A Novel Framework for Semantic Oriented Abstractive Text Summarization

N. Moratanch[1] and S. Chitrakala[2]

[1]*Research Scholar, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai*
[2]*Professor, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai*
E-mail: tancyanbil@gmail.com; chitrakala.au@gmail.com

Received 01 February 2019;
Accepted 29 March 2019

Abstract

Internet continues to be the most communal of all the mass media turning information content from being scarce to superabundant that is evidenced by its increase tenfold every five years. A powerful text summarizer can aid in balancing this overload but generating quality summaries of the target content thereby reducing time and effort to mine the required information. The proposed system aims is to develop a Semantic Oriented Abstractive Summarization to generate abstractive summaries with increased readability and qualitative content. The contribution of our works are Joint Model Predicate Sense Disambiguation and Semantic Role Labelling termed as Joint (PSD+SRL) is proposed to better capture the semantic representation of text. The content selection involves semantic based content selection and feature extraction are selected by Genetic Algorithm. The proposed system can be very useful for the students who want to read a whole book in a short time. Our experimental study is carried out using DUC, a typical corpus for text summarization.

Journal of Web Engineering, Vol. 17_8, 675–716.
doi: 10.13052/jwe1540-9589.1784

Keywords: Abstractive Summarization, Predicate Sense Disambiguation, Semantic Role Labelling, Genetic algorithm, Language generation.

1 Introduction

Text summarization is a popular research area which cannot be sidelined due to the problem of the unprecedented information overload on the web. There is a popular saying "We are not starving for data but for drowning in data". It has increased the necessity of a more strong and powerful text summarizers. There is an ardent need for condensation of information from text and that can be achieved by text summarization by reducing the length of the original text and delivering only the relevant details to the end user. The text summarization is commonly of two types; extractive and abstractive. Extractive summarization intends to extract few sentences from the source document based on some statistical factors or scores using techniques of statistical analysis [24], various machine learning techniques [26], sentence extraction [25], and discourse structures. Extractive summarization usually focuses on sentence extraction rather than summarization. On the other hand, abstractive summarization results are more powerful than extractive summarization [32] because they generate the summary sentences based on their semantic meaning [36]. Hence this urges to a meaningful summarization task which is more accurate and coherent than extractive summaries.

Abstractive summarization is one of the primary goals of text summarization research, and of course is its greatest challenges too. It is worth to note that one of the literature reviews [29] even we conclude that "Abstractive paradigms will become one of the main challenges to solve" in text summarization. In building an abstractive summarization model, however, it is often hard to imagine where to begin and how to proceed in order to incorporate some kind of semantic interpretation of the sources documents to create a shorter text which contains only the relevant content for the task at hand.

Depending on the type of information being delivered summaries take five major forms namely: descriptive, evaluative, indicative, informative and aggregative. A descriptive summary describes both form content of the source text. An evaluative summary offers a critical response to the source text. Indicative summaries give abbreviated information on the main topics dealt in the document. Informative summaries provide digest for full document retaining the important details while reducing the length of it. Based on these categorization the applications of Automatic Text Summarization spans across many major use cases.

Automatic Text Summarization has proven to be very useful for Natural Language Processing (NLP) tasks such as Question Answering, Text Classification and other related fields of computer science, such as Information Retrieval. Geographical Information Retrieval being considered an extension of the information retrieval field is also benefited by automatic text summarization. A lot of attention has been paid to the development of automatic systems that specifically deals with the retrieval of geographic information, available in vast amount of unstructured documents in the Web for which text summarization system comes handy. The generation of summaries could be seamlessly integrated into these systems by acting as an intermediate stage, with the purpose of reducing the document length. In this way, the access time for information searching will be improved, while at the same time relevant documents will also be retrieved. Recently, Folksonomy systems also has gained its application in integration with summarization system.

Capturing Semantic Content [45] is the critical phase of any summarization framework. Our method which hits the aim of qualitative content selection and more readable summaries. Since the proposed work deals with Semantic Oriented methodology, semantic representation of text is built by using a joint model Markov logic network based semantic role labelling. In this semantic role labelling has been extensively enforced in text content analysis task such as, sentiment analysis [17], text retrieval [44], text categorization [39] and information extraction [9]. The strength of predicate sense disambiguation and semantic role labelling [43] is joint modelled to present the semantic representation of the text. Major focus of this work lies in: Enhanced Semantic Role Labelling using a Joint Model approach.

The Contributions of the proposed work are:

- Generate an abstractive summaries which summarize sections of news articles with control over content and structure in summaries generated.
- The model constructed for Semantic text representation (Joint Model of PSD+SRL) achieving enhanced content selection of abstractive summaries.
- The summary is produced by analyzing the text semantically and even complex sentences and paragraphs could be summarized effectively.
- The abstractive summarization is produced by analyzing sentences syntactically and semantically.
- The abstractive summary is produced by the language generator with the higher Rouge Values.

Rest of the paper is organized as follows: In 2, discuss the literature survey pertaining to abstractive summarization. System architecture and details about the proposed model in described in Section 3. Experimental results are discussed in Section 4 followed by conclusion and future work in Section 5.

2 Related Works

The literature survey is made of two portions, namely; Structure/Syntactic based Abstractive Summarization (by using prior knowledge) and semantic based abstractive summarization (by using natural language generation techniques).

Since the focus of this paper is semantic based abstractive text summarization approach, the following works discuss various advancements made in this research area.

Munot et al. [37] compared numerous text summarization techniques for extraordinary sort of programs and recognition for information condensation. The problem of information overload can be solved using strong textual content summaries of the report that can assist the customers. They have discussed two main categories of text summarization methods: extractive and abstractive summarization methods and have supplied a taxonomy of summarization structures and statistical and linguistic techniques for summarization.

Barros et al. [5] presented a narrative abstractive summarization (NATSUM) method that produces narrative chronologically ordered summary about a target entity from several news documents related to the same topic. Consequences suggest that narrative abstractive summary rather than just a timeline summary offers better applicable data that is generated from the information found in distinct sources.

Ng et al. [38] proposed ROUGE-WE (ROUGE word embeddings) that permitsto head past surface lexicographic matches, and capture as a substitute the semantic similarities among words used in a generated summary and a human-written model summary. This improved ROUGE-WE metric produces precise correlations with human checks, measured with the Spearman and Kendall rank coefficients.

Liu et al. [28] presented an Abstract Meaning Representation (AMR) framework in which a set of AMR graphs remodeling into a summary graph through which then the textual content is generated. This framework provides a structured prediction algorithm that transforms semantic graphs of the input into a single summary semantic graph.

Chen et al. [7] proposed an accurate and speedy summarization model uses a sentence-level policy gradient technique to bridge the nondifferentiable computation between two neural networks in a hierarchical way while maintaining language fluency.

Liao et al. [27] studied the feasibility of using Abstract Meaning Representation (AMR) for multi-document summarization that condenses source documents to a set of summary graphs following the AMR formalism.

Kallimani et al. [19] discussed various different statistical approaches for abstractive Summarization in the Telugu language to create abstractive summaries by creating new sentences in the past work of extractive summarization methods.

Zhang et al. [47] proposed an abstractive cross-language summarization framework to target language summary by merging multiple bilingual PAS structures the use of integer linear optimization that tries to maximize the salience score and the translation high-quality concurrently.

Azmi et al. [3] presented a novel generic abstract summarizer for a single document in the Arabic language that segments the input text topic wise and extractive summary for each segment and abstractive summary is generated by applying rule-based reduction technique.

Saggion et al. [41] discussed various automatic text summarization and evaluation methods. Automatic text summarization relies on extractive methods that select the most relevant sentences from the collection of original documents in order to produce a condensed text rendering important pieces of information.

Khan et al. [23] presented a semantic graph approach that improves graph ranking algorithm by incorporating PASs semantic similarity and the two kinds of PAS relationships. Existing graph-based approaches consider sentence as a vector of words and cannot detect semantically equivalent redundant sentences, as they mostly rely on a content similarity measure.

Genest et al. [12] work on a framework for abstractive summarization focusing on a module for text generation. Each sentence of the summary is generated a document sentence containing a smaller amount of information and fewer words.

Guo et al. [15] proposed LDA-based Semantic Emotion-Topic Model(SETM) to enhance the retrieval performance of emotion mining models in social media that uses statistical-based and graph-based approaches and also semantically interpret the mining results considering inter-words relations.

Bakaev et al. [4] proposed a WUI measurement platform that supports both code-based and image-based WUI analysis to perform statistical analysis of the different metrics related to complexity-related subjective evaluations obtained from 63 human subjects of various nationalities.

Avanija et al. [2] proposed an ontology-based clustering algorithm that uses semantic similarity measure and Particle Swarm Optimization (PSO) to optimize the result of annotated documents. This method recovers documents based on their annotation features and relations using the Jena API and GATE tool API.

Lexicon based approach called senticircles is used in the system and sentimental analysis is done at entity-level as well as tweet-level [42]. The Twitter dataset is used for the sentiment analysis. Co-occurrence patterns in different tweets are taken into account in senticircles. Senticircles shows higher efficiency for partial sentences not for the complete context as in our system.

The sentence similarity measure is used in the system for extractive summarization [33]. Similarity measure could be used in the semantic analysis of the sentences for our proposed system. Since the similarity measure shows the significant improvement in Rouge measure, it could also be used in our system to avoid cluttering of context in the abstractive summary.

Greenbacker et al. [14] proposed a multimodal model for summarization. A Multimodal document contains both text and images. Firstly, a semantic model is constructed using knowledge representation based on objects (concepts) organized by Ontology. Secondly, informational content (concepts) is rated based on information density metric. The computed metric determines the relevance of concepts based on completeness of attributes, the count of relationships with other concepts and the count of expressions showing the occurrence of concept in the current document. Thirdly, the important concepts are expressed as sentences. The expressions observed by the parser are stored in a semantic model for expressing concepts and relationship. The limitation of this framework which does not employ semantic model for text representation is dependent on ontology network.

Moawad et al. [35] devised a model which represents the input document semantically using Rich Semantic Graph (RSG). RSG is an ontology based representation where the graph nodes are instances of ontology nouns and verbs. It reduces the generated RSG of the source document to a more reduced version using by some heuristic rules. Finally, the system generates the abstractive summary from the reduced rich semantic graph. But the system is heavily dependent on ontology rules.

Lloret et al. [30] built a model for generating ultra-concise concept-level summaries. The system initially converts the input document into its syntactic representation after lexical analysis. Summary generation is achieved using language generation tool with the lexical units as the input. This system lacks semantic representation of text and works on an assumption that all the sentences are anaphora resolved. Deeper semantics involving word sense disambiguation is also not incorporated in the model.

Genest et al. [13] proposed a framework for abstractive summarization which consists of following modules; Information item retrieval, Sentence generation, Content selection and Summary generation. In Information Item (INIT) retrieval, first syntactic analysis of text is done. So, an INIT is defined as a dated and located subject-verb-object triple. In sentence generation module, a sentence is generated from INIT by using a language generator. The content selection module ranks the sentences generated from INIT based on average Document Frequency (DF) score. Finally, a summary generation step accounts for the planning stage and includes dates and locations for the highly ranked generated sentences. The Major drawback of this model is that the candidate Information items are rejected because it fails to generate meaningful and grammatical sentences. Also, linguistic quality of summaries is very low due to incorrect parses of the input text.

Khan et al. [21] proposed a framework for abstractive summarization of multiple documents in the form of semantic representation of the source documents. Content selection is done by the ranking of the most significant predicate argument structures. Finally, summary is generated using a language generation tool. But the system does not handle more detailed semantics in the summarization approach due to its assumption that the text to be handled are anaphora resolved and sense disambiguated.

