Analyzing the Digital Brains of Virtual Assistants

Ann Clifton* and Ben Choi

***Abstract*—In this paper, we analyze the digital brains of virtual assistants by reviewing their inner workings of learning and thinking. We review the process for creating the digital brains (knowledgebase and inference engine) by focusing on knowledge acquisition (learning) and knowledge discovery (thinking). We then attempt to extend the ability of virtual assistants by allowing them to read and write. To read documents conceptually in order to write abstractive summaries, our system makes use of one of the world's largest knowledgebases and one of the most powerful inference engines. The resultant AI system first uses natural language processing techniques to extract syntactic structure of the documents and then maps the words of the sentences and their parts of speech into related concepts in the knowledgebase. It then uses the inference engine to generalize and fuse concepts to form more abstract concepts. The system then composes new sentences based on the key concepts by linking subject concepts with their related predicate concepts. The system has been implemented and tested. The test results showed that the system can create 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.**

***Index Terms*—knowledgebase, natural language processing, knowledge engineering.**

I. Introduction

Virtual or digital assistants (intelligent personal assistants) have become commonplace in our homes, on our wrists, and in our pockets. Examples include Amazon's Alexa, Apple's Siri, Google Assistant, and Microsoft's Cortana. The ability to have nearly any query answered by simply speaking to a device represents a powerful advancement in technology. Virtual assistants do have their limitations; for instance, they have trouble replying to questions with high levels of ambiguity and nuance. In this paper, we begin by reviewing the process for creating the digital brains (knowledgebase and inference engine) of virtual assistants by focusing on knowledge acquisition (learning) and knowledge discovery (thinking).

We then attempt to extend the ability of virtual assistants by allowing them to read and write. The ability to read, understand, and write human languages requires not only processing the given text data, but also requires commonsense knowledge. The additional knowledge required are encoded into computer programs and databases. Programs for parsing, encoded with the knowledge of grammar, read and analyze the given text data syntactically to produce parts of speech. Knowledgebases, encoded with the ontology of human knowledge, provide concepts that relate words semantically to their meanings. Inference engines, encoded with the knowledge of reasoning, deduce and generate new concepts from the contents of the given text documents and thus give rise to "understanding". Programs for writing, encoded with the knowledge of grammar, utilize the resultant new concepts and the relationships between concepts to compose new sentences.

The remainder of this paper is organized as follows. Section II provides the reviews of the process for creating digital brains. Section III provides the details of our project to use digital brains for reading and writing. Section IV gives the conclusion and outlines future research.

II. Reviewing the Process for Creating Digital Brains

Digital brains must possess knowledge and abilities to process the knowledge. From a knowledge engineering [1] point of view, the process for creating digital brains involves the process of creating the knowledgebases and the inference engines. The choice of how to encode knowledge (knowledge representation) is the key feature that effects both the knowledgebases and the inference engines.

We first review using knowledge graphs as the structure for storing and representing knowledge. More than a simple data graph in which a collection of data is represented as nodes and edges, a knowledge graph is enhanced with representations of schema, identity, context, and rules [2]. There are various types of knowledge graphs including Resource Description Framework (RDF), heterogenous graphs, property graphs, and complex graphs [2]. The RDF model was developed to capture information on the web and is recommended by W3C. The underlying structure of an RDF graph is subject-predicate-object triples arranged as a directed edge-labeled graph. Property graphs allow additional flexibility by incorporating labels on both nodes and edges [2]. While the choice of representation affects the process of thinking, it is possible to convert from RDF to property graphs and vice versa [3]. We review how to create the knowledgebase and the inference engines in the next two subsections.

A. Knowledge Acquisition (Learning)

The process of learning is extracting knowledge from a source and representing that knowledge in a usable format. Knowledge can be extracted from structured data, text, and images. Extraction methods include manual extraction by human experts, as in the case of Cyc [4], fully automated machine learning techniques, and most commonly, a combination of the two.

The process of extracting knowledge from structured data begins with schema mapping. A semantic schema is used to define hierarchical classes of nodes and properties of edges.

