#$conceptAssociation" Cyc predicate with no parameter specified (?Type) to derive all concept's (?Concept) syntactic dependency associations

This module communicates with Cyc development platform to derive mapped Cyc concepts for each defining micro theory in the input dictionary and to find the number of the concept's syntactic dependency associations labelled as "subject" relation and the number of all syntactic dependency associations of the concept. Calculations of the subjectivity ratio and the subjectivity rank are handled outside of the Cyc development platform. The output of the module is the dictionary that contains top-n subjects with the highest subjectivity rank. This dictionary serves as an input for the "New sentence generation" module.

5.7 "New Sentences Generation" Module

The "New sentences generation" module takes the dictionary of top-n most informative subjects as an input and produces new sentences for each of the subject to form a summary of the input documents. The module starts by deriving a natural language representation of each subject Cyc concept in the dictionary (1). Then it picks the adjective Cyc concept modifier with the highest subject-adjective syntactic dependency association weight (2) and derives its natural language representation. Next, it picks top-n predicate Cyc concepts with the highest subject-predicate syntactic dependency association weights (3) and derives their natural language representations. Then it picks the adverb Cyc concept modifier with the highest predicate-adverb syntactic dependency association weight (4) and derives its natural language representation. Next, it picks top-n object Cyc concepts with the highest product of subject-object and predicate-object syntactic dependency association weights (5.1, 5.2) and derives their natural language representations. Then, it picks the adjective Cyc concept modifier with the highest object-adjective syntactic dependency association weight and derives its natural language representation. Finally, it composes the new sentence using subject,

subject-adjective, predicate, predicate-adverb, object and object-adjective natural language representations. Table 6 provides the description of Cyc commands used to implement each step.

Table 6. Description of Cyc commands used by "New sentence generation" module.

Step	Cyc command	Description
1	(#$generate-phrase ?Concept)	Command uses built-in "#$generate-phrase" Cyc predicate to retrieve corresponding natural language representation for a Cyc concept (?Concept)
2	(#$conceptAssociation ?Concept "amod" ?HeadConcept ?Weight)	Command uses user-defined "# $conceptAssociation" Cyc predicate with "amod" parameter to derive Cyc concept (?Concept) associations labelled as adjective modifier syntactic dependency relation
3	(#$conceptAssociation ?Concept "pred" ?HeadConcept ?Weight)	Command uses user-defined "# $conceptAssociation" Cyc predicate with "pred" parameter to derive Cyc concept (?Concept) associations labelled as predicate syntactic dependency relation
4	(#$conceptAssociation ?Concept "advmod" ?HeadConcept ?Weight)	Command uses user-defined "# $conceptAssociation" Cyc predicate with "advmod" parameter to derive Cyc concept (?Concept) associations labelled as adverb modifier syntactic dependency relation
5.1	(#$conceptAssociation ?Concept "obj" ?HeadConcept ?Weight)	Command uses user-defined "# $conceptAssociation" Cyc predicate with "obj" parameter to derive Cyc concept (?Concept) associations labelled as object syntactic dependency relation
5.2	(#$conceptAssociation ?Concept "subj-obj" ?HeadConcept ?Weight)	Command uses user-defined "# $conceptAssociation" Cyc predicate with "subj-obj" parameter to derive Cyc concept (?Concept) associations labelled as subject-object syntactic dependency relation

This module communicates with Cyc development platform to derive appropriate Cyc concepts for each sentence element based on the weights of their syntactic dependency associations and derive their natural language representation. New sentences are composed outside of the Cyc development platform and serve as an output for the module and the whole summarization system. The "Subjects identification" and the "New sentences generation" modules together constitute the knowledge representation step of the summarization process.

6 Testing and Results

We have tested our system on various encyclopedia articles describing concepts from different domain. First, we conducted an experiment using multiple articles about grapefruits. In this experiment, we increased the number of analyzed articles on each run of the system, starting with a single article. Figure 11 illustrates new sentences created by the system. These results show the progression of sentence structure from simple subject-predicate-object triplet to more complex structure enhanced by the adjective and adverb modifiers when more articles were processed by the system.