Hou et al. [16] worked on text summarization, based on reader's understandability about the document which conveys the reader's efficiency. The baseline of this method is to find the decision variable for controlling the optimization problem and text summarizing in the context. The objective function helps to summarize the high quality of information to reduce the reading time in critical information. The constraints summarization represents the mathematical problem such as representation and allocation rule for context selection.

Karwa et al. [20] proposed a differential evolution algorithm which summarizes the important sentence in the document from the clustering, to recall the abstractive information. After that differential evolution algorithm helps to find the optimal value in large search engine. Cosine similarity eliminates the similarity between the same sentence. Fitness function achieves the compactness in the same clustering and separability in the different clustering.

Issues under Semantic based abstractive text summarization are as follows:

- Context of the text not considered
- Lacks a model to include an abstract representation for content selection
- Semantic representation over syntactic representation of text
- Human expert required to construct domain ontology
- Manual effort required for framing of heuristics rules
- Absence of Predicate Sense Disambiguation (PSD) in summarization process
- Absence of Anaphora Resolution (AR) in summarization process

By focussing on the above-mentioned issues, semantic oriented abstractive summarization takes a great leap forward for text understanding phase and the summary generation phase incorporates the semantics of the content thereby delivering more meaningful summary. Semantic oriented abstractive summary is a difficult task because it does not exist an ideal summary for a given document or set of documents. Thus it renders the task of evaluating the summary quality as a very ambitious task. The objective of our proposed system is to implement a Semantic Oriented Abstractive Summarization System that would generate abstractive summaries of news articles with semantic rich coherency in content. An efficient summarization approach is to be implemented by the system incorporating NLP techniques such as Enhanced Predicate Sense Disambiguation.

3 Proposed System

As shown in Figure 1, the system "Semantic Oriented Abstractive Text Summarizer" is accomplished to generate abstractive summaries maintaining control over content and structure of the summaries generated. Semantic text representation (Joint Model of PSD + SRL) is to be used for content selection of abstractive summary generation. The entire pipeline involved in a summary generation would be geared towards achieving better quality in the summary content.

Summarization process would involve major phases such as content representation, content selection and language generation. The proposed system "Semantic Oriented Abstractive Summarizer" deals each of the above mentioned phases with its corresponding techniques required.

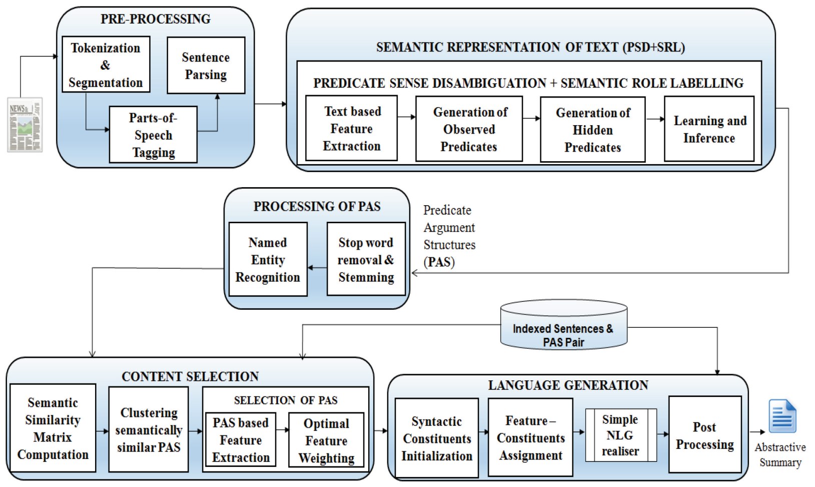

PRE-PROCESSING
Tokenization & Segmentation
Sentence Parsing
Parts-of-Speech Tagging
SEMANTIC REPRESENTATION OF TEXT (PSD+SRL)
PREDICATE SENSE DISAMBIGUATION + SEMANTIC ROLE LABELLING
Text based Feature Extraction
Generation of Observed Predicates
Generation of Hidden Predicates
Learning and Inference
PROCESSING OF PAS
Named Entity Recognition
Stop word removal & Stemming
Predicate Argument Structures (PAS)
Indexed Sentences & PAS Pair
CONTENT SELECTION
Semantic Similarity Matrix Computation
Clustering semantically similar PAS
SELECTION OF PAS
PAS based Feature Extraction
Optimal Feature Weighting
LANGUAGE GENERATION
Syntactic Constituents Initialization
Feature – Constituents Assignment
Simple NLG realiser
Post Processing
Abstractive Summary

Figure 1 Overall architecture diagram for proposed system.

In contrast the aim of our work is to get coherent and more accurate summaries semantic representation of text is built using joint model incorporating predicate sense disambiguation and semantic role labelling that results in increased readability and content selection.

Pre-processing of the input data involves tokenization and segmentation of input sentences, followed by the parts-of-speech tagging and the sentence parsing. Pre-processed data is stored in specific format indexed with respect to each sentence.

Semantic representation of text is obtained by a joint model incorporating predicate sense disambiguation and semantic role labelling. This phase involves the formulation of observed and hidden predicates for Markov Logic Network to form the predicate argument structures. Further, an additional level of processing is executed on the PAS.

The Content selection phase is critical for summary generation, therefore in this system, it is dealt with deeper analysis. Semantic similarity between predicate argument structures is computed and built into a similarity matrix. A type of hierarchical clustering is used to cluster semantically similar predicate argument structures using the similarity matrix. Feature extraction is performed, based on the text features extracted from predicate argument structures. The optimal feature weighting helps in allocating weights with respect to nature and significance of each feature. Natural Language Generation (NLG) realizer system is used for generation of summary sentences from high scored target predicate argument structures.

Figure 1 shows the system architecture depicting the abstractive summarization model.

3.1 Text Pre-Processing

Pre-processing phase is very critical to any system as indicated by a popular saying "garbage in, garbage out". In this work, the target document is fed as input which comprises of a set of sentences pertaining to any topic chosen by the user. Pre-processing methods applied in this system are (i) Tokenization and Segmentation (ii) Part-of-speech tagging (iii) Sentence Parsing. Sentence splitting may seem to be a trivial task in NLP but with the presence of email addresses, different punctuation marks and unknown abbreviations it may not seem trivial. Input document is split into sentences and for each sentence its corresponding attributes that is required for the system is extracted and stored in a file. The storage structure used in this work is in form of Extensible Markup Language (XML) files in order to cater to the need of storing details indexed with its sentence identifier. Figure 2 depicts the pre-processing process followed in this work.

Algorithm 1 explains each of the above mentioned pre-processing steps and results are stored as XML files for further modules to act on.

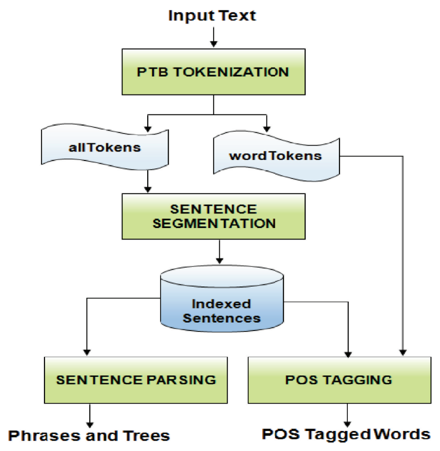


Figure 2 Pre-processing for input text document.

Algorithm 1: Pre-processing for input text document

```
Input: Input Text
Output: POS Tagged words, phrases & Trees
1  PTB Tokenizer
2  Initialize: S represent as Sentence
3  begin
4      allTokens[ ] ← PTBTokenizer(input text)
5      wordTokens[ ] ← filterWords(allTokens)
6      sentences[ ] ← postProcess(allTokens)
7      for each s in sentences do
8          store (s, wordTokens)
9      end
10     posWords[ ] ← POSTagger(s, wordTokens)
11     phrases[ ] ← LexicalizedParser(s)
12 end
```

3.1.1 Tokenization and segmentation

A fast, rule-primarily based tokenizer implementation, which produces penn treebank (PTB) style tokenization of english textual content. The tokenization is the process of chunking the given document into meaningful words or phrases, called tokens. A token is an instance of a sequence of characters in the given document that are grouped together as useful semantic units for processing. Typically, tokenization results in chunking of the sentences to words. Before chunking, punctuations are removed. Sentences are formed by joining the words with spaces, so, words are chunked by sub-setting the sentences by spaces. Regular Expression is used for these kinds of chunking. Altogether, tokenisation results in the list of words in the given document without punctuations. It is based on the type, after the tokenization is performed, sentence segmentation is performed. Sentence segmentation is dividing the running text into sentences. The presence of the punctuation marks and abbreviation convolutes the task.

3.1.2 Parts of Speech Tagging

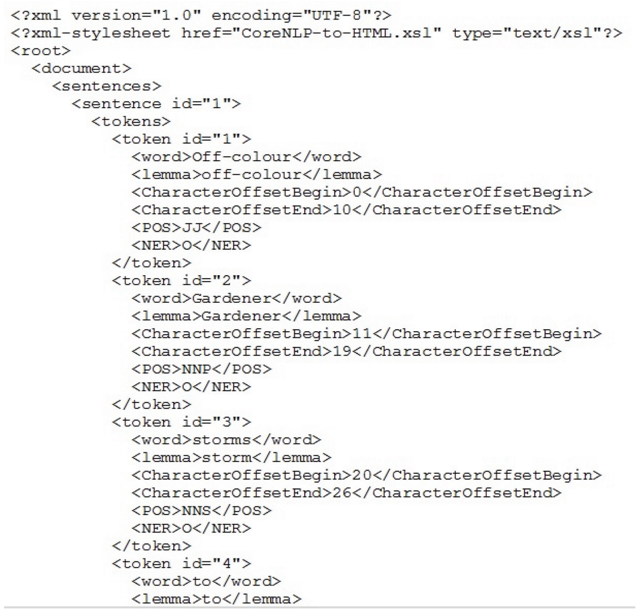
Parts of Speech Tagging involves tagging the words with the sentence offset, word offset and parts-of-speech of the word with other morphological, lexical and syntactic features. The efficiency of anaphora resolution system depends on POS tagging phase since the errors during POS tagging propagates down the resolution phase which will eventually subvert the performance of the system. For the purpose of POS tagging, Stanford POS tagger is used since it is easy to use and readily available.

3.1.3 Sentence parsing

Sentence parsing is the process of analyzing a string of symbols, either in natural language, computer languages or in data structures to conform the rules of a formal grammar and to find structural relationships between words in a sentence. It is the task of recognizing a sentence and assigning a syntactic structure to it. The most widely used syntactic structure is the parse tree which can be generated by using some parsing algorithms. These parse trees are useful in various applications like grammar checking or more importantly it plays a critical role in the semantic analysis stage. These dependencies show the importance of a word and how each word is related to different words that are placed in different positions of the tree.

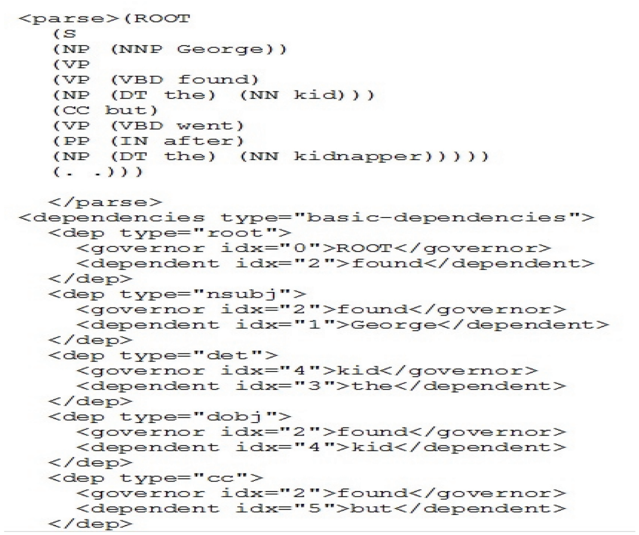
Figure 3 shows the XML file structure in which the pre-processed data is written. For each token of a sentence, attributes like lemma, character offset start-index and end-index, parts-of-speech label and named entity recognition label would be stored along with sentence ID and token ID.