Manuscript received March 18, 2023; revised April 24, 2023.

A. Clifton is with the Mathematics and Statistics Program, Louisiana Tech University, Ruston, LA 71272 USA (e-mail: aclifton@latech.edu).

B. Choi is with the Computer Science Program, Louisiana Tech University, Ruston, LA 71272 USA (e-mail: pro@benchoi.org).

*Correspondence: aclifton@latech.edu

Due to the semantic nature, when importing or combining knowledge graphs, careful consideration must be made to ensure a correct mapping between the existing schema and the schema of the new source. There exist automated techniques for suggesting these mappings, but they still require human intervention [5]. In the case of RDF graphs, International Resource Identifiers (standardized global identifiers of entities on the Web) and "sameAs" links are used for direct mapping [5].

Identity mapping is the next step in learning from structured data and involves verifying whether two entities in given knowledge graphs represent the same real-world entity. As in the case of schema mapping, there is not a fully automated process available for identity mapping, so human intervention is required. The process of identity mapping is to first block the data into a random forest then use active learning to randomly select pairs from the two datasets, use similarity functions to obtain features, and apply the learned rules to new selected pairs and iterate. Once the process terminates, a matching is proposed but must be verified by a domain expert [2].

The process of extracting knowledge from text relies heavily on Natural Language Processing for entity and relation extraction as well as entity resolution. The extracted entities will form the nodes of the knowledge graph while the extracted relations form the edges (see Fig. 1) [6]. To incorporate newly extracted knowledge to an existing knowledge graph, entity resolution is performed. Knowledge can also be extracted from unstructured data and relies on machine learning algorithms [5].

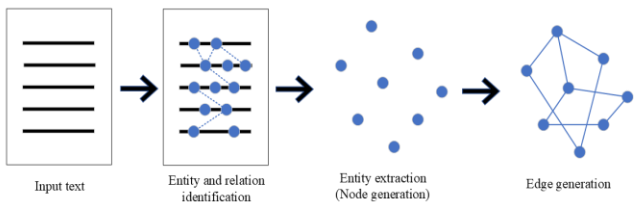

Input text
Entity and relation identification
Entity extraction (Node generation)
Edge generation

Fig. 1. Overview of knowledge graph extraction from text [6]

In its push to add common sense reasoning to AI, Cycorp used manual data extraction to construct its knowledge base beginning in 1984 [7]. Since its inception, the Cyc project has dedicated over a century of person-years to building its knowledgebase [4]. The language used to express the Cyc ontology is CycL which can be thought of as a full first-order predicate calculus [8]. The Cyc knowledgebase (Fig. 2) contains more than 3 million facts and rules [9, 10] and currently it is the largest known knowledge base with a focus on common sense [3]. The Cyc project is now considering machine creation for knowledge acquisition since its knowledgebase contains sufficient entities and predicates [9].

Wikidata's knowledge graph is a more recent example which also primarily used human curation for knowledge acquisition [3]. In contrast, the Amazon Product Graph and Microsoft Academic Graph relied more heavily on automated methods [3]. Although the Amazon Product Graph and Microsoft Academic Graph were able to leverage machine creation, human intervention was still required to create training data and perform verification [3].

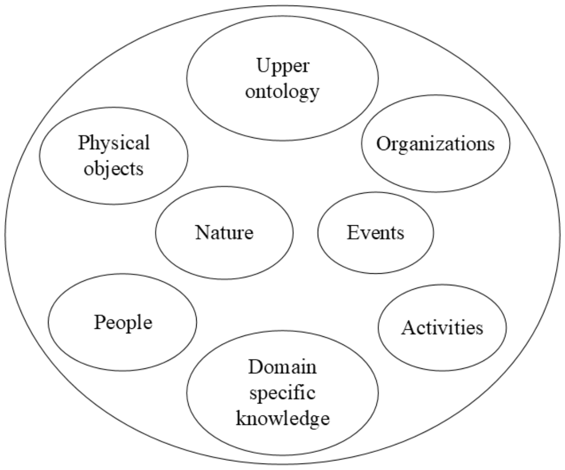


Fig. 2. Overview of Cyc Knowledge Base Topic Map [7]

Google's Assistant acquires knowledge from a variety of sources including Google's own knowledge graph, user interactions, and external APIs. Google's knowledge graph was built from Wikipedia and Freebase which used an RDF model [11].

