Interpretable Natural Language Segmentation Based on Link Grammar

(Paper with reproducible results)

Vignav Ramesh
Research
SingularityNET Foundation
Saratoga, CA, USA
rvignav@gmail.com

Anton Kolonin
Research
SingularityNET Foundation
Novosibirsk, Russia
akolonin@gmail.com

***Abstract*— Natural language segmentation (NLS), or text segmentation, refers to the process of dividing written text into meaningful units. Sentence segmentation, a subfield of text segmentation, is the problem of dividing a string of natural language text into its component sentences. Current methods of sentence segmentation are often either hardcoded—they require manual implementation of fixed grammar and segmentation rules—or require extensive training on labeled corpora and are not explainable—they are "black box" algorithms that cannot be understood by humans. In this paper, we present a novel explainable sentence segmentation method capable of separating bodies of text into grammatically valid sentences solely based on the grammatical relationships between individual words or tokens. The proposed NLS architecture can both automate the input query parsing and semantic query execution components of voice-activated question answering and information retrieval systems as well as enable automatic summarization, entity extraction, sentiment identification, and a variety of other natural language processing (NLP) algorithms that operate at the sentential level.**

Keywords—explainable artificial intelligence, natural language segmentation, natural language processing, question answering, formal grammar

I. Introduction

A. Natural Language Segmentation

Natural language segmentation (NLS), otherwise known as text segmentation, refers to the process of dividing written text into meaningful units, such as words, sentences, topics, morphemes, or paragraphs. Text segmentation encompasses a variety of problems, such as word segmentation, intent segmentation, topic segmentation, and more; our work is concerned with sentence segmentation.

Sentence segmentation is the problem of dividing a string of natural language text into its component sentences. In this paper, we refer to sentence segmentation specifically in the context of separating bodies of text into grammatically valid sentences solely based on the semantic and morphological relationships between individual words, i.e. we do not use punctuation, capitalization, or other indicators of the beginnings or endings of sentences. Our approach is thus generalizable to various languages other than English, since not all written languages contain punctuation characters useful for approximating sentence boundaries.

B. Sources of Unsegmented Text

There are two primary sources of unsegmented text on which our proposed NLS algorithm can operate: the text output from speech-to-text (STT) recognition engines, which are often used to preprocess the inputs of question answering or chatbot pipelines, and crawled web pages. STT engines transcribe audio content and return sequences of tokens, excluding delimiters. Thus, our NLS architecture can be used to accurately segment the transcribed output into sentences. Similarly, web scraping, or webpage crawling, returns raw HTML and CSS output; the returned content often does not contain proper punctuation, and HTML or CSS tags are usually neither accurate nor consistent sentence boundary identifiers. Here, our NLS architecture can be used to segment crawled content without requiring the presence of delimiters.

C. Motivations

1) Interpretable Natural Language Processing

Explainable AI (XAI) refers to artificial intelligence (AI) methods whereby both the AI model and its results can be reasonably understood by humans. It contrasts with the idea of "black box" algorithms where the AI model and its outcome cannot be understood even by the developer. Interpretable natural language processing (INLP) is an application of the XAI concept to natural language processing (NLP); INLP allows for the learning of natural language (NL) material, comprehension of bodies of text, and generation of textual material via logical and transparent methods that rely on interpretable models of formal grammars such as Link Grammar [1]. Our NLS architecture is expected to serve as an INLP method allowing for the interpretable segmentation of large bodies of text into grammatically valid sentences. Our work provides both explainable results as well as an interpretable model for sentence segmentation, since we utilize Link Grammar which is an explainable formal grammar.

2) Unsupervised Language Learning

Deep neural networks (DNNs) and other existing methods of grammar learning often rely on substantial supervised training on large labeled corpora. However, humans can easily and quickly acquire explainable rules of sentence comprehension using grammar rules and conversational patterns and clearly understand semantic word categories. This gives rise to the idea of unsupervised language learning (ULL), which allows for the acquisition of grammar from unlabeled text corpora computationally and unsupervisedly. In ULL systems, an example being OpenCog's unsupervised grammar induction platform, the learned knowledge is stored in a human-readable representation [2].

There exist five primary components that comprise the ULL pipeline. First is the text pre-cleaner, which is responsible for preprocessing corpora with configurable cleanup and normalization options. The cleaned corpora are then passed to the sense pre-disambiguator, which determines the "senses" or meanings corresponding with each token and thereby performs word disambiguation. The text parser then parses the sentences of the corpora into tokens; afterwards, a grammar learner categorizes words and assigns rules to each word based on these parses. Finally, an evaluator measures the quality of the inferred grammar. The ULL's final output is a model of the human language; one such model is the Link Grammar database itself, which serves as the formal grammar utilized by our NLS architecture [3]. If not obtained from a ULL system, transparent grammatical models of any NL must be created manually by computational linguists and periodically maintained as the language develops over time.