This form of storage is desired and beneficial because throughout the implementation it is required to backtrack these attributes and link to its respective sentence.



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="CoreNLP-to-HTML.xsl" type="text/xsl"?>
<root>
  <document>
    <sentences>
      <sentence id="1">
        <tokens>
          <token id="1">
            <word>Off-colour</word>
            <lemma>off-colour</lemma>
            <CharacterOffsetBegin>0</CharacterOffsetBegin>
            <CharacterOffsetEnd>10</CharacterOffsetEnd>
            <POS>JJ</POS>
            <NER>O</NER>
          </token>
          <token id="2">
            <word>Gardener</word>
            <lemma>Gardener</lemma>
            <CharacterOffsetBegin>11</CharacterOffsetBegin>
            <CharacterOffsetEnd>19</CharacterOffsetEnd>
            <POS>NNP</POS>
            <NER>O</NER>
          </token>
          <token id="3">
            <word>storms</word>
            <lemma>storm</lemma>
            <CharacterOffsetBegin>20</CharacterOffsetBegin>
            <CharacterOffsetEnd>26</CharacterOffsetEnd>
            <POS>NNS</POS>
            <NER>O</NER>
          </token>
          <token id="4">
            <word>to</word>
            <lemma>to</lemma>
```

Figure 3 Pre-processing – token wise attributes.



```
<parse>(ROOT
  (S
  (NP (NNP George))
  (VP
  (VP (VBD found)
  (NP (DT the) (NN kid)))
  (CC but)
  (VP (VBD went)
  (PP (IN after)
  (NP (DT the) (NN kidnapper)))))
  (. .)))

  </parse>
<dependencies type="basic-dependencies">
  <dep type="root">
    <governor idx="0">ROOT</governor>
    <dependent idx="2">found</dependent>
  </dep>
  <dep type="nsubj">
    <governor idx="2">found</governor>
    <dependent idx="1">George</dependent>
  </dep>
  <dep type="det">
    <governor idx="4">kid</governor>
    <dependent idx="3">the</dependent>
  </dep>
  <dep type="dobj">
    <governor idx="2">found</governor>
    <dependent idx="4">kid</dependent>
  </dep>
  <dep type="cc">
    <governor idx="2">found</governor>
    <dependent idx="5">but</dependent>
  </dep>
```

Figure 4 Pre-processing – parse tree and dependency tree.

Figure 4 shows the parse tree and dependency tree for a sample sentence. Parse tree of a sentence gives a rooted ordered tree structure based on its syntactic constituents. And a dependency trees depicts the basic dependencies that exist within the tokens of each sentence.

3.2 Semantic Representation of Text (Joint Model Predicate Sense Disambiguation + Semantic Role Labelling)

Semantic representation of text is obtained by extracting the predicate argument structures by the process of semantic role labelling. The objective of semantic role labelling is to determine the syntactic constituents of a sentence with respect to its predicate and identifing the semantic roles played such as Agent, Direct and Indirect Object, Temporal marker etc. Primary focus is on identifying the semantic role between the predicate and arguments. Since abstractive summarization requires a deeper analysis of text, the proposed system implements an efficient joint model for semantic role labelling termed as joint model (PSD+SRL). Motivation behind incorporating predicate sense disambiguation is from the benefits achieved in the efficiency of SRL task [6]. This Joint model is achieved through the principles of Markov logic. The Markov Logic model (MLM) jointly labels and disambiguates all the predicate senses. Usually sense disambiguation and semantic role labelling is regarded

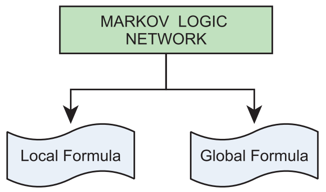


Figure 5 Markov logic network.

Algorithm 2: Algorithm for Joint Markov Logic Model

```
    Input: Indexed Sentences with POS tagged words, Phrases and Trees
    Output: Predicate Argument Structures(PAS)
1   begin
2       extract text level local features $F_i$
3       generate local formula based on $F_i$ using Equation (1–5)
4       generate global formula based on $F_i$ using Equation (6–10)
5       prepare the input data files $I_i$
6       $I_i \leftarrow$ Evidence files, query files
7       trigger learning and inference of MLM convert the output data files
8   end
```

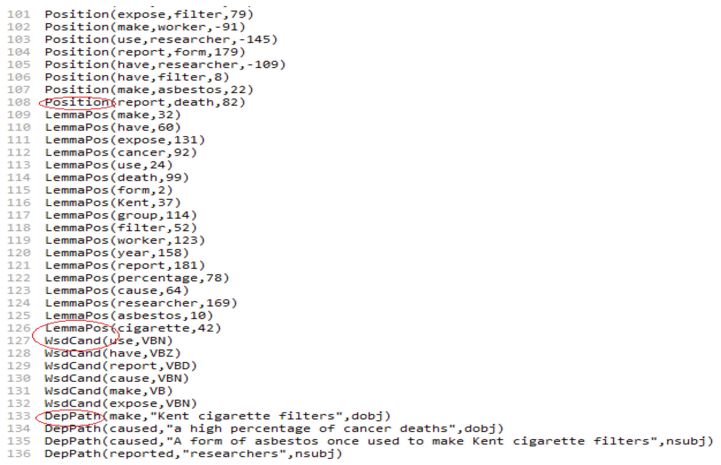
as two separate tasks performed in isolation. But over recent years, Markov Logic has become a powerful framework for joint models.

Above Figure 5 depicts that the Markov logic network is comprised of weighted local and global formula. The following Algorithm 2 discuss the steps involved in joint Markov modelling for semantic role labelling.

Local Features Category:

Following are the features categories that are extracted which aids in formulating the local Markov logic network.

- Argument Related
- Predicate Related
- Word and Predicate Related
- Argument Related Sense
- Predicate Related Sense
- Word and Predicate Related Sense



```
Position(expose,filter,79)
Position(make,worker,-91)
Position(use,researcher,-145)
Position(report,form,179)
Position(have,researcher,-109)
Position(have,filter,8)
Position(make,asbestos,22)
Position(report,death,82)
LemmaPos(make,32)
LemmaPos(have,60)
LemmaPos(expose,131)
LemmaPos(cancer,92)
LemmaPos(use,24)
LemmaPos(death,99)
LemmaPos(form,2)
LemmaPos(Kent,37)
LemmaPos(group,114)
LemmaPos(filter,52)
LemmaPos(worker,123)
LemmaPos(year,158)
LemmaPos(report,181)
LemmaPos(percentage,78)
LemmaPos(cause,64)
LemmaPos(researcher,169)
LemmaPos(asbestos,10)
LemmaPos(cigarette,42)
WsdCand(use,VBN)
WsdCand(have,VBZ)
WsdCand(report,VBD)
WsdCand(cause,VBN)
WsdCand(make,VB)
WsdCand(expose,VBN)
DepPath(make,"Kent cigarette filters",dobj)
DepPath(caused,"a high percentage of cancer deaths",dobj)
DepPath(caused,"A form of asbestos once used to make Kent cigarette filters",nsubj)
DepPath(reported,"researchers",nsubj)
```

Figure 6 Evidence file.

Figure 6 shows sample evidences captured from the input document where each evidence variable has a structure defined.

Weight Computation of Markov Logic Networks

Markov logic network (MLN) is comprised of weighted formula as mentioned previously. A system called Tuffy [46] which is a scalable Markov logic Network Inference Engine is used in this work.

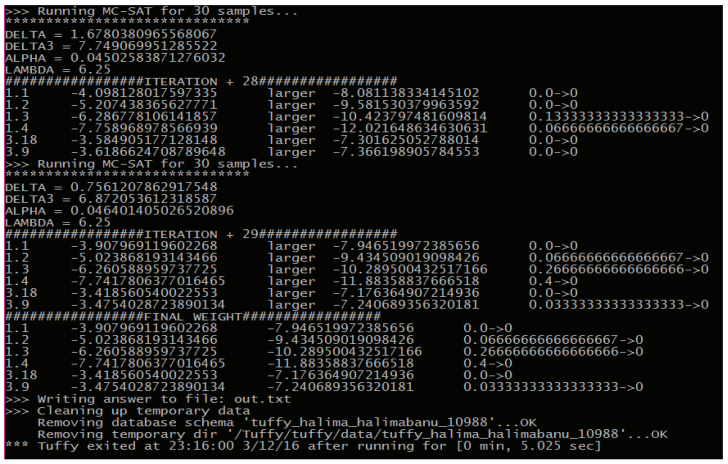
Weight learning takes a input evidence file and an MLN program without weight; it tries to compute the best weights of the local and global MLN rules by maximizing the likelihood of the evidence file. Tuffy implements the Diagonal Newton discriminative learner which uses inverse of the diagonalized Hessian as described by Lowd et al. [31]. If weights are fed while learning, Tuffy would ignore and continue the process of finding the optimal weights.

Figure 7 shows sample weight values during the execution set of iterations. These weights influences the final probabilities of the various predicate and argument labelling.

As mentioned earlier local formula are considered as observed predicates and global formula are considered as hidden predicates.

Observed Predicates

Following are the observed predicates which form the local formula of Markov Logic Network.



```
>>> Running MC-SAT for 30 samples...
******************************
DELTA = 1.6780380965568067
DELTA3 = 7.749069951285522
ALPHA = 0.04502583871276032
LAMBDA = 6.25
#################ITERATION + 28################
1.1     -4.098128017597335      larger  -8.081138334145102      0.0->0
1.2     -5.207438365627771      larger  -9.581530379963592      0.0->0
1.3     -6.286778106141857      larger  -10.423797481609814     0.13333333333333333->0
1.4     -7.758968978566939      larger  -12.021648634630631     0.06666666666666667->0
3.18    -3.584905177128148      larger  -7.301625052788014      0.0->0
3.9     -3.6186624708789648     larger  -7.366198905784553      0.0->0
>>> Running MC-SAT for 30 samples...
******************************
DELTA = 0.7561207862917548
DELTA3 = 6.872053612318587
ALPHA = 0.046401405026520896
LAMBDA = 6.25
#################ITERATION + 29################
1.1     -3.907969119602268      larger  -7.946519972385656      0.0->0
1.2     -5.023868193143466      larger  -9.434509019098426      0.06666666666666667->0
1.3     -6.260588959737725      larger  -10.289500432517166     0.26666666666666666->0
1.4     -7.7417806377016465     larger  -11.88358837666518      0.4->0
3.18    -3.418560540022553      larger  -7.176364907214936      0.0->0
3.9     -3.4754028723890134     larger  -7.240689356320181      0.03333333333333333->0
#################FINAL WEIGHT################
1.1     -3.907969119602268      -7.946519972385656      0.0->0
1.2     -5.023868193143466      -9.434509019098426      0.06666666666666667->0
1.3     -6.260588959737725      -10.289500432517166     0.26666666666666666->0
1.4     -7.7417806377016465     -11.88358837666518      0.4->0
3.18    -3.418560540022553      -7.176364907214936      0.0->0
3.9     -3.4754028723890134     -7.240689356320181      0.03333333333333333->0
>>> Writing answer to file: out.txt
>>> Cleaning up temporary data
    Removing database schema 'tuffy_halima_halimabanu_10988'...OK
    Removing temporary dir '/Tuffy/tuffy/data/tuffy_halima_halimabanu_10988'...OK
*** Tuffy exited at 23:16:00 3/12/16 after running for [0 min, 5.025 sec]
```

Figure 7 Weights of markov logic network.

- Predicate in Equation (1) states the rule with lemma of the sentence and it simplied predicate is has Role(p, a) which indicates that the token at position "*a*" is an argument of the predicate at position "*p*".

$$lemma(p, +l_1)\Lambda lemma(a, +l_2) \implies has\ Role(p, a) \tag{1}$$

- Predicate in Equation (2) states the rule with position of the sentence in which *role(p,a,r)* indicates that the token "*a*" plays a semantic role "*r*" with respect to the token "*p*".

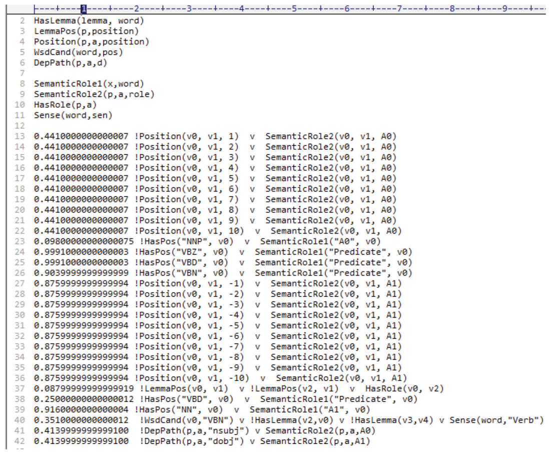
$$position(p, a, +p_0) \implies role(p, a, +r) \tag{2}$$

- Predicate in Equation (3) states the rule with dependency path of the sentence having the same predicaterole *(p,a, +r).*

$$depPath(p, a, +d) \implies role(p, a, +r) \tag{3}$$

- Predicate in Equations (4) and (5) states the rule with word sense candidate of the sentence in which wsdCand *(w, +* t_w*)* states that current word "*w*" has the POS tag "t_w".

$$wsdCand(w, +t_w)\Lambda\ lemma(w, +l_w)\Lambda\ lemma(1, +l_1) \\ \implies sense(w, +s) \tag{4}$$