B. Knowledge Discovery (Thinking)

"Thinking" (knowledge discovery) refers to the process of reasoning over the knowledge with the goal of generating new knowledge. When the knowledge is presented with knowledge graphs, the graphs can be queried for existing knowledge using SPARQL for RDF graphs or one of the many property graph querying languages such as Cypher, Gremlin, or G-CORE [2]. To generate new knowledge, techniques for deductive and inductive reasoning are used.

Deductive reasoning is a consequence of logical statements, but some conclusions can be drawn from the hierarchical nature of semantic schemas (for example, the subclass relationship) [2]. To perform deductive reasoning, first define an ontology which provides a precise definition of the terms within the domain of their use. The Web Ontology Language (OWL) is a W3C recommended RDF compatible ontology language. Ontology languages allow for the abstraction of a knowledge graph into a domain graph that is then combined with defined logical rules to form novel entailments (new conclusions that can be represented as new edges in the domain graph). OWL is defined under a No Unique Name Assumption (NUNA) and an Open World Assumption (OWA) which means that nodes in the knowledge graph may represent the same entity in the domain graph (NUNA) and that the domain graph may include entailments not realized by the knowledge graph provided no contradiction arises (OWA) [2]. Different ontologies may be defined under different assumptions.

Inductive reasoning is the process of generalizing an observed pattern. Inductive reasoning utilizes graph algorithms such as path finding, centrality detection, and community detection and ontology-based (or rule-based) algorithms [2]. Machine learning algorithms are implemented by first embedding the nodes and edges of the knowledge graph as vectors while preserving as much of the discrete structure as possible. Alternatively, machine learning algorithms can be constructed around the graph structure as in the case of Graph Neural Networks (GNNs) [2]. The following are example use cases of graph algorithms.

Consider a knowledge graph that represents a friend network. To find the set of friends of friends of Alice, begin at the node "Alice" and traverse each edge with the "friend" relation. The destination nodes will form the set of Alice's friends. Then, for each node in the set of Alice's friends, traverse each edge with the "friend" relation (do not traverse edges that were traversed in the first step). The destination nodes from the second edge traversal form the set of friends of friends of Alice. See Fig. 3.

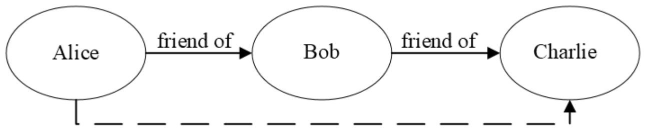


Fig. 3. The "friend of" path and the inferred "friend of friend" relation.

Fig. 4 represents a part of a knowledge graph containing information on the city of San Francisco, California. By traversing the "part of" edges, we can infer that San Francisco is a city in the United States.

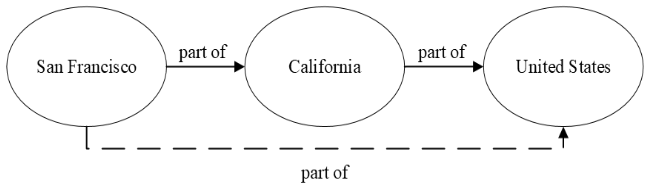


Fig. 4. Knowledge graph containing information on San Francisco with the inferred "part of" edge.

Centrality is a measure of the degree of a node, that is, the number of relations coming into and out of a particular node. Ways of measuring centrality include (1) finding the nodes with largest degree (the sum of the incoming and outgoing edges), (2) determining all of the shortest paths and finding the node that appears in the most such paths, (3) considering only a subset of nodes and finding the node in the original graph that is the shortest distance from all of the nodes in the subset, and (4) finding the node with the largest number of incoming edges [6].