3) Question Answering

Question answering, a subfield of information retrieval, is the problem of automatically answering questions posed by humans in the form of NL material via computational means. Within an explainable question answering pipeline, there exist two primary components: natural language comprehension (NLC), and natural language generation (NLG). NLC requires first parsing the question, or input query, using a knowledge base structured as a formal grammar, and then performing semantic interpretation (learning semantic relationships between tokens of the parsed query found in the previous step). In the context of this paper, we focus on voice-activated question answering pipelines, where the input query is captured as an audio recording and transcribed via an STT engine. The NLG component then involves semantic query execution (deriving the answer to the query using the relationships extracted during NLC) and sentence construction (building grammatically valid sentences from the words and relationships determined during query execution).

Our work is concerned with both parsing the question as well as semantic query execution. While parsing the input query, our NLS architecture can first separate the token sequence that represents the question into its component sentences, which can then be used to build a parse tree that is passed on to the semantic interpretation step. Semantic query execution matches relevant texts with the input query, often expressed as a set of keywords between which exist various semantic relationships. Given an NLP architecture that deals with the problem of matching the query with available documents (such as those available on the Internet), we focus on segmenting these documents into their grammatically valid component sentences. A subset of these sentences most relevant to the input query can then be passed on to the relationship extraction and sentence construction steps of the NLG component of the question answering pipeline [4].

D. Link Grammar

Link Grammar is based upon the existence of rules, corresponding to lexical entries or grammatical categories, whereby each rule describes a list of words corresponding to that rule and a set of defining disjuncts. Disjuncts correspond to sets of typed connectors that define the valid use of a given word, and each connector represents either the left or right half of a grammatical link of a given type. Two directionally opposite connectors of the same type form a link in a parsed sentence. For instance, if word A has the connector S+, this means that A can form an S link to its right; similarly, a word B with the connector S- can form an S link to its left. Thus, if A occurs to the left of B in a sentence, the two words can be connected via an S link [5].

A sentence refers to a set of tokens (words, punctuation, and other syntactic structures) that can be connected by matching up connectors between certain pairs of tokens. The Link Grammar database maps all common words of a given language to the connectors that define them.

Link Grammar requires additional constraints beyond the matching of connectors to be satisfied, specifically the planarity and connectivity metarules. Planarity means that links cannot cross; connectivity means that the links and tokens of a sentence must form a connected graph wherein all tokens are connected to each other via some path.

1) Link Grammar Database

The Link Grammar database contains dictionaries across more than 10 languages. The English database contains approximately 1,430 distinct word clusters—groups of words that share the same grammar rules—and 86,863 word forms. Each dictionary is structured as a hierarchical tree of files; the ".dict" file maps all common words to their defining connector expressions, referencing supporting files that contain additional word clusters. An example cluster, along with its associated connector expression, is shown in Fig. 3.

2) Why Link Grammar?

Besides Link Grammar, other APIs and grammar rule dictionaries exist, such as spaCy[1] and Universal Dependencies (UD)[2]. Both spaCy and UD rely on or are structured as a dependency grammar, which requires a head-dependent relationship, i.e. links must be directional; the English Link Grammar database does not require links to indicate direction, and thus can be applied to a greater variety of sentential forms (a sample UD parse with directional links is shown in Fig. 4.).

More importantly, Link Grammar enables grammar comprehension without requiring grammatical knowledge to be hardcoded into software programs as spaCy does, allowing the Link Grammar database to continually be updated and enhanced via any manual or automated input (such as a ULL system) without having to modify the client code. In considering this independence of Link Grammar from the programming language in which client architectures are built, our proposed Loader architecture creates extensive value in providing the first native Java support for Link Grammar. The human-readable and editable nature of Link Grammar allows our grammar induction algorithm to better serve as an XAI when integrated into the rest of our NLS architecture.

E. Prior Work

Bayesian text segmentation methods utilize a generative probabilistic model wherein a document is represented as a set of topics, each of which imposes a distribution over the vocabulary. Riedl and Biemann perform best among these methods; they define a coherence score between pairs of sentences, and compute a segmentation by finding drops in coherence scores between pairs of adjacent sentences [6].