```
HasLemma(lemma, word)
LemmaPos(p,position)
Position(p,a,position)
WsdCand(word,pos)
DepPath(p,a,d)

SemanticRole1(x,word)
SemanticRole2(p,a,role)
HasRole(p,a)
Sense(word,sen)

0.4410000000000007 !Position(v0, v1, 1)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 2)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 3)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 4)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 5)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 6)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 7)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 8)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 9)  v  SemanticRole2(v0, v1, A0)
0.4410000000000007 !Position(v0, v1, 10)  v  SemanticRole2(v0, v1, A0)
0.09800000000000075 !HasPos("NNP", v0)  v  SemanticRole1("A0", v0)
0.9991000000000003 !HasPos("VBZ", v0)  v  SemanticRole1("Predicate", v0)
0.9991000000000003 !HasPos("VBD", v0)  v  SemanticRole1("Predicate", v0)
0.9039999999999999 !HasPos("VBN", v0)  v  SemanticRole1("Predicate", v0)
0.8759999999999994 !Position(v0, v1, -1)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -2)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -3)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -4)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -5)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -6)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -7)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -8)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -9)  v  SemanticRole2(v0, v1, A1)
0.8759999999999994 !Position(v0, v1, -10)  v  SemanticRole2(v0, v1, A1)
0.08799999999999919 !LemmaPos(v0, v1)  v !LemmaPos(v2, v1)  v  HasRole(v0, v2)
0.25000000000000012 !HasPos("VBD", v0)  v  SemanticRole1("Predicate", v0)
0.9160000000000004 !HasPos("NN", v0)  v  SemanticRole1("A1", v0)
0.3510000000000012  !WsdCand(v0,"VBN") v !HasLemma(v2,v0) v !HasLemma(v3,v4) v Sense(word,"Verb")
0.4139999999999100  !DepPath(p,a,"nsubj") v SemanticRole2(p,a,A0)
0.4139999999999100  !DepPath(p,a,"dobj") v SemanticRole2(p,a,A1)
```

Figure 8 Local markov logic network.

$$
\begin{aligned}
&wsdCand(w, +t_w)\Lambda\ lemma(w, +l_w)\Lambda\ lemma(1, +l_n)\\
&\quad\implies sense(w, +s)
\end{aligned}
\tag{5}
$$

Figure 8 shows the observed predicates along with the weights assigned to each predicate which is collectively termed as Local Markov Logic Network.

Hidden Predicates

Predicate in Equations (6) to (10) states the rule with argument, predicate, sense and role.

$$role(p, a, r_1)\Lambda r_1 \neq r_2 \implies \neg role(p, a, r_2) \tag{6}$$

$$sense(p, s_1)\Lambda s_1 \neq S_2 \implies \neg sense(p, r_2) \tag{7}$$

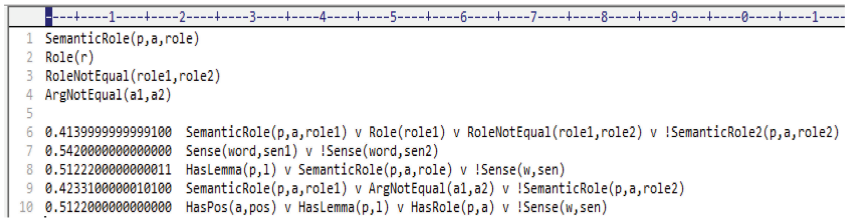
$$role(p, a_1, r)\Lambda\neg mod(r)\Lambda a_1 \neq a_2 \implies \neg role(p, a_2, r) \tag{8}$$

$$lemma(p, +l)\Lambda ppos(a, +p)\Lambda hasRole(p, a) \implies sense(p, +f) \tag{9}$$

$$lemma(p, +l)\Lambda role(p, a, +r) \implies sense(p, +s) \tag{10}$$

Figure 9 shows the hidden predicates along with the weights assigned to each predicate which is collectively termed as Global Markov Logic Network.

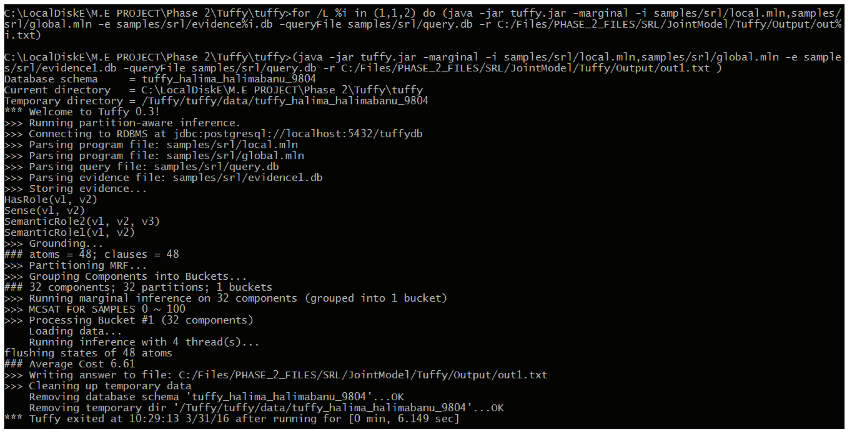
Figure 10 shows the execution of Tuffy system which grounds the atoms, partitions the data, infers and learns followed by publishing the probabilities of each predicate.



```
SemanticRole(p,a,role)
Role(r)
RoleNotEqual(role1,role2)
ArgNotEqual(a1,a2)

0.4139999999999100  SemanticRole(p,a,role1) v Role(role1) v RoleNotEqual(role1,role2) v !SemanticRole2(p,a,role2)
0.5420000000000000  Sense(word,sen1) v !Sense(word,sen2)
0.5122200000000011  HasLemma(p,l) v SemanticRole(p,a,role) v !Sense(w,sen)
0.4233100000010100  SemanticRole(p,a,role1) v ArgNotEqual(a1,a2) v !SemanticRole(p,a,role2)
0.5122000000000000  HasPos(a,pos) v HasLemma(p,l) v HasRole(p,a) v !Sense(w,sen)
```

Figure 9 Global markov logic network.



```
C:\LocalDiskE\M.E PROJECT\Phase 2\Tuffy\tuffy>for /L %i in (1,1,2) do (java -jar tuffy.jar -marginal -i samples/srl/local.mln,samples/
srl/global.mln -e samples/srl/evidence%i.db -queryFile samples/srl/query.db -r C:/Files/PHASE_2_FILES/SRL/JointModel/Tuffy/Output/out%
i.txt)

C:\LocalDiskE\M.E PROJECT\Phase 2\Tuffy\tuffy>(java -jar tuffy.jar -marginal -i samples/srl/local.mln,samples/srl/global.mln -e sample
s/srl/evidence1.db -queryFile samples/srl/query.db -r C:/Files/PHASE_2_FILES/SRL/JointModel/Tuffy/Output/out1.txt )
Database schema     = tuffy_halima_halimabanu_9804
Current directory   = C:\LocalDiskE\M.E PROJECT\Phase 2\Tuffy\tuffy
Temporary directory = /Tuffy/tuffy/data/tuffy_halima_halimabanu_9804
*** Welcome to Tuffy 0.3!
>>> Running partition-aware inference.
>>> Connecting to RDBMS at jdbc:postgresql://localhost:5432/tuffydb
>>> Parsing program file: samples/srl/local.mln
>>> Parsing program file: samples/srl/global.mln
>>> Parsing query file: samples/srl/query.db
>>> Parsing evidence file: samples/srl/evidence1.db
>>> Storing evidence...
HasRole(v1, v2)
Sense(v1, v2)
SemanticRole2(v1, v2, v3)
SemanticRole1(v1, v2)
>>> Grounding...
### atoms = 48; clauses = 48
>>> Partitioning MRF...
>>> Grouping Components into Buckets...
### 32 components; 32 partitions; 1 buckets
>>> Running marginal inference on 32 components (grouped into 1 bucket)
>>> MCSAT FOR SAMPLES 0 ~ 100
>>> Processing Bucket #1 (32 components)
    Loading data...
    Running inference with 4 thread(s)...
flushing states of 48 atoms
### Average Cost 6.61
>>> Writing answer to file: C:/Files/PHASE_2_FILES/SRL/JointModel/Tuffy/Output/out1.txt
>>> Cleaning up temporary data
    Removing database schema 'tuffy_halima_halimabanu_9804'...OK
    Removing temporary dir '/Tuffy/tuffy/data/tuffy_halima_halimabanu_9804'...OK
*** Tuffy exited at 10:29:13 3/31/16 after running for [0 min, 6.149 sec]
```

Figure 10 Joint semantic role labelling.

3.3 Processing of Predicate Argument Structures

Once the predicate argument structures are obtained, second level of processing has to be applied. This process involves splitting the PAS, removal of stop words and extracting the root word of the token using porter stemming algorithm [40]. POS tagger [8] is used for retrieve part of speech. Named entity recognition details, if applicable, is extracted for each token. Figure 11 shows the processing of PAS.

Algorithm 3 below discuss about the processing of PAS and results of processing is stored as XML file.

Figure 12 displays the XML file of the processed predicate argument structures each labelled with PAS ID.

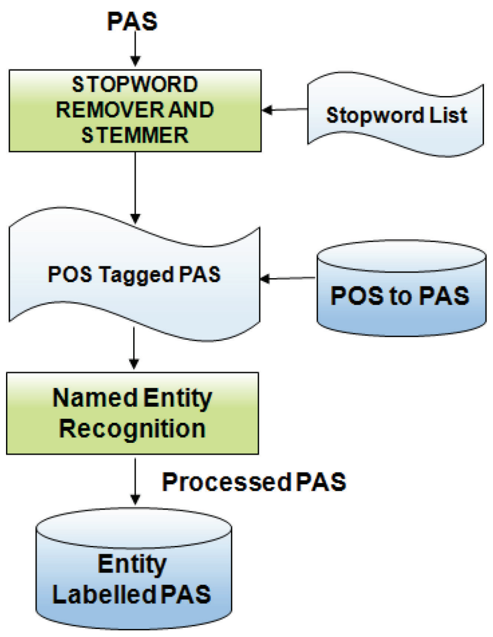
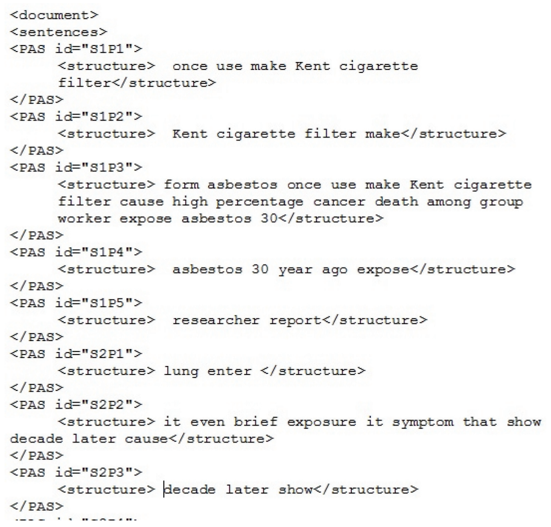


Figure 11 Processing of PAS.

Algorithm 3: Algorithm for processing of Predicate Argument Structures