A "community" in a graph refers to a group of connected nodes, meaning that there is a path between every two nodes in the community. In a directed graph, we say a set of nodes is strongly connected if for every two nodes, u and v, there is a directed path from u to v and a directed path from v to u. In graph terminology, such a community is said to be a strong component if the community is a maximal strongly connected subdigraph [2]. Finding the strong components of a directed graph begins by selecting a node and finding a depth first search tree with root the chosen vertex. Then it is determined how each edge not in the tree connects the nodes of the directed graph. The process for finding communities in undirected graphs is similar but has fewer possibilities for edge behavior [2].

Algorithms for community detection that leverage the information contained in a knowledge graph (versus a simple graph) include label propagation and the idea of modularity [2]. Label propagation begins by assigning a label (a community, in this case) to each vertex. Then the label of each vertex is updated based on the labels of its neighbors. The algorithm terminates when each vertex is in a community that is shared by most of its neighbors [5]. Modularity algorithms such as the Louvain method successively compare the number of relations between subsets of nodes to output a partition of the nodes into communities with the greatest density of relations [5].

To respond to user requests, intelligent personal assistants, like Alexa, use a combination of natural language generation and text-to-speech techniques. The natural language generation component uses a combination of template-based and machine learning-based techniques to generate the response text. Then the text-to-speech techniques component uses concatenative text-to-speech to convert the generated text into speech by concatenating pre-recorded samples [12].

III. Using Digital Brain For Reading And Writing

We attempt to extend the ability of virtual assistants by allowing them to read and write [13]. We make use of a digital brain (cyc.com) that is one of the world's largest knowledgebases and one of the most powerful inference engines to process documents conceptually to create abstractive summaries. Our system uses both the syntactic structure provided by the given documents and the commonsense knowledge provided by the knowledgebase. 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 (cyc.com). The Cyc development platform consists of the world's largest ontology of commonsense knowledge and a reasoning engine that allows information comprehension and abstraction. In addition, Cyc ontology serves as a backbone for semantic analysis, knowledge generalization, and natural language generation.

Our system conducts summarization process in three principal stages: knowledge acquisition, knowledge discovery, and knowledge representation (Fig. 5) [14]. The knowledge acquisition stage derives syntactic structure of each sentence of the input document and maps words and their relations into Cyc knowledgebase. 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 that was tested on various documents and webpages. The test results show that our system is capable of identifying key concepts and discovering main topics comprised in the original text, generalizing new concepts 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. The sentence was created as the result of linked key concepts. The linked concepts are then mapped back to words to form the sentence.

During the knowledge acquisition step (Fig. 5), the algorithm receives text documents as an input, performs syntactic analysis, and maps the words with their syntactic relationships into the Cyc knowledgebase. During the knowledge discovery step, the system performs a generalization of new concepts by propagating the concepts that were mapped into Cyc knowledgebase 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 step, the system generates new sentences using knowledge derived from the input text documents and the capabilities of the Cyc inference engine. These steps are described in more detail in the following three subsections.

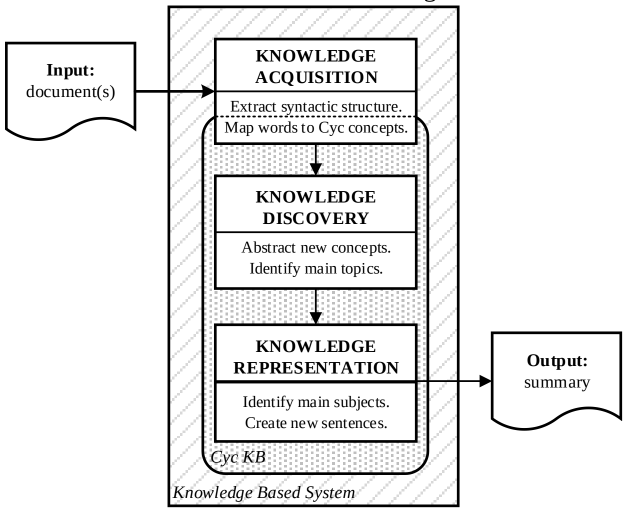


Fig. 5. System Workflow.