Another notable text segmentation approach is GRAPHSEG (Glavaš et al., 2016), an unsupervised graph method, which performs competitively on synthetic datasets [7]. It is widely known for outperforming Bayesian approaches on the Manifesto dataset [8]. GRAPHSEG works by building a graph with sentences as nodes, where an edge between two nodes signifies that the sentences represented by those nodes are semantically similar. GRAPHSEG then computes a segmentation by heuristically finding maximal cliques of adjacent sentences [9].

For sentence segmentation specifically, Matusov et al. proposed a novel sentence segmentation algorithm capable of detecting sentence-like units (SUs) in a statistical machine translation (MT) framework [10]. Matusov et. al. built three different MT systems: a phase-based MT system without punctuation marks, an MT algorithm with implicit punctuation mark prediction, and a phrase-based MT system with punctuation marks. In each of these MT contexts with different rules for punctuation prediction, the sentence segmentation algorithm decides segment boundaries based on a log-linear combination of language model and prosodic

[1] https://spacy.io/usage/linguistic-features#sbd
[2] https://universaldependencies.org/introduction.html

features, and the length of each segment is optimized explicitly in a manner similar to that of the HMM model [11].

Favre et al. proposed a sentence segmentation system that solves for sentences in a text via a global classification problem, where unsuitable segmentations are ruled out based on their score when tested using a generative syntactic language model induced by a probabilistic context-free grammar (PCFG) from a syntactic parser capable of capturing long distance dependencies and providing the segmentation algorithm with global syntactic information in a lattice-based manner [12]. Specifically, the NLS system initially constructs a hypothesis segmentation using local features via a two-pass approach involving a word boundary level model based on prosodic and n-gram features as well as unsupervised domain adaptation; the candidate sentences are assigned syntactic language model scores which are combined with local low-level scores in a log-linear model. These scores are then used to compute a final segmentation.

Lamprier et al. proposed ClassStruggle, an algorithm for linear text segmentation on general corpora. ClassStruggle first creates an initial clustering of the sentences of the text with respect to their similarities; by computing a similarity score between pairs of sentences in the form of encoded vectors via a variant of the "Single Pass" algorithm, the model classifies sentences into different categories, which provide a global view on the semantic relationships existing in the text. Afterwards, the clusters evolve, by considering a notion of proximity and of layout in the text, in order to create groups based on contextual and topical commonality. ClassStruggle then segments the text based on the distribution of the occurrences of the members of each class, with boundaries created between sentences of different classes [13].

Recent studies have focused on conditional random field (CRF), a class of statistical modeling method used for structured prediction, and neural networks for sentence segmentation [14]. Wang et al. [15] and Hasan et al. [16] used CRF based methods to identify word boundaries in speech corpora datasets. Khomitsevich et al. proposed an architecture combining two models, one based on support vector machines to deal with prosodic information and the other based on CRF to handle lexical information, using a logistic regression classifier; while this architecture performs very competitively on speech corpora datasets, it relies on punctuation and other syntactic structures specific to the Russian language, thus preventing the algorithm from generalizing to other languages [17]. Xu et al. proposed a combination of CRF and a deep neural network (DNN) to identify sentence boundaries on broadcast news data [18].

II. Methodology

Our NLS architecture consists of two main classes: Loader and Segment. Loader, a utility program used by Segment to load Link Grammar into memory, is not specific to our NLS architecture but is rather a tool for storing Link Grammar locally that can be used by any algorithm. Fig. 5. displays our proposed NLS architecture, containing the workflow of Loader and Segment as well as the question answering pipeline into which these two components can be integrated.

A. Loader

Loader loads and stores the Link Grammar database in various classes for future usage. Specifically, the Dictionary class stores a list of Word objects; each Word object contains a unique Rule object; a Rule object contains a list of Disjunct objects; and each Disjunct object stores a list of connectors that constitute the legal use of the given word. Currently, the Loader architecture only supports the English language and does not handle more complex morphological structures, including those needed to support languages such as Russian that require extensive morphology usage.

The MAKEDICT function, which comprises the core of the Loader, utilizes the information in an array of lines obtained from the Link Grammar database to create an array of Dictionary objects that map each common word or phrase that Link Grammar supports to its associated rule. The basic structure of the MAKEDICT algorithm can be seen in Fig. 6.

After parsing the contents of the Link Grammar database, Loader calls MAKEDICT to obtain the Dictionary objects for use in the Segment class. As shown in Fig. 6., MAKEDICT extracts a Link Grammar rule from each element of *lines*, storing any new macros (single links that define larger connector expressions) in *macros*. The algorithm then replaces each macro within the rule with its expanded expression and assigns the modified rule to the word it defines.