```
Input: Predicate Argument Structures (PAS)
Output: Processed Predicate Argument Structures
begin
    for each PAS do
        remove StopWords(PAS[i])
    end
    for each PAS do
        porterStemming(PAS)
    end
    retrieve parts of speech tagged PAS
    for each PAS do
        entityLabel $\leftarrow$ CRFClassifier(PAS[i])
    end
end
```



```
<document>
<sentences>
<PAS id="S1P1">
     <structure>  once use make Kent cigarette
     filter</structure>
</PAS>
<PAS id="S1P2">
     <structure>  Kent cigarette filter make</structure>
</PAS>
<PAS id="S1P3">
     <structure> form asbestos once use make Kent cigarette
     filter cause high percentage cancer death among group
     worker expose asbestos 30</structure>
</PAS>
<PAS id="S1P4">
     <structure>  asbestos 30 year ago expose</structure>
</PAS>
<PAS id="S1P5">
     <structure>  researcher report</structure>
</PAS>
<PAS id="S2P1">
     <structure> lung enter </structure>
</PAS>
<PAS id="S2P2">
     <structure> it even brief exposure it symptom that show
decade later cause</structure>
</PAS>
<PAS id="S2P3">
     <structure> decade later show</structure>
</PAS>
```

Figure 12 Processed PAS XML file.

3.4 Semantic Based Content Selection

3.4.1 Semantic similarity matrix computation

As in [21] semantic similarity matrix computation module aims at building a matrix of semantic similarity scores for each pair of predicate argument structure. Jiang's measure is used to compute the semantic similarities [18]. Figure 13 shows the process flow of semantic similarity matrix computation phase.

Equation (11) below is the Jiang Similarity measure which finds the similarity between two concepts.
Jiang's Similarity Measure:

$$Jiang_{dist}(Cl, C2) = IC(C1) + IC(C2) - 2IC(Iso(C1, C2)) \qquad (11)$$

where,

$$IC(C) = -\log P(C)$$
$$P(C) = Freq(C)/N$$

Jiang's measure uses WordNet to compute the least common subsequence (lso) then determines the information content IC. Information content is computed by retrieving the probability of concept occurrence in the text corpus and quantified as in Equation (11).

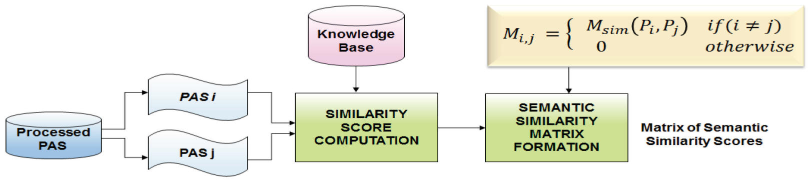

Knowledge Base
$M_{i,j} = \begin{cases} M_{sim}(P_i, P_j) & if (i \neq j) \\ 0 & otherwise \end{cases}$
Processed PAS
PAS i
PAS j
SIMILARITY SCORE COMPUTATION
SEMANTIC SIMILARITY MATRIX FORMATION
Matrix of Semantic Similarity Scores

Figure 13 Semantic similarity matrix computation.

Algorithm 4: Jiang's Measure for relatedness of PAS

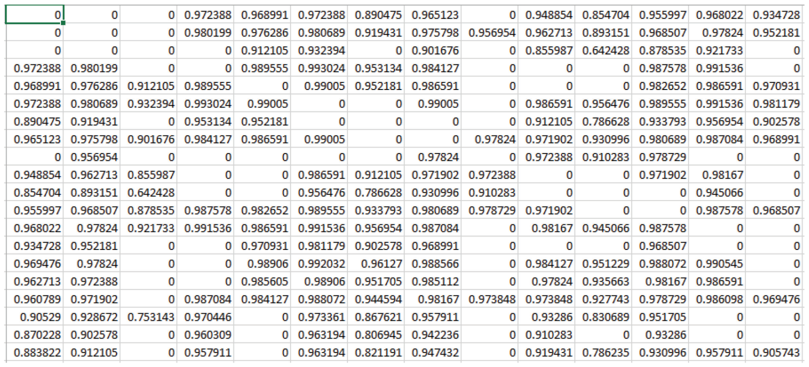
```
Input: Processed PAS
Output: Matrix of Semantic Similar Scores
begin
    foreach PAS k of sentence $s_i$ do
        foreach PAS l of sentence $s_j$ do
            compute Jiang similarity score using Equation (12)
            compute predicates using Equation (13)
            Find temporal argument using Equation (14) and Equation (15)
            calculate final score using Equation (16)
        end
    end
end
```

Algorithm 4 above describes the steps to be followed to compute and build semantic similarity matrix.

Equation (12) is the similarity score between arguments, Equation (13) is the similarity score between predicates, Equations (14) and (15) is the similarity score between temporal and location and finally Equation (16) is the final similarity score.

$$\begin{aligned} sim_{arg}(v_{ik}, v_{jl}) &= sim(A0_i, A0_j) + sim(A1_i, A1_j) \\ &\quad + sim(A2_i, A2_j) \qquad (12) \\ sim_p(v_{ik}, v_{jl}) &= sim(P_i, P_j) \qquad (13) \\ sim_{tmp}(v_{ik}, v_{jl}) &= sim(Tmp_i, Tmp_j) \qquad (14) \\ sim_{loc}(v_{ik}, v_{jl}) &= sim(Loc_i, Loc_j) \qquad (15) \\ sim(v_{ik}, v_{jl}) &= sim_p(v_{ik}, v_{jl}) + sim_{arg}(v_{ik}, v_{jl}) \\ &\quad + sim_{loc}(v_{ik}, v_{jl}) + sim_{tmp}(v_{ik}, v_{jl}) \qquad (16) \end{aligned}$$

Figure 14 below displays the PAS semantic similarity matrix which is result of normalized results obtained by a scaling factor of e$^{-asim(vik,vjl)}$ as in [1].



0	0	0	0.972388	0.968991	0.972388	0.890475	0.965123	0	0.948854	0.854704	0.955997	0.968022	0.934728
0	0	0	0.980199	0.976286	0.980689	0.919431	0.975798	0.956954	0.962713	0.893151	0.968507	0.97824	0.952181
0	0	0	0	0.912105	0.932394	0	0.901676	0	0.855987	0.642428	0.878535	0.921733	0
0.972388	0.980199	0	0	0.989555	0.993024	0.953134	0.984127	0	0	0	0.987578	0.991536	0
0.968991	0.976286	0.912105	0.989555	0	0.99005	0.952181	0.986591	0	0	0	0.982652	0.986591	0.970931
0.972388	0.980689	0.932394	0.993024	0.99005	0	0	0.99005	0	0.986591	0.956476	0.989555	0.991536	0.981179
0.890475	0.919431	0	0.953134	0.952181	0	0	0	0	0.912105	0.786628	0.933793	0.956954	0.902578
0.965123	0.975798	0.901676	0.984127	0.986591	0.99005	0	0	0.97824	0.971902	0.930996	0.980689	0.987084	0.968991
0	0.956954	0	0	0	0	0	0.97824	0	0.972388	0.910283	0.978729	0	0
0.948854	0.962713	0.855987	0	0	0.986591	0.912105	0.971902	0.972388	0	0	0.971902	0.98167	0
0.854704	0.893151	0.642428	0	0	0.956476	0.786628	0.930996	0.910283	0	0	0	0.945066	0
0.955997	0.968507	0.878535	0.987578	0.982652	0.989555	0.933793	0.980689	0.978729	0.971902	0	0	0.987578	0.968507
0.968022	0.97824	0.921733	0.991536	0.986591	0.991536	0.956954	0.987084	0	0.98167	0.945066	0.987578	0	0
0.934728	0.952181	0	0	0.970931	0.981179	0.902578	0.968991	0	0	0	0.968507	0	0
0.969476	0.97824	0	0	0.98906	0.992032	0.96127	0.988566	0	0.984127	0.951229	0.988072	0.990545	0
0.962713	0.972388	0	0	0.985605	0.98906	0.951705	0.985112	0	0.97824	0.935663	0.98167	0.986591	0
0.960789	0.971902	0	0.987084	0.984127	0.988072	0.944594	0.98167	0.973848	0.973848	0.927743	0.978729	0.986098	0.969476
0.90529	0.928672	0.753143	0.970446	0	0.973361	0.867621	0.957911	0	0.93286	0.830689	0.951705	0	0
0.870228	0.902578	0	0.960309	0	0.963194	0.806945	0.942236	0	0.910283	0	0.93286	0	0
0.883822	0.912105	0	0.957911	0	0.963194	0.821191	0.947432	0	0.919431	0.786235	0.930996	0.957911	0.905743

Figure 14 PAS semantic similarity matrix.

3.4.2 Clustering semantically similar PAS

Agglomerative hierarchical clustering is well-known technique in the hierarchical clustering techniques. Of the several linkage methods available in clustering algorithms, it was found that the average linkage method is the best for document clustering. Therefore this work also uses average linkage method to find the similarity between the old cluster and newly formed cluster. This module takes the semantic similarity matrix as input in which the value at position (i,j) is the similarity score between ith and jth predicate argument structures. Since in hierarchical clustering each element is assumed to be a cluster initially, value at position (i,j) would be considered as a single cluster.

Figure 15 depicts the process flow of Agglomerative clustering in which the compression rate mentioned is the limiting point specified to cut the dendrogram.

Algorithm 5 below details about the agglomerative clustering of predicate argument structures using a compression rate of 20%. Average Linkage measure is given in Equation (17).

$$D(c_1, c_2) = \frac{1}{c_1}\frac{1}{c_2}\sum_{x_1 \epsilon c_1}\sum_{x_2 \epsilon c_2} D(x_1, x_2) \tag{17}$$

where, $C_1,\ C_2$ – Clusters, X_1 – New cluster, X_2 – Original Cluster

Figure 16 shows the dendrogram details along with its distance scores. Cluster numbers in figure below is the PAS number that has to be merged into clusters.

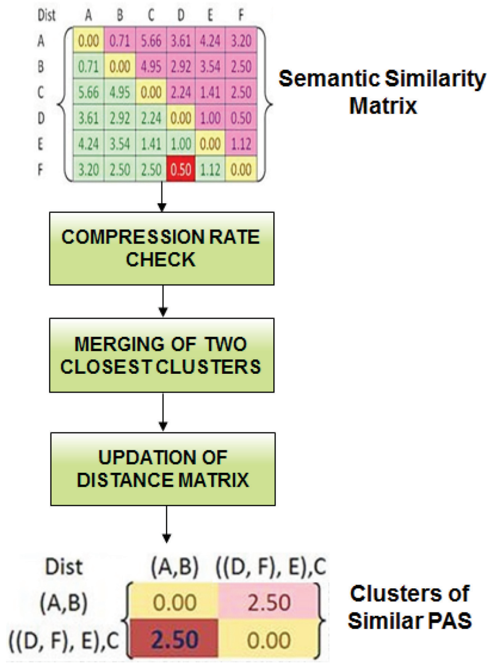


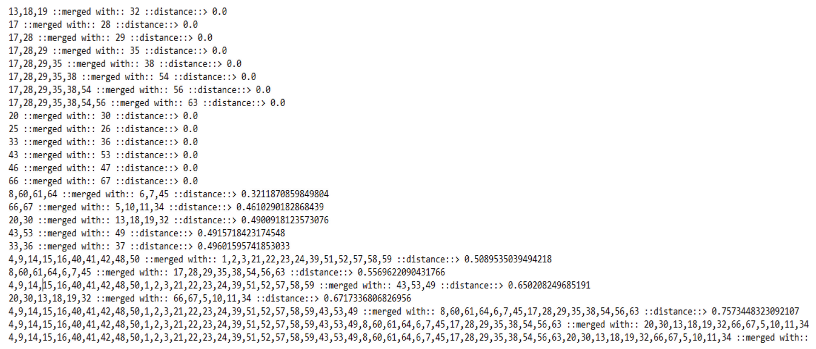
Figure 15 Agglomerative clustering for PAS.

Algorithm 5: Agglomerative Clustering of PAS