A. Knowledge Acquisition (Reading)

The knowledge acquisition step consists of two subprocesses. The first subprocess extracts the syntactic structures from the given documents (Fig. 6). This subprocess serves as a data preprocessing and transformation step. It normalizes raw text data and transforms it into syntactic representation.

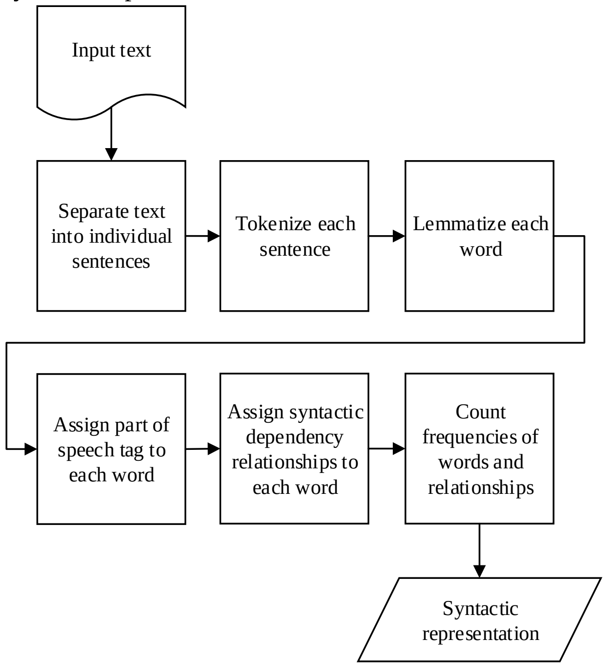

Input text
Separate text into individual sentences
Tokenize each sentence
Lemmatize each word
Assign part of speech tag to each word
Assign syntactic dependency relationships to each word
Count frequencies of words and relationships
Syntactic representation

Fig. 6. Syntactic structure extraction.

The second subprocess maps words from syntactic representation of the text to Cyc concepts (Fig. 7). Mapped Cyc concepts are utilized for reasoning during subsequent steps of the algorithm.

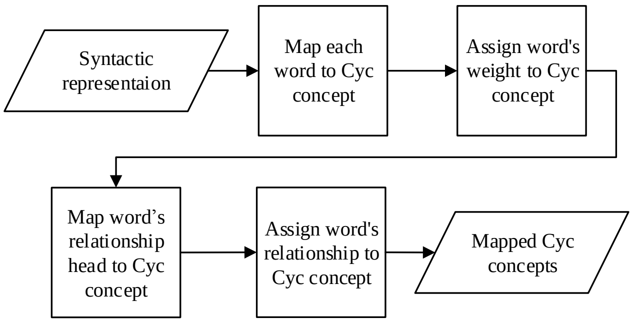

Syntactic representaion
Map each word to Cyc concept
Assign word's weight to Cyc concept
Map word's relationship head to Cyc concept
Assign word's relationship to Cyc concept
Mapped Cyc concepts

Fig. 7. Mapping words to Cyc concepts.

B. Knowledge Discovery (Thinking)

The knowledge discovery step performs two subprocesses: it abstracts new concepts and identifies main topics described in the input text. New concepts abstraction subprocess (Fig. 8) 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 [15]. 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.