B. Segment

Given the Dictionaries generated by Loader, the Segment class computes a segmentation for a string of text by first extracting the tokens (words and punctuation) from that text via the PROCESSSENTENCES function and then clustering those tokens into valid sentences as per the SEGMENT function.

As shown in Fig. 7., the SEGMENT function loops through the input array *tokens* and determines if certain subsets of *tokens* can form grammatically valid sentences via the ISVALID and CHECK functions. The CHECK function simply determines if the first two tokens following *arr* satisfy initial checks of the planarity and connectivity metarules (e.g., one partial connectivity check that CHECK implements involves ensuring that the first and second tokens after *arr* are capable of forming links to the right and left, respectively).

ISVALID determines if *arr* can form a grammatically valid sentence via Link Grammar rules by ensuring that every pair of consecutive words in *arr* can be connected via links given in the Dictionary objects created by the Loader. To do so, ISVALID uses the CONNECTS function, which returns a boolean value indicating whether its two parameters, the tokens *left* and *right*, can be linked together.

As shown in Fig. 8., CONNECTS determines the lists of rules *leftList* and *rightList* that correspond with *left* and *right*, and then checks if any Disjunct in any Rule in *leftList* matches with any Disjunct in any Rule in *rightList*.

An example segmentation query is as follows: SEGMENT("tuna is a fish eagle is a bird dog is a mammal") ["Tuna is a fish.", "Eagle is a bird.", "Dog is a mammal."]

III. Results

Our algorithm was tested twice on distinct corpora. We found that the accuracy of our results was affected primarily by the issue of grammatical ambiguity, which refers to situations in which the same word may take on different parts of speech (such as the word "saw" in its verb and noun forms). Consider the string of text "dad has a hammer mom has a hammer". It would be grammatically appropriate to segment this text into the two sentences "Dad has a hammer" and "Mom has a hammer" because the words "hammer" and "Mom" cannot be connected. However, this is only true for the noun form of "hammer"; the verb form of "hammer" can technically be linked to the object "Mom". The presence of these grammatically correct yet contextually wrong phrases decreased our sentence boundary identification accuracy.

Semantic, or word sense, disambiguation—a solution to the grammatical ambiguity problem—determines which "sense" or definition of a word is activated by that word's use in a particular context; for instance, the word "saw," when used in the sentence "The child saw a dog," will have a different sense—and thus different Link Grammar rules—than when used in the sentence "The carpenter is holding a saw." Current unsupervised semantic disambiguation methods, such as Goertzel et al.'s algorithm capable of inferring word senses and parts of speech from vectors built using a neural language model as a sentence probability oracle, include dictionary-based algorithms that utilize knowledge encoded in lexical resources to learn the senses of words [19]. Implementing one such method will be part of our future work.

Our algorithm was primarily tested on 88 sentences with words all part of SingularityNET's "small world" POC-English corpus[3]:

TABLE I. "SMALL WORLD" CORPUS NLS RESULTS

Metric	Result
Ground Truth (POC-English Corpus)	
Total number of sentences	88
Average sentence length	5.51136
NLS Algorithm Results	
Total number of sentences	87
Average sentence length	5.57471
Overall Statistics	
Runtime	57 sec
Number of sentences matching exactly	78
Number of sentence boundaries accurately identified	85/87
Accuracy of boundary identification	0.97701

Fig. 1. Results when tested on SingularityNET's "small world" corpus.

When tested on an excerpt of Lucy Maud Montgomery's "Anne's House of Dreams" as found in the Gutenberg Children corpus[4], our NLS algorithm performed as follows:

TABLE II. GUTENBERG CORPUS NLS RESULTS

Metric	Result
Ground Truth (Gutenberg Corpus)	
Total number of sentences	10
Average sentence length	13.2
NLS Algorithm Results	
Total number of sentences	11
Average sentence length	13.2
Overall Statistics	
Runtime	14 sec
Number of sentences matching exactly	7
Number of sentence boundaries accurately identified	7/9
Accuracy of boundary identification	0.77778

Fig. 2. Results when tested on "Anne's House of Dreams."