"Grapefruit being fruit." (a)
"Grapefruit being colored edible fruit." (b)
"Colored grapefruit being sweet edible fruit." (c)

Fig. 11. Test results of new sentences created for multiple articles about grapefruit; (a) – single article, (b) – two articles, (c) – three articles [1].

Next, we applied our system on five encyclopedia articles describing different types of felines, including cats, tigers, cougars, jaguars and lions. Figure 12 shows main topics and concepts extracted from the text and newly created sentences.

Topics (micro theories): • *#$BiologyMt* • *#$BiologyVocabularyMt* • *#$HumanSocialLifeMt* Concepts: • *#$Cat* • *#$DomesticCat* • *#$FelisGenus* • *#$FelidaeFamily* • *#$Animal*	Sentences: *"Cat usually being native animal."* *"Big felis usually being natural predatory animal."* *"Big felis usually being exotic animal."* *"Big felis often using killing method."* *"Big felis often using marking."* *"Male feline often killing prey."* *"Male feline living historical mountain range."*

Fig. 12. Test results of new sentences, concepts and main topics for encyclopedia articles about felines [1].

These results show that the system is able to abstract new concepts and create new sentences that contain information synthesized from different parts of the documents. Concepts like "canis", "mammal meat" and "felis" were derived by the generalization process and were not explicitly mentioned in the original documents. Our system yields better results compared to the reported in [19]. New sentences created by the system have structure that is more complex and contain information fused from various parts of the text. More testing results are reported in [1].

7 System's Performance

The computational complexity of our proposed system is upper bounded by the polynomial expression in the size of the vocabulary of the input documents and therefore, the system is considered to be of the polynomial time complexity. Vocabulary of the document is the number of the unique lemmas contained in the document. Table 7 illustrates the performance of the system when applied to the encyclopedia articles. The experiments were conducted on a machine with a 2.0 GHz Intel Xeon E5-2620 CPU and 32 GB of RAM.

Table 7. System performance scores using encyclopedia articles.

# of articles	Article name(s)	Source(s)	Vocabulary size (Lemmas)	CPU time (Seconds)
3	"Grapefruit"	Wikipedia, Morton, New world Encyclopedia	1988	2608
5	"Cat" "Tiger" "Cougar" "Jaguar" "Lion"	Wikipedia	5812	6974

8 Conclusions and Future Work

The task of producing an abstractive summary of a given text is considered challenging for humans and even more so for machines. Employing the semantic features and the syntactic structure of the text together with the world's largest knowledge base shows great potential in creating abstractive summaries. In this chapter, we thoroughly described the design and implementation of a knowledge-based abstractive summarization system that can automatically composes new sentences as the summary. Our knowledge-based system employs the Cyc knowledge base and its reasoning engine as a backbone for commonsense and inferencing ability. The system creates a summary of a given text by composing new sentences that contain the information aggregated from the various parts of the text. The structure of the summary sentences is enhanced from simple subject-predicate-object triplets to a more complex structure by adding adjective and adverb modifiers. Although our system is able to generate new abstractive sentences, there is much more research potential to further develop such a knowledge based system to compose new sentences as summary.

Future research potential includes enhancing the domain knowledge since the semantic knowledge and reasoning are limited to the functionality and performance of the underlining commonsense knowledge base. Our system is currently as knowledgeable as the capabilities of the Cyc knowledge base that is currently the largest ontology of commonsense knowledge. For future improvement, a system could use the information derived from the whole World Wide Web as a domain knowledge. This would possess challenging research questions such as information inconsistency and sense disambiguation. In addition, a robust inference engine would be required to process the information correctly and in a timely fashion. Another potential research is to improve the process and the structure of composing new sentences. Our system currently uses subject-predicate-object triplets enhanced by adjective and adverb modifiers. It does not yet resemble the structure of the sentences created by human. The structure of newly created sentences can be improved by using a more sophisticated representation of the syntactic structure of the sentence, such as graph representation. Moreover, another future research direction is to compose several connected sentences to form a coherent abstract. Currently, the sentences created by our system are not directly connected to each other. One possible enhancement is by representing the whole document as a graph of connected concepts with various relationships among them and then creating new sentences based on these relationships. Much more research is needed for a machine to create a coherent abstract to summarize documents.