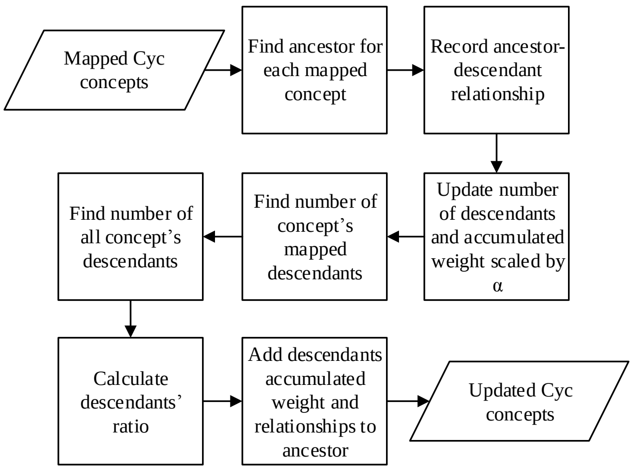
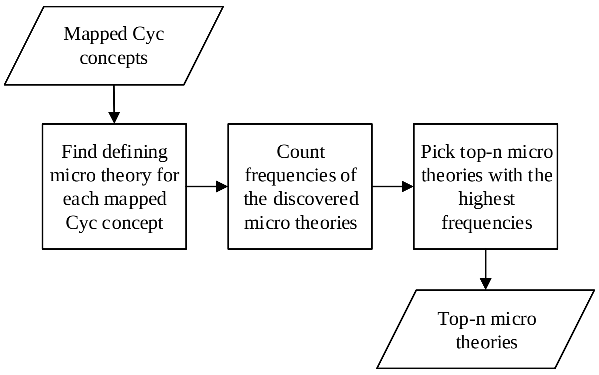

Mapped Cyc concepts
Find ancestor for each mapped concept
Record ancestor-descendant relationship
Update number of descendants and accumulated weight scaled by α
Find number of concept's mapped descendants
Find number of all concept's descendants
Calculate descendants' ratio
Add descendants accumulated weight and relationships to ancestor
Updated Cyc concepts

Fig. 8. New concepts abstraction.

The main topics identification subprocess (Fig. 9) 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.

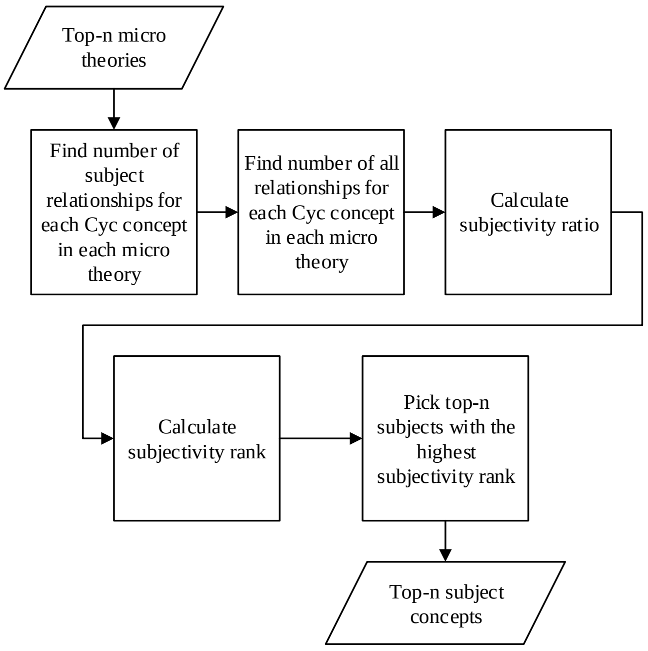


Mapped Cyc concepts
Find defining micro theory for each mapped Cyc concept
Count frequencies of the discovered micro theories
Pick top-n micro theories with the highest frequencies
Top-n micro theories

Fig. 9. Main topics identification.