We compared our NLS architecture to three widely used open-source sentence segmentation frameworks: Syntok[5], PragmaticNet[6], and DeepSegment[7]. Syntok computes segmentations by recognizing "terminal markers," which are syntactic structures—such as periods, exclamation marks, question marks, etc.—that signify sentence boundaries. PragmaticNet is an unsupervised, opinionated, and conservative segmentation framework that, similar to Syntok, segments text into sentences based on punctuation, quotations, and parentheticals. Both Syntok and PragmaticNet identified zero boundaries in the text from both the POC-English and Gutenberg corpora, thereby returning one total sentence for each corpus and yielding an accuracy of 0%. Since Syntok and PragmaticNet rely on punctuation and other syntactic structures for segmentation, they are unable to segment bodies of text without punctuation. DeepSegment utilizes bidirectional long short-term memory networks (BiLSTMs) in a CRF based supervised segmentation model aimed at segmenting unpunctuated bodies of text into sentences. Even though it does not rely on punctuation, DeepSegment still identified only 1 segmentation boundary in the POC-English corpus and 2 boundaries in the Gutenberg corpus (all of which were accurate), yielding accuracies of 1.15% and 22.2%, respectively. Our proposed NLS architecture far exceeds these baselines and, most importantly, provides accurate support for unpunctuated bodies of text, thus generalizing to segmentation tasks across languages with varying syntactic structures.

IV. CONCLUSION

Our NLS architecture can primarily be applied to the semantic query execution component of the question answering pipeline; one such scenario is the Aigents Social Media Intelligence Platform [20]. Currently, the Aigents framework only handles written text in the form of oversimplified "pidgin" English; our NLS algorithm can enable Aigents to support question answering from spoken audio input transcribed by STT as well as extract information from crawled web pages to answer such questions and thus approach general conversational intelligence.

Another application of our NLS architecture is text simplification. Text simplification is the process of enhancing a corpus of human-readable text as to simplify the grammar and structure of the text while maintaining the original meaning. For instance, consider the unclear sentence, "Mom saw Dad, who saw Mom sawing." This text would be much simpler if split into the following three sentences: "Mom saw Dad," "Dad saw Mom," and "Mom was sawing." Our segmentation algorithm can be extended to perform text simplification, thereby improving the quality of corpora that would otherwise contain vocabulary and complex sentence constructions not easily processed via computational means.

More generally, our architecture can be applied to any NLP algorithms that operate at the sentential level, including automatic summarization (shortening a set of data structured as a body of text via computational means to create a subset, or summary, of that dataset containing the most important or relevant information within the original content), entity extraction (identifying and classifying important elements of a text into pre-defined categories), and sentiment identification (extracting emotions and opinions presented in a text).

Our further work will be dedicated to: 1) implementing semantic disambiguation; and 2) extending the algorithm's segmentation capabilities to languages other than English.

V. CODE AVAILABILITY

Our NLS architecture is open-source and available under the MIT License (a permissive, limited-restriction license) on GitHub at https://github.com/aigents/aigents-java-nlp.

[3] http://langlearn.singularitynet.io/data/poc-english/
[4] http://langlearn.singularitynet.io/data/cleaned/English/Gutenberg ChildrenBooks/capital/pg544.txt
[5] https://github.com/fnl/syntok
[6] https://www.tm-town.com/natural-language-processing
[7] https://github.com/notAI-tech/deepsegment

Additional Figures

```
ideas.n opinions.n statements.n beliefs.n facts.n arguments.n
principles.n theories.n philosophies.n signs.n impressions.n
conclusions.n contentions.n complaints.n proofs.n doubts.n
suspicions.n allegations.n reports.n claims.n announcements.n
positions.n risks.n hopes.n explanations.n restrictions.n threats.n
thoughts.n myths.n feelings.n discoveries.n rumors.n comments.n
realizations.n probabilities.n remarks.n notions.n convictions.n
hunches.n assumptions.n concepts.n hypotheses.n assertions.n
expectations.n implications.n perceptions.n proclamations.n
reassurances.n fears.n sayings.n senses.n messages.n disclosures.n
accusations.n views.n concerns.n understandings.n acknowledgments.n
acknowledgements.n possibilities.n premonitions.n prerequisites.n
prereqs.n pre-reqs.n pre-requisites.n
corequisites.n co-requisites.n coreqs.n co-reqs.n
provisos.n truisms.n fallacies.n assurances.n speculations.n
propositions.n prospects.n presuppositions.n inklings.n suppositions.n
findings.n amounts.n rules.n dangers.n warnings.n indications.n
answers.n suggestions.n:
  (<noun-modifiers> &
    (({{Jd-} & Dmc-} & {@M+} & {(TH+ or (R+ & Bp+)) & {[[@M+]]}} & {@MXp+} &
      (<noun-main2-p> or
      (Sp*t+ & <CLAUSE>) or
      SIp*t- or
      <rel-clause-p>)) or
    ({Dmc-} & <noun-and-p>) or
    Up- or
    (YP+ & {Dmc-}) or
    (GN+ & (DD- or [()])))) or
  [[AN+]];
```

Fig. 3. Example word cluster and associated connector expression, as seen in the English Link Grammar database.