```
  Input: Semantically Similar Matrix (SSM)
  Output: Clusters of Similar PAS
1 Initialize:
2 $C_1$, $C_2$ are represented as clusters
3 begin
4     merge $C_1$, $C_2$
5     update SSM calculation of step 4
6     compute similarity between $X_1$ and $X_2$ using Equation (17)
7     continue until achieve compression rate of summary
8 end
```

3.4.3 PAS based feature extraction

PAS based feature extraction aims at extracting ten distinguishing features from PAS and input document. This forms a vector of size ten for each PAS and the final score will be based on these features. Algorithm 6 below details about the feature extraction process.



```
13,18,19 ::merged with:: 32 ::distance::> 0.0
17 ::merged with:: 28 ::distance::> 0.0
17,28 ::merged with:: 29 ::distance::> 0.0
17,28,29 ::merged with:: 35 ::distance::> 0.0
17,28,29,35 ::merged with:: 38 ::distance::> 0.0
17,28,29,35,38 ::merged with:: 54 ::distance::> 0.0
17,28,29,35,38,54 ::merged with:: 56 ::distance::> 0.0
17,28,29,35,38,54,56 ::merged with:: 63 ::distance::> 0.0
20 ::merged with:: 30 ::distance::> 0.0
25 ::merged with:: 26 ::distance::> 0.0
33 ::merged with:: 36 ::distance::> 0.0
43 ::merged with:: 53 ::distance::> 0.0
46 ::merged with:: 47 ::distance::> 0.0
66 ::merged with:: 67 ::distance::> 0.0
8,60,61,64 ::merged with:: 6,7,45 ::distance::> 0.3211870859849804
66,67 ::merged with:: 5,10,11,34 ::distance::> 0.4610290182868439
20,30 ::merged with:: 13,18,19,32 ::distance::> 0.4900918123573076
43,53 ::merged with:: 49 ::distance::> 0.4915718423174548
33,36 ::merged with:: 37 ::distance::> 0.49601595741853033
4,9,14,15,16,40,41,42,48,50 ::merged with:: 1,2,3,21,22,23,24,39,51,52,57,58,59 ::distance::> 0.5089535039494218
8,60,61,64,6,7,45 ::merged with:: 17,28,29,35,38,54,56,63 ::distance::> 0.5569622090431766
4,9,14,15,16,40,41,42,48,50,1,2,3,21,22,23,24,39,51,52,57,58,59 ::merged with:: 43,53,49 ::distance::> 0.650208249685191
20,30,13,18,19,32 ::merged with:: 66,67,5,10,11,34 ::distance::> 0.6717336806826956
4,9,14,15,16,40,41,42,48,50,1,2,3,21,22,23,24,39,51,52,57,58,59,43,53,49 ::merged with:: 8,60,61,64,6,7,45,17,28,29,35,38,54,56,63 ::distance::> 0.7573448323092107
4,9,14,15,16,40,41,42,48,50,1,2,3,21,22,23,24,39,51,52,57,58,59,43,53,49,8,60,61,64,6,7,45,17,28,29,35,38,54,56,63 ::merged with:: 20,30,13,18,19,32,66,67,5,10,11,34
4,9,14,15,16,40,41,42,48,50,1,2,3,21,22,23,24,39,51,52,57,58,59,43,53,49,8,60,61,64,6,7,45,17,28,29,35,38,54,56,63,20,30,13,18,19,32,66,67,5,10,11,34 ::merged with::
```

Figure 16 Hierarchical clustering of PAS similarity matrix.

Algorithm 6: PAS based Feature Extraction

```
Input: Clusters of Similar PAS and input document
Output: P for each PAS
1 begin
2     foreach $c_i \in c$ do
3         foreach $PAS\ k\ in\ c_i$ do
4             extract $F_i$, where i=1,2,...,n
5             build feature vector P $= (P_{Fi})$ using Equation (18 to 27)
6         end
7     end
8 end
```

Equation (18) is the title feature which is determined by counting the number of matched words in both the predicate argument structures and the document title.

$$P_{F1} = \frac{\text{No: of title words in PAS}}{\text{No: of words in document title}} \tag{18}$$

Equation (19) is the Length of the predicate argument structure in which the normalized length of PAS is calculated.

$$P_{F2} = \frac{\text{No: of words occuring in PAS}}{\text{No: of words occuring in longest PAS}} \tag{19}$$

Equation (20) is the PAS-PAS similarity in which the similarity score for each pair of PAS is obtained and then ratio of the sum of similarities with all PAS over maximum similarity is calculated.

$$P_{F3} = \frac{\sum sim(P_i, P_j)}{Max \sum sim(Pi, Pj)} \tag{20}$$

Equation (21) computes the position of predicate argument structure indicating its importance.

$$P_{F4} = \frac{\mathrm{Len(doc)} - \mathrm{pos(PAS)} + 1}{\mathrm{Len(doc)}} \tag{21}$$

Equation (22) computes the proper nouns since more number of proper nouns is considered significant for inclusion in summary generation.

$$P_{F5} = \frac{\mathrm{No\colon\ of\ proper\ nouns\ in\ PAS}}{\mathrm{Len(PAS)}} \tag{22}$$

Equation (23) extracts the numerical data such as number of people killed etc. for inclusion in summary generation.

$$P_{F6} = \frac{\mathrm{No\colon\ of\ numerical\ data\ in\ PAS}}{\mathrm{Len(PAS)}} \tag{23}$$

Equation (24) extracts the nouns and verbs in the predicate argument structures.

$$P_{F7} = \frac{\mathrm{Total\ no\colon\ of\ nouns\ verbs\ in\ PAS}}{\mathrm{Len(PAS)}} \tag{24}$$

Equation (25) extracts the temporal feature in PAS.

$$P_{F8} = \frac{\mathrm{No: \ of\ temporal\ info\ in\ PAS}}{\mathrm{Len(PAS)}} \tag{25}$$

Equation (26) extracts the frequent semantic term to learn the terms that reflect the topic of the document.

$$P_{F9} = \frac{\mathrm{No: \ of\ freq\ semantic\ terms\ in\ PAS}}{\mathrm{Max(No: \ of\ freq\ semantic\ terms\ in\ PAS)}} \tag{26}$$

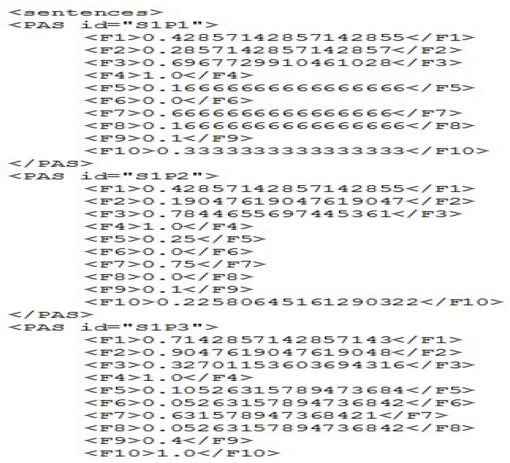
Equation (27) extracts semantic term weights obtained via term frequency.

$$P_{F10} = \frac{\sum_{k}^{i=1} W_i(P)}{Max \sum_{k}^{i=1} W_i(P)} \tag{27}$$

where,

$$W_i = Tf_i \times log\frac{N}{n_i}$$

Figure 17 displays the XML file of ten feature values indexed with PAS ID. These feature values will be used to compute the final PAS score after the execution of optimal feature weighting module.



```
<sentences>
<PAS id="S1P1">
        <F1>0.42857142857142855</F1>
        <F2>0.2857142857142857</F2>
        <F3>0.6967729910461028</F3>
        <F4>1.0</F4>
        <F5>0.16666666666666666</F5>
        <F6>0.0</F6>
        <F7>0.6666666666666666</F7>
        <F8>0.16666666666666666</F8>
        <F9>0.1</F9>
        <F10>0.3333333333333333</F10>
</PAS>
<PAS id="S1P2">
        <F1>0.42857142857142855</F1>
        <F2>0.19047619047619047</F2>
        <F3>0.7844655697445361</F3>
        <F4>1.0</F4>
        <F5>0.25</F5>
        <F6>0.0</F6>
        <F7>0.75</F7>
        <F8>0.0</F8>
        <F9>0.1</F9>
        <F10>0.22580645161290322</F10>
</PAS>
<PAS id="S1P3">
        <F1>0.7142857142857143</F1>
        <F2>0.9047619047619048</F2>
        <F3>0.32701153603694316</F3>
        <F4>1.0</F4>
        <F5>0.10526315789473684</F5>
        <F6>0.05263157894736842</F6>
        <F7>0.631578947368421</F7>
        <F8>0.05263157894736842</F8>
        <F9>0.4</F9>
        <F10>1.0</F10>
```

Figure 17 PAS based features.

3.4.4 Optimal feature weighting

In text summarization the quality of summaries are sensitive to text features. But, not all features have the same significance so there is a need to assign weights for each feature reflecting its importance.

Figure 18 shows the process flow of optimal feature weighting whose output gives the final PAS score.

Algorithm 7 below describes the steps for optimal feature weighting using Genetic Algorithm. The fitness value for each individual is

$$F(x) = \frac{\sum_{S\epsilon}\{RefSummaries\}gram_n\epsilon S^{\sum^{count(m-gram)}}}{\sum_{S\epsilon}\{RefSummaries\}gram_n\epsilon S^{\sum^{count(gram)}}} \tag{28}$$

According to genetic algorithm first step is to create the initial population which are randomly selected unique individuals. The values chosen fall within the range of 0 to 1. Each individual represents the weights of features in form of (w1,w2,w3,. . . ,w10). Using the fitness score individuals would be selected as target individuals. Figure 19 below shows the values of target population.

Mutation operator to build the donor individuals are computed using Equation (29) as follows:

Mutation operator

$$\text{donor individual} = \text{target}_{rand\ 1} + F\{\text{target}_{rand\ 2} - \text{target}_{rand\ 3}\} \tag{29}$$

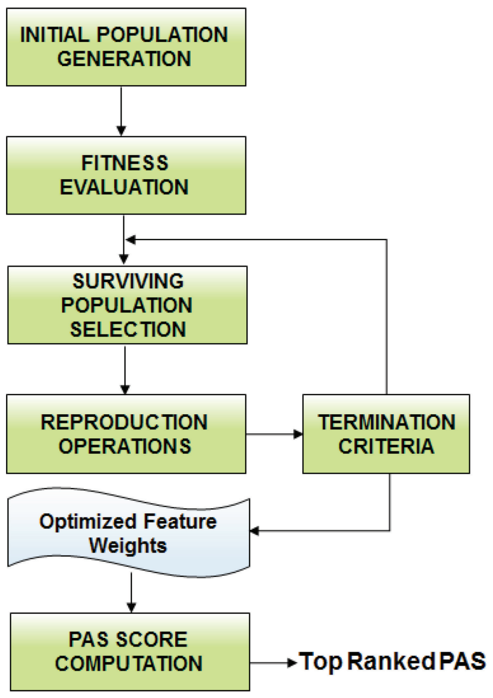
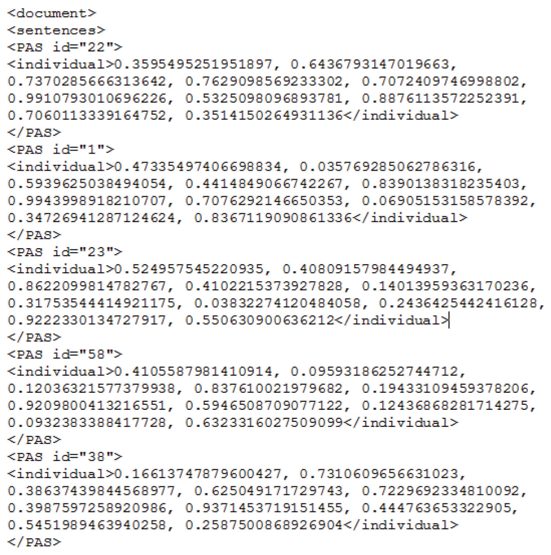


Figure 18 Optimal feature weighting.

Algorithm 7: Genetic Algorithm for Optimal Feature Weighting