C. Knowledge Representation (Writing)

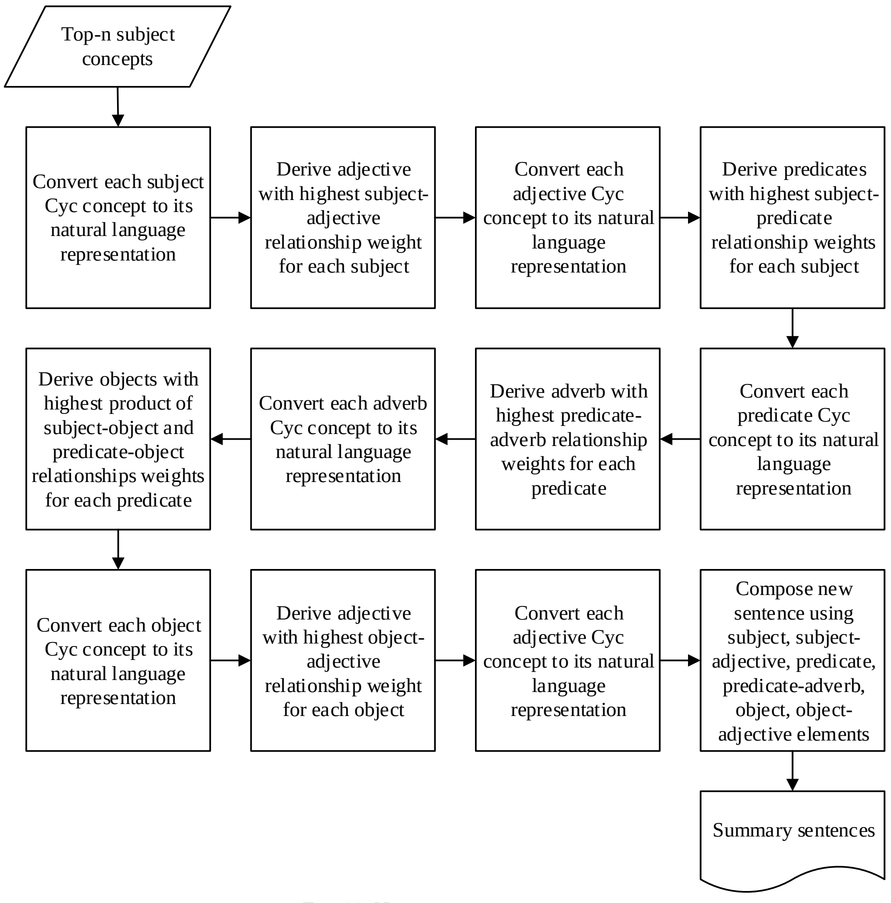
The knowledge representation utilizes powerful capabilities of the Cyc inference engine to generate new sentences based on the information discovered during knowledge acquisition and knowledge discovery steps. This step uses mapped and generalized Cyc concepts, their syntactic dependency relationships, and the most frequent micro theories as inputs. The knowledge representation step consists of two subprocesses: candidate subject discovery and new sentence generation. The candidate subject discovery subprocess (Fig. 10) identifies significant subject concepts out of all the mapped and generalized Cyc concepts [14].



Top-n micro theories
Find number of subject relationships for each Cyc concept in each micro theory
Find number of all relationships for each Cyc concept in each micro theory
Calculate subjectivity ratio
Calculate subjectivity rank
Pick top-n subjects with the highest subjectivity rank
Top-n subject concepts

Fig. 10. Candidate subject discovery.

The new sentences generation subprocess (Fig. 11) composes new sentences for each of the identified candidate subject concepts. The generated sentences serve as a final summary of the input text.



Top-n subject concepts
Convert each subject Cyc concept to its natural language representation
Derive adjective with highest subject-adjective relationship weight for each subject
Convert each adjective Cyc concept to its natural language representation
Derive predicates with highest subject-predicate relationship weights for each subject
Convert each predicate Cyc concept to its natural language representation
Derive adverb with highest predicate-adverb relationship weights for each predicate
Convert each adverb Cyc concept to its natural language representation
Derive objects with highest product of subject-object and predicate-object relationships weights for each predicate
Convert each object Cyc concept to its natural language representation
Derive adjective with highest object-adjective relationship weight for each object
Convert each adjective Cyc concept to its natural language representation
Compose new sentence using subject, subject-adjective, predicate, predicate-adverb, object, object-adjective elements
Summary sentences

Fig. 11. New sentence generation.