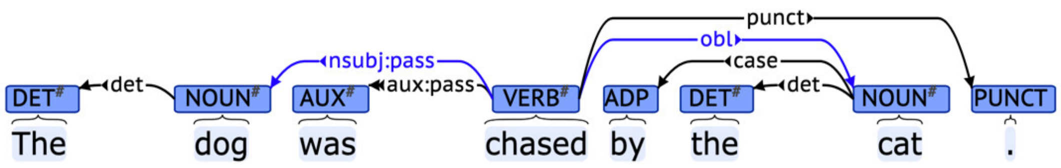
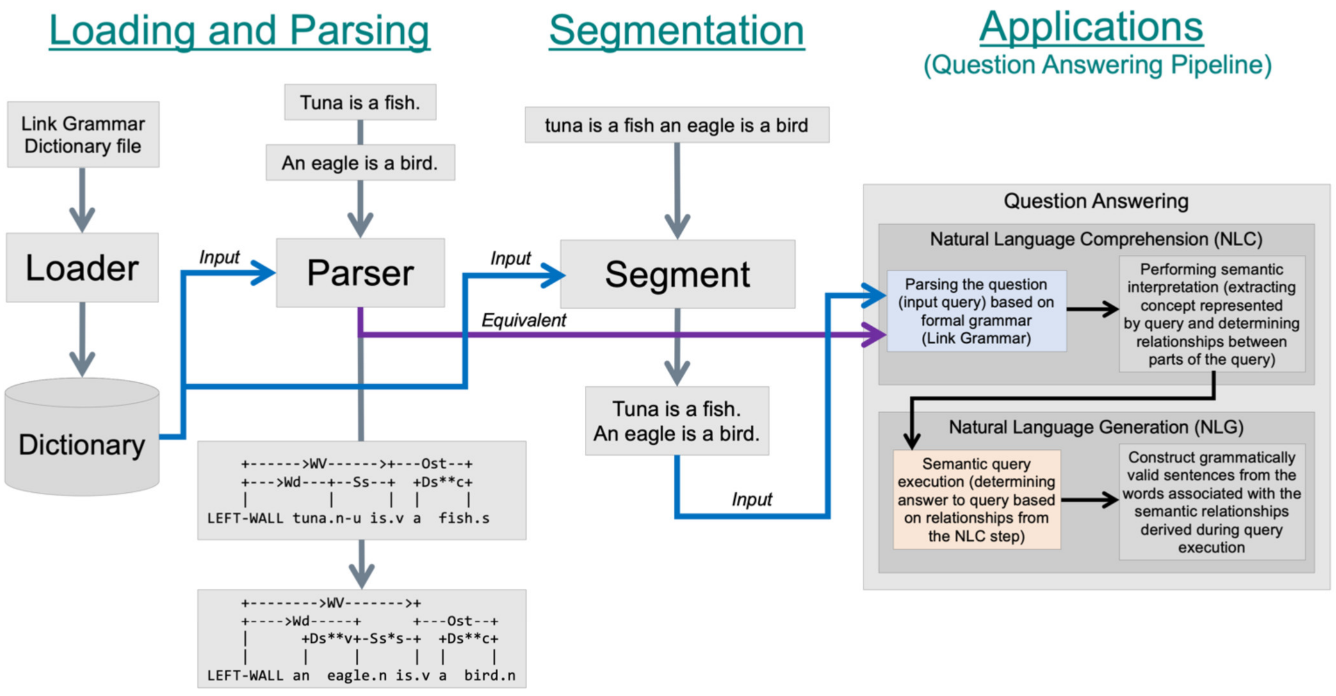


Fig. 4. UD parse of “The dog was chased by the cat.”


Loading and Parsing
Segmentation
Applications
(Question Answering Pipeline)
Link Grammar Dictionary file
Loader
Dictionary
Input
Tuna is a fish.
An eagle is a bird.
Parser
Input
Equivalent
tuna is a fish an eagle is a bird
Segment
Tuna is a fish. An eagle is a bird.
Input
Question Answering
Natural Language Comprehension (NLC)
Parsing the question (input query) based on formal grammar (Link Grammar)
Performing semantic interpretation (extracting concept represented by query and determining relationships between parts of the query)
Natural Language Generation (NLG)
Semantic query execution (determining answer to query based on relationships from the NLC step)
Construct grammatically valid sentences from the words associated with the semantic relationships derived during query execution

```
    +------>WV------>+---Ost--+
    +--->Wd---+--Ss--+  +Ds**c+
    |         |      |  |     |
LEFT-WALL tuna.n-u is.v a  fish.s
```

```
    +-------->WV------->+
    +---->Wd-----+      +---Ost--+
    |      +Ds**v+-Ss*s-+  +Ds**c+
    |      |     |      |  |     |
LEFT-WALL an  eagle.n is.v a  bird.n
```

Fig. 5. NLS architecture and question answering workflow.