References

1. Timofeyev, A., Choi, B.: Knowledge based automatic summarization. In: Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3 K 2017). pp. 350–356. SCITEPRESS (2017). https://doi.org/10.5220/0006580303500356
2. Cycorp – Cycorp Making Solutions Better. http://www.cyc.com
3. Cheung, J., Penn, G.: Towards robust abstractive multi-document summarization: a caseframe analysis of centrality and domain. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp. 1233–1242. Association for Computational Linguistics (2013)
4. Luhn, H.: The automatic creation of literature abstracts. IBM J. Res. Dev. **2**, 159–165 (1958). https://doi.org/10.1147/Rd.22.0159
5. Nenkova, A., Mckeown, K.: A survey of text summarization techniques. In: Charu, A., Zhai, C. (eds.) Mining Text Data, pp. 43–76. Springer, Heidelberg (2012)
6. Hovy, E., Chin-Yew, L.: Automated text summarization and the SUMMARIST system. In: Proceedings of a Workshop Held at Baltimore, Maryland, 13–15 October 1998, pp. 197–214. Association for Computational Linguistics (1998). https://doi.org/10.3115/1119089.1119121
7. Radev, D., Jing, H., Styś, M., Tam, D.: Centroid-based summarization of multiple documents. Inf. Process. Manag. **40**, 919–938 (2004). https://doi.org/10.3115/1117575.1117578
8. Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. Adv. Autom. Text Summ., 111–121 (1999). https://doi.org/10.7916/d85b09vz
9. Ye, S., Chua, T., Kan, M., Qiu, L.: Document concept lattice for text understanding and summarization. Inf. Process. & Manag. **43**, 1643–1662 (2007). https://doi.org/10.1016/J.Ipm.2007.03.010
10. Gong, Y., Liu, X.: Generic text summarization using relevance measure and latent semantic analysis. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 19–25. ACM (2001). https://doi.org/10.1145/383952.383955

11. Shen, D., Sun, J., Li, H., Yang, Q., Chen, Z.: Document summarization using conditional random fields. In: Proceedings of International Joint Conference on Artificial Intelligence, pp. 2862–2867. IJCAI (2007)
12. Bing, L., Li, P., Liao, Y., Lam, W., Guo, W., Passonneau, R.: Abstractive multi-document summarization via phrase selection and merging. In: Proceedings of the ACL-IJCNLP, pp. 1587–1597. Association for Computational Linguistics (2015)
13. Ganesan, K., Zhai, C., Han, J.: Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In: Proceedings of the 23rd International Conference on Computational Linguistics, pp. 340–348. Association for Computational Linguistics (2010)
14. Moawad, I., Aref, M.: Semantic graph reduction approach for abstractive text summarization. In: 2012 Seventh International Conference Computer Engineering & Systems (ICCES), pp. 132–138. IEEE (2012). https://doi.org/10.1109/icces.2012.6408498
15. Bellare, K., Sharma, A.D., Loiwal, N., Mehta, V., Ramakrishnan, G., Bhattacharyya, P.: Generic text summarization using WordNet. In: Language Resources and Evaluation Conference, pp. 691–694. LREC (2004)
16. Pal, A., Saha, D.: An approach to automatic text summarization using WordNet. In: 2014 IEEE International Advance Computing Conference (IACC), pp. 1169–1173. IEEE (2014). https://doi.org/10.1109/iadcc.2014.6779492
17. Nallapati, R., Zhai, F., Zhou, B.: SummaRuNNer: a recurrent neural network based sequence model for extractive summarization of documents. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017). pp. 3075–3081. AAAI (2017)
18. Rush, A.M., Chopra, S., Wetson, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 379–389. EMNLP (2015). https://doi.org/10.18653/v1/d15-1044
19. Choi, B., Huang, X.: Creating new sentences to summarize documents. In: The 10th IASTED International Conference on Artificial Intelligence and Application (AIA 2010), pp. 458–463. IASTED (2010)
20. Jpype - Java to Python Integration. http://jpype.sourceforge.net
21. Honnibal, M., Johnson, M.: An improved non-monotonic transition system for dependency parsing. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1373–1378. EMNLP (2015). https://doi.org/10.18653/v1/d15-1162