```
Input: Clusters of PAS
Output: Optimal Feature Weighting
1 begin
2     generate random individual weights of features $w_j$, where j=1,2,. . . ,10
3     compute the fitness value F(x) using Equation (28)
4     perform cross over and mutation operations using Equation (29) and
        Equation (30)
5     continue until achieve termination criteria
6 end
```

Crossover operator to build the trial individuals are computed using Equation (30) and the constants such as Mutation scale factor and crossover constant are computed using Equation (31) and Equation (32) as follows:



```
<document>
<sentences>
<PAS id="22">
<individual>0.3595495251951897, 0.6436793147019663,
0.7370285666313642, 0.7629098569233302, 0.7072409746998802,
0.9910793010696226, 0.532509809689378l, 0.8876113572252391,
0.7060113339164752, 0.3514150264931136</individual>
</PAS>
<PAS id="1">
<individual>0.47335497406698834, 0.035769285062786316,
0.5939625038494054, 0.4414849066742267, 0.8390138318235403,
0.9943998918210707, 0.7076292146650353, 0.06905153158578392,
0.34726941287124624, 0.8367119090861336</individual>
</PAS>
<PAS id="23">
<individual>0.524957545220935, 0.40809157984494937,
0.8622099814782767, 0.4102215373927828, 0.14013959363170236,
0.31753544414921175, 0.03832274120484058, 0.2436425442416128,
0.9222330134727917, 0.550630900636212</individual>
</PAS>
<PAS id="58">
<individual>0.4105587981410914, 0.09593186252744712,
0.12036321577379938, 0.837610021979682, 0.19433109459378206,
0.920980041321655l, 0.5946508709077122, 0.12436868281714275,
0.0932383388417728, 0.6323316027509099</individual>
</PAS>
<PAS id="38">
<individual>0.16613747879600427, 0.7310609656631023,
0.38637439844568977, 0.625049171729743, 0.7229692334810092,
0.3987597258920986, 0.9371453719151455, 0.444763653322905,
0.5451989463940258, 0.2587500868926904</individual>
</PAS>
```

Figure 19 Target population.

Crossover operator

$$\text{trial individual} = \begin{cases} \text{donor individual} & if\ rand(0,1) \\ & \leq -Corj == j_{rand} \\ \text{target individual} & \text{otherwise} \end{cases} \tag{30}$$

Mutation scale factor

$$F = \begin{cases} 0.1 + 0.9 \times\ rand(0,1) & \text{if } rand(0,1) < 0.1 \\ F & \text{otherwise} \end{cases} \tag{31}$$

Crossover constant

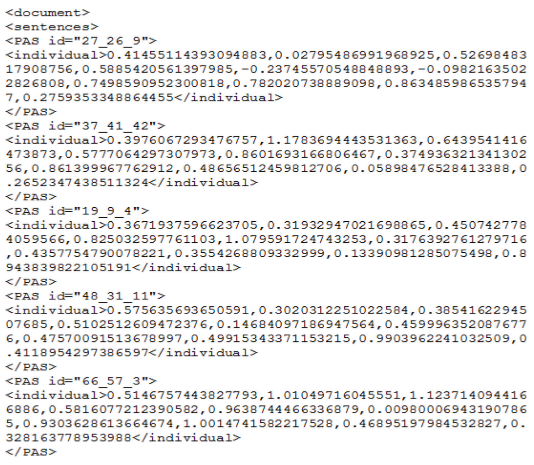
$$C = \begin{cases} rand(0.1) & \text{if } rand(0,1) \leq 0.1 \\ C & \text{otherwise} \end{cases} \tag{32}$$

Algorithm 8 below discuss the steps in computation of final score for each PAS. Compute the score for all the Predicate argument structures of each cluster using Equation (33).

$$Score(P_i) = \sum_{k=1}^{10} W_k \times P_{F_k}(P_i) \tag{33}$$

Algorithm 8: Computation of PAS score

```
Input: Optimal Feature Weighting
Output: Top Ranked PAS
1 begin
2     foreach $c_i$ in $c$ do
3         foreach PAS k in $c_i$ do
4             compute the score of PAS using Equation (33)
5             return T-Score ($P_i$)
6         end
7     end
8 end
```



```
<document>
<sentences>
<PAS id="27_26_9">
<individual>0.41455114393094883,0.02795486991968925,0.52698483
17908756,0.5885420561397985,-0.23745570548848893,-0.0982163502
2826808,0.7498590952300818,0.782020738889098,0.863485986535794
7,0.2759353348864455</individual>
</PAS>
<PAS id="37_41_42">
<individual>0.3976067293476757,1.1783694443531363,0.6439541416
473873,0.5777064297307973,0.8601693166806467,0.374936321341302
56,0.861399967762912,0.48656512459812706,0.05898476528413388,0
.2652347438511324</individual>
</PAS>
<PAS id="19_9_4">
<individual>0.3671937596623705,0.31932947021698865,0.450742778
4059566,0.825032597761103,1.079591724743253,0.3176392761279716
,0.4357754790078221,0.3554268809332999,0.13390981285075498,0.8
943839822105191</individual>
</PAS>
<PAS id="48_31_11">
<individual>0.575635693650591,0.3020312251022584,0.38541622945
07685,0.5102512609472376,0.14684097186947564,0.459996352087677
6,0.47570091513678997,0.49915343371153215,0.9903962241032509,0
.4118954297386597</individual>
</PAS>
<PAS id="66_57_3">
<individual>0.5146757443827793,1.01049716045551,1.123714094416
6886,0.5816077212390582,0.9638744466336879,0.00980006943190786
5,0.9303628613664674,1.0014741582217528,0.46895197984532827,0.
328163778953988</individual>
</PAS>
```

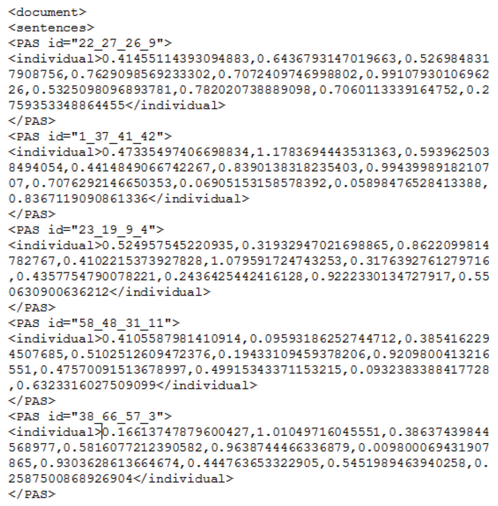
Figure 20 Donor population.

Figure 20 shows the values of donor individuals obtained via Mutation process.

Figure 21 shows the values of trial individuals obtained via Crossover process.

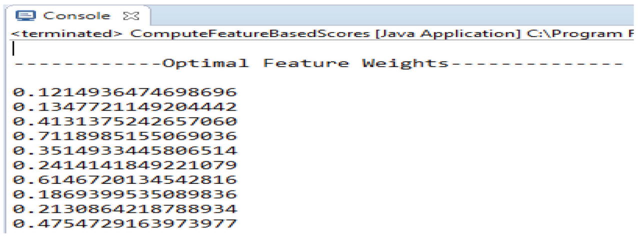
Figure 22 shows optimal feature weights obtained as a result of several iterative mutation and crossover process executions. Higher the feature value the more important that particular feature is to each PAS.

Figure 23 shows the final scores for each PAS which indicates its significance to the summary generation. Top scored PAS would be stored to be processed as source to the language generation module.



```
<document>
<sentences>
<PAS id="22_27_26_9">
<individual>0.4145511439309488З,0.6436793147019663,0.526984831
7908756,0.7629098569233302,0.7072409746998802,0.99107930106962
26,0.5325098096893781,0.782020738889098,0.7060113339164752,0.2
759353348864455</individual>
</PAS>
<PAS id="1_37_41_42">
<individual>0.47335497406698834,1.1783694443531363,0.593962503
8494054,0.441484906674226 7,0.8390138318235403,0.99439989182107
07,0.7076292146650353,0.06905153158578392,0.05898476528413388,
0.8367119090861336</individual>
</PAS>
<PAS id="23_19_9_4">
<individual>0.524957545220935,0.3193294702169886 5,0.8622099814
782767,0.4102215373927828,1.079591724743253,0.3176392761279716
,0.4357754790078221,0.2436425442416128,0.9222330134727917,0.55
0630900636212</individual>
</PAS>
<PAS id="58_48_31_11">
<individual>0.4105587981410914,0.09593186252744712,0.385416229
4507685,0.5102512609472376,0.19433109459378206,0.9209800413216
551,0.47570091513678997,0.49915343371153215,0.0932383388417728
,0.6323316027509099</individual>
</PAS>
<PAS id="38_66_57_3">
<individual>0.1661374787960042 7,1.01049716045551,0.38637439844
568977,0.5816077212390582,0.9638744466336879,0.009800069431907
865,0.9303628613664674,0.444763653322905,0.5451989463940258,0.
2587500868926904</individual>
</PAS>
```

Figure 21 Trial population.