Algorithm 1: MAKEDICT

```
Input  : An array $lines$ of all lines in the Link Grammar Database
Output: An array $[dict, hyphenated]$ of Dictionary objects, one for common words and
        one for common phrases with words separated by underscores

Initialize $dict$ and $hyphenated$
Initialize $macros$, which maps single links to the large connector expressions they define

define ASSIGN($w$, $r$):
if $w$ is a hyphenated phrase then
    Add $(w, r)$ to $hyphenated$
else
    Add $(w, r)$ to $dict$
end
end

for $line$ in $lines$ do
    if $line$ starts with a macro then
        Split the single link, $macro$, from its definition, $rule$
        Add $(macro, rule)$ to $macros$
    else
        if $line$ contains a filename $f$ then
            Parse $f$ to obtain the list of words it contains
            Replace all instances of macros in the rule $rule$ specified in the following lines
              of the Link Grammar database with their expanded definitions as contained in
              $macros$
            Store $rule$ in a Rule object $r$
            for word $w$ in $f$ do
                ASSIGN($w$, $r$)
            end
        else
            Split the word, $w$, from its definition, $rule$
            Process $rule$ and store it in a Rule object $r$
            Replace all instances of macros in $rule$ with their expanded definitions as
              contained in $macros$
            ASSIGN($w$, $r$)
        end
    end
end
return $[dict, hyphenated]$
```

Fig. 6. MAKEDICT algorithm.

```
Algorithm 2: SEGMENT
Input  : An array $tokens$ of words and commas extracted from the input text by
         PROCESSSENTENCES
Output: A list of sentences obtained by segmenting $tokens$ into grammatically and
         morphologically valid arrays of tokens

Start a counter $idx$, representing the index of the current token in $tokens$
Initialize an empty list $ret$, which will eventually contain the sentences that SEGMENT
  will return

while $idx < length(tokens)$ do
    for $i$ in $[idx, length(tokens)]$ do
        Create array $arr$ containing the subset of $tokens$ from indices $idx$ to $i$
        if ISVALID($arr$) and CHECK($tokens[i+1], tokens[i+2]$) then
            $threshold = n$ (default value of 2)
            Add $tokens[i+1], tokens[i+2]$ ... $tokens[i+n]$ to $arr$
            if ISVALID($arr$) and CHECK($tokens[i+n+1], tokens[i+n+2]$) then
                Construct a sentence from $arr$, i.e. create a string with the tokens in $arr$
                  separated by spaces and add appropriate punctuation
                Add the sentence to $ret$
                $idx \leftarrow i + n + 1$
            else
                Construct a sentence from the original value of $arr$
                Add the sentence to $ret$
                $idx \leftarrow i + 1$
            end
        end
    end
end
return $ret$
```

Fig. 7. SEGMENT algorithm.

```
Algorithm 3: CONNECTS
Input  : A pair of strings $left$ and $right$, representing the two words to potentially be
         connected
Output: An boolean value indicating whether $left$ and $right$ can be connected via valid
         Link Grammar rules

Obtain $leftList$, the list of rules corresponding with $left$ (i.e. the rule when $left$ is a verb,
  the rule when $left$ is a gerund, etc.), from the global Dictionary variables $dict$ and
  $hyphenated$
Obtain $rightList$ in a similar manner

for $leftRule$ in $leftList$ do
    for $rightRule$ in $rightList$ do
        Split $leftRule$ and $rightRule$ into lists of Disjuncts $ld$ and $rd$
        for $l$ in $ld$ do
            for $r$ in $rd$ do
                Replace all instances of '$-$' in $l$ with '$+$' and vice versa
                if $l = r$ then
                    return true
                else
                    continue
                end
            end
        end
    end
end
return false
```

Fig. 8. CONNECTS algorithm.

REFERENCES

[1] D. Grinberg, J. Lafferty and D. Sleator, “A Robust Parsing Algorithm For Link Grammars,” *arXiv Computing Research Repository (CoRR)*, August 1995.