IV. Conclusion and Future Research

Now your virtual assistants can read documents and create summaries for you. We first analyze the digital brains of virtual assistants by reviewing their inner workings of learning and thinking. We then attempt to extend the ability of virtual assistants by allowing them to read and write. 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 knowledgebase shows great potential in creating abstractive summaries. Although our system can 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 creating a better digital brain that can make use of the World Wide Web as a knowledgebase. This is in fact the purpose of the first part of this paper, which reviews current systems for creating digital brains. Any virtual assistant is limited by the functionality and performance of the underlying commonsense knowledgebase. Our system is currently as knowledgeable as the capabilities of the Cyc knowledgebase 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.

We reviewed using formal models (such as graphs and logics) to encode knowledge. Another approach is to use neural networks to encode knowledge. Using a type of neural network called Generative Pre-trained Transformer (GPT) has achieved remarkable results. However, this neural network approach has its drawbacks such as the inability to acquire new knowledge after the completion of the training and the difficulties for anyone to understand or modify the underlying connections of the neural networks. A better digital brain might require a combination of neural networks and formal models (including graphs, logics, and automata [16, 17]). Much research remains to be done for building a better digital brain.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

BC developed the main concept of the paper and wrote the second part of the paper; AC wrote the first part of the paper. Both authors approved the final version.

References

[1] B. Choi, "Knowledge engineering the web," *International Journal of Machine Learning and Computing*, vol. 11, no. 1, pp. 68-76, 2021.

[2] A. Hogan, *et al.*, "Knowledge graphs," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–37, 2021.

[3] V. Chaudhri, *et al.*, "Knowledge graphs: Introduction, history and, perspectives," *AI Magazine*, vol. 43, no. 1, pp. 17–29, 2022.

[4] D. B Lenat. "CYC: A large-scale investment in knowledge infrastructure," in *Proc. Communications of the ACM*, vol. 38, no. 11, 1995, pp. 33–38.

[5] V. K. Chaudhri, CS520: Knowledge Graphs Seminar (Spring 2021). YouTube. url: https://www.youtube.com/playlist?list=PLDhh0lALedc5paY4N3NRZ3j_ui9foL7Qc.

[6] I. Melnyk, P. Dognin, and P. Das, "Knowledge graph generation from text," In: arXiv preprint arXiv:2211.10511 (2022).

[7] D. B Lenat and R. V. Guha, "Building large knowledge-based systems; representation and inference in the Cyc project," Addison-Wesley Longman Publishing Co., Inc., 1989.

[8] CYC. White Paper: Technology Overview. Tech. rep.

[9] C. Matuszek, *et al.*, "Searching for common sense: Populating cyc from the web," in *Proc. UMBC Computer Science and Electrical Engineering Department Collection*, 2005.

[10] M. J. Witbrock, *et al.*, "Knowledge begets knowledge: Steps towards assisted knowledge acquisition in cyc," in *Proc. AAAI Spring Symposium: Knowledge Collection from Volunteer Contributors*, 2005, pp. 99–105.

[11] N. Chah, "OK Google, what is your ontology? Or: exploring freebase classification to understand Google's knowledge graph," In: arXiv preprint arXiv:1805.03885 (2018).

[12] S. Karlapati, *et al.*, "Prosodic representation learning and contextual sampling for neural text-to-speech," in *Proc. ICASSP 2021*, 2021. url: https://www.amazon.science/publications/prosodic-representation-learning-and-contextual-samplingfor-neural-text-to-speech.

[13] B. Choi, A. Timofeyev, and A. Bobunova, "Teaching computers to read, understand, and write human languages," in *Proc. Languages and Migration in a Globalized World*, pp. 92-103, Dec. 2020.

[14] A. Timofeyev and B. Choi, "Knowledge based system for composing sentences to summarize documents," *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pp. 164-183, 2019.

[15] B. Choi and X. M. Huang, "Creating New Sentences to Summarize Documents," in *Proc. the 10th IASTED International Conference on Artificial Intelligence and Application (AIA 2010)*, 2010, pp. 458-463.

[16] B. Choi, "Automata for learning sequential tasks," *New Generation Computing: Computing Paradigms and Computational Intelligence*, vol. 16, no. 1, pp. 23-54, 1998.

[17] B. Choi, "Inductive inference by using information compression," *Computational Intelligence*, 2023.