```
Console ⌧
<terminated> ComputeFeatureBasedScores [Java Application] C:\Program F

------------Optimal Feature Weights--------------

0.1214936474698696
0.1347721149204442
0.4131375242657060
0.7118985155069036
0.3514933445806514
0.2414141849221079
0.6146720134542816
0.1869399535089836
0.2130864218788934
0.4754729163973977
```

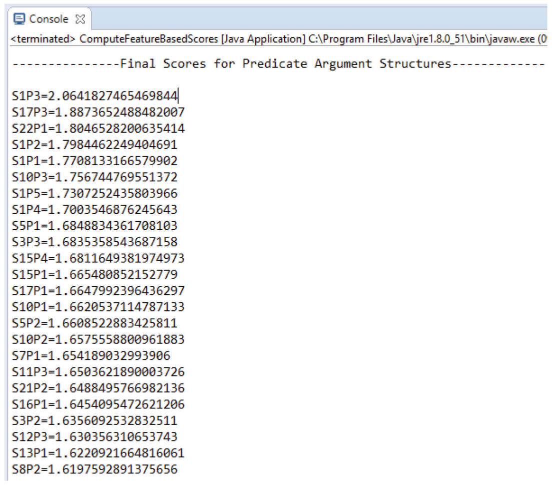
Figure 22 Optimal feature weights.

3.5 Target Language Generation

Language Generation module demonstrates how the argument and predicates are combined, transformed and realized as summary sentences. In this work SimpleNLG [29, 11] is used to generate summary sentences from predicate argument structures.

SimpleNLG is an English realization engine which provides interface to produce syntactical structures and then transform those into sentences using naive grammar rules. This engine has a good coverage of English morphology, syntax and it is also robust. Algorithm 9 below details about the steps to be followed to generate the target summary.

Figure 24 displays the final target semantic oriented abstractive summary of the input document which is an article about social news.



```
Console
<terminated> ComputeFeatureBasedScores [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (0
---------------Final Scores for Predicate Argument Structures-------------

S1P3=2.0641827465469844
S17P3=1.8873652488482007
S22P1=1.8046528200635414
S1P2=1.7984462249404691
S1P1=1.7708133166579902
S10P3=1.756744769551372
S1P5=1.7307252435803966
S1P4=1.7003546876245643
S5P1=1.6848834361708103
S3P3=1.6835358543687158
S15P4=1.6811649381974973
S15P1=1.665480852152779
S17P1=1.6647992396436297
S10P1=1.6620537114787133
S5P2=1.6608522883425811
S10P2=1.6575558800961883
S7P1=1.654189032993906
S11P3=1.6503621890003726
S21P2=1.6488495766982136
S16P1=1.6454095472621206
S3P2=1.6356092532832511
S12P3=1.630356310653743
S13P1=1.6220921664816061
S8P2=1.6197592891375656
```

Figure 23 Final PAS score.

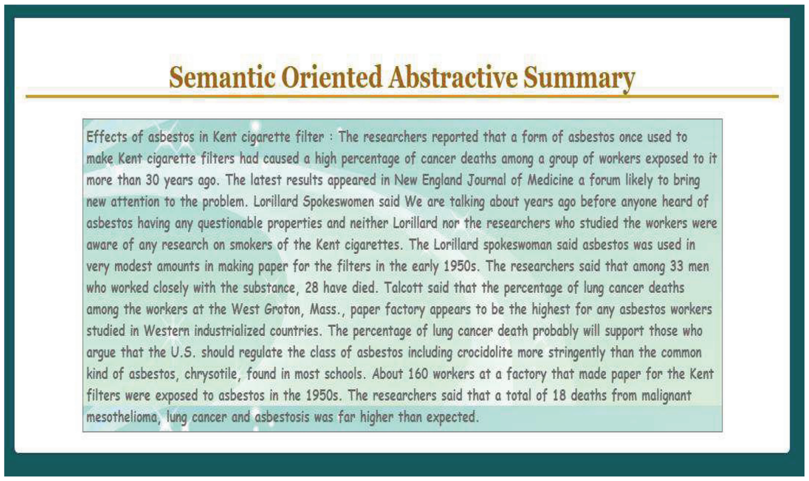
Algorithm 9: Language Generation using SimpleNLG Realiser

```
   Input: Top Ranked Predicate Argument Structures(PAS)
   Output: Summary Sentences formed from PAS
1  Initialize: constituent 1= lexical item, constituent 2= phrasal item
2  begin
3      Coherence Check
4      foreach PAS_ID do
5          re-arrange based on the position PAS
6          store the rearranged sentences as step 5
7      end
8  end
9  begin
10     SimpleNLG Realiser
11     compute feature set 1 from constituent 1
12     compute feature set 2 from constituent 2
13     combine step 11 and step 12
14     pass step 13 lineariser
15     summaryString[] ← realized string
16 end
```

4 Experimental Evaluation and Discussion

4.1 Dataset

Benchmark dataset used for the entire proposed system i.e. for evaluating the summarization performance is DUC 2002 [10].


Semantic Oriented Abstractive Summary

Effects of asbestos in Kent cigarette filter : The researchers reported that a form of asbestos once used to make Kent cigarette filters had caused a high percentage of cancer deaths among a group of workers exposed to it more than 30 years ago. The latest results appeared in New England Journal of Medicine a forum likely to bring new attention to the problem. Lorillard Spokeswomen said We are talking about years ago before anyone heard of asbestos having any questionable properties and neither Lorillard nor the researchers who studied the workers were aware of any research on smokers of the Kent cigarettes. The Lorillard spokeswoman said asbestos was used in very modest amounts in making paper for the filters in the early 1950s. The researchers said that among 33 men who worked closely with the substance, 28 have died. Talcott said that the percentage of lung cancer deaths among the workers at the West Groton, Mass., paper factory appears to be the highest for any asbestos workers studied in Western industrialized countries. The percentage of lung cancer death probably will support those who argue that the U.S. should regulate the class of asbestos including crocidolite more stringently than the common kind of asbestos, chrysotile, found in most schools. About 160 workers at a factory that made paper for the Kent filters were exposed to asbestos in the 1950s. The researchers said that a total of 18 deaths from malignant mesothelioma, lung cancer and asbestosis was far higher than expected.

Figure 24 Semantic oriented abstractive summary.

4.2 Experimental Setup

For the purpose of experimental setup, our work is implemented in Java language using Eclipse IDE (Luna 64bit). Intermediate outputs are stored partially in database and partially as files.

Tuffy 0.3 [46] system is used as the Markov Engine in this work. PostgreSQL 8.4 is the database that supports the functioning of Tuffy system. Apart from the standalone system, primary storage structure followed in this system is XML file format.

4.3 Test Cases

The test cases identified so far are tabulated below along with the expected output in Table 1.

4.4 Evaluation Metrics

Evaluation of the above work is Performance Evaluation of Abstractive Summarization. Intrinsic Content Evaluation of abstractive summarization is examined to focus Co-selection measures and Content based measures. Pyramid Score, Average Precision and Longest Common Subsequence measures the Abstractive Summarization system. The following are the metrics used for evaluation. Further details on these metrics is provided in [21, 34]

Table 1 Test cases

Test Case ID	Test Case	Event	Expected Behavior
TC_1	Validate the module when Undefined Rule Variable is used in Observable Predicate Formulation is fed as input	Markov Logic Networks and Evidence files are fed into the system along with the query	An error message stating "An unknown predicate name: QueryVariable" would be displayed
TC_2	Validate the module when Undefined Rule Variable is used in Hidden Predicate Formulation	Markov Logic Networks and Evidence files are fed into the system along with the query	An error message stating "An unknown predicate name: Predicate-Variable" would be displayed
TC_3	Validate the module when Undefined Query is fed as input	Markov Logic Networks and Evidence files are fed into the system along with the query	An error message stating "An unknown predicate name: QueryVariable" would be displayed
TC_4	Validate the module when Special characters are fed as input	Predicate Argument Structures (PAS) pre-processed and fed into Content Selection System (Semantic Similarity Matrix Computation)	A message stating "Special Characters are not allowed" would be displayed
TC_5	Validate Clustering of PAS when Empty Similarity Matrix (i.e. N=0) is fed as input	N × N similarity matrix with compression rate of summary fed into Hierarchical Clustering System	An error message stating "Number of items must be greater than zero" would be displayed
TC_6	Validate Clustering of PAS when the Order of Similarity Matrix are unequal (i.e. N!=N in N × N matrix)	N × N similarity matrix with compression rate of summary fed into Hierarchical Clustering System	An error message stating "Distance Matrix must be a square matrix" would be displayed
TC_7	Validate the selection of Population to ensure Uniqueness of samples in Donor ndividuals	Target Individuals selected, Fitness value defined followed by selection of Donor Individuals by Mutation Process	A message stating "Sample already picked under Target Individual Set" would be displayed
TC_8	Validate Realization of Sentences from PAS	Top scored PAS are fed into the Language Generation System and Syntactic Constituents are defined	Well-formed sentences from the respective PAS are to be stored

a) Pyramid Score:

Captures different sentences in summaries that uses different words but express similar meaning as per Equation (34).

$$\text{Pyramid Score} = \frac{\text{Total Peer SCUs Weight}}{\text{Average SCU in the Model Summary}} \tag{34}$$

where, Total SCU weight: $D = \sum_{i=1}^{n} i * D_i$

b) Average Precision:

Measures the degree of match between system generated summaries and model summaries as per Equation (35).

$$\text{Avg.Precision} = \frac{\text{Number of Model SCUs expressed in Peer Summary}}{\text{Average SCU in the Peer Summary}} \tag{35}$$

c) Longest Common Subsequence (ROUGE-L):

Evaluates based on length of the longest common subsequence between system summary and model summary as per Equation (36).

$$\begin{aligned}&\text{lcs}(\text{Seq}_{\text{system}},\text{Seq}_{\text{human}})\\&= \frac{\text{length}(\text{Seq}_{\text{system}}) + \text{length}(\text{Seq}_{\text{human}}) - \text{d}(\text{Seq}_{\text{system}}/\text{Seq}_{\text{human}})}{2}\end{aligned} \tag{36}$$

4.5 Comparison Result

The proposed system is tested with different standard datasets. The datasets that are chosen for testing are DUC2002, DUC2003, DUC2004. Rouge values for unigram, bigram, and n-gram for the chosen datasets are shown in the Table 2.

Table 2 Overall results for standard datasets

	R1 P	R1 R	R1 F	R2 P	R2 R	R2 F	RL P	RL R	RL F
DUC 2002	0.4184	0.6087	0.4959	0.2416	0.3673	0.2915	0.3861	0.5652	0.4294
DUC 2003	0.3987	0.5961	0.4778	0.3451	0.3267	0.3356	0.3521	0.5891	0.3967
DUC 2004	0.4333	0.6123	0.5075	0.3491	0.352	0.3505	0.4121	0.6951	0.4657

4.5.1 Comparison with Baseline System for Abstractive Summarization

The proposed system semantic oriented abstractive summarization have been compared with existing systems and the results have been tabulated below.

Table 3 shows the results recorded for comparative analysis of proposed semantic oriented abstractive summarizer with baseline summarization systems and model summaries evaluated with DUC 2002 dataset. Various combinations of the proposed system is analysed against the baseline system [21] with and without genetic algorithm. Best automatic summarization system and average automatic summarization system in DUC 2002 is also used for comparison. The proposed method has an average precision on 0.73 for Abs_GA_JSRL, which is better than the baseline method which has an average precision of 0.68 for Abs_GA_SRL.

Figure 25 illustrates the comparative analysis of the proposed system in which it is observed that proposed system's version Abs_GA_JSRL performs similar to baseline system Abs_GA_SRL in terms of pyramid score performance.Proposed Abs_GA_JSRL achieves the better performance than the other combination and this signifies that the system has tried to capture the summary content with the model summaries.

When comparison is made with the proposed and baseline system both incorporating genetic algorithm, proposed system still performs better due to the fact the Joint SRL out performs the trivial SRL process. The same is evident in the results of Abs_SRL and Abs_JSRL. In Sem-Graph system analyzing the semantic graph of the given input text with PageRank Algorithm and Maximum Marginal References, abstractive summary is produced. In Sem-Graph System, Word importance and Noun-Pronoun Matching are not taken into account. Moreover, Analysis of