[2] A. Glushchenko, et al., “Unsupervised Language Learning in OpenCog,” *11th International Conference on Artificial General Intelligence*, AGI 2018, Prague, Czech Republic, pp. 109–118, July 2018.

[3] D. Sleator and D. Temperley, “Parsing English with a Link Grammar,” in *Proceedings of the Third International Workshop on Parsing Technologies*. Association for Computational Linguistics, pp. 277–292, 1993.

[4] V. Prince and A. Labadié, “Text Segmentation Based on Document Understanding for Information Retrieval,” in *Proceedings of the 12th International Conference on Applications of Natural Language to Information Systems*. NLDB, pp. 295-304, June 2007.

[5] R. Lian, et. al, “Syntax-Semantic Mapping for General Intelligence: Language Comprehension as Hypergraph Homomorphism, Language Generation as Constraint Satisfaction,” *International Conference on Artificial General Intelligence*, Springer, Berlin, Heidelberg, pp. 158-167, December 2012.

[6] M. Riedl and C. Biemann, “Text Segmentation with Topic Models,” *JLCL*, vol. 27, January 2012.

[7] G. Glavaš, F. Nanni, and S. P. Ponzetto, “Unsupervised Text Segmentation Using Semantic Relatedness Graphs,” in *Proceedings of the Fifth Joint Conference on Lexical and Computer Semantics*. Association for Computational Linguistics, Berlin, Germany, pp. 125-130, August 2016.

[8] P. Lehmann, T. Matthieß, N. Merz, S. Regel, and A. Werner, “Manifesto Corpus Version 2015-1,” *WZB Berlin Social Science Center*, Berlin, Germany, 2015.

[9] O. Koshorek, et al., “Text Segmentation as a Supervised Learning Task,” arXiv:1803.09337 [cs.CL], March 2018.

[10] E. Matusov, A. Mauser and H. Ney, “Automatic Sentence Segmentation and Punctuation Prediction for Spoken Language Translation,” in *Proceedings of the International Workshop on Spoken Language Translation*. ISCA, November 2006.

[11] F. J, Och and H. Ney, “A Systematic Comparison Of Various Statistical Alignment Models,” *Computational Linguistics*, pp. 19–51, March 2003.

[12] B. Favre, D. Hakkani-Tur, S. Petrov and D. Klein, "Efficient sentence segmentation using syntactic features,” *2008 IEEE Spoken Language Technology Workshop*, Goa, pp. 77-80, 2008, doi: 10.1109/SLT.2008.4777844.

[13] S. Lamprier, T. Amghar, B. Levrat and F. Saubion, “ClassStruggle: A Clustering Based Text Segmentation,” in *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*. Association for Computing Machinery, Seoul, Korea, pp. 600-604, March 2007.

[14] M. Treviso, C. Shulby, and S. Aluísio, “Sentence Segmentation in Narrative Transcripts from Neuropsychological Tests using Recurrent Convolutional Neural Networks,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pp. 315-325, April 2017.

[15] X. Wang, H. Ng, and K. Sim, “Dynamic Conditional Random Fields for Joint Sentence Boundary and Punctuation Prediction,” *13th Annual Conference of the International Speech Communication Association (INTERSPEECH) 2012,* January 2012.

[16] M. Hasan, R. S. Doddipatla, and T. Hain, “Multi-pass Sentence-end Detection of Lecture Speech,” in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*. INTERSPEECH, pp. 2902-2906, January 2014.

[17] O. Khomitsevich, P. Chistikov, T. Krivosheeva, N. Epimakhova, and I. Chernykh, “Combining Prosodic and Lexical Classifiers for Two-Pass Punctuation Detection in a Russian ASR System,” in *Proceedings of the International Conference on Speech and Computer*. SPECOM, pp. 161-169, September 2015.

[18] C. Xu, L. Xie, G. Huang, X. Xiao, E. Chng, and H. Li, “A Deep Neural Network Approach for Sentence Boundary Detection in Broadcast News,” in *Proceedings of the Annual Conference on the International Speech Communication Association (INTERSPEECH)*. INTERSPEECH, pp. 2887-2891, January 2014.

[19] B. Goertzel, A. Suárez-Madrigal, and G. Yu, “Guiding Symbolic Natural Language Grammar Induction via Transformer-Based Sequence Probabilities,” arXiv:2005.12533v1 [cs.CL], May 2020.

[20] A. Kolonin, “Personal Analytics for Societies and Businesses: With Aigents Online Platform,” *2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, Novosibirsk, pp. 272-275, September 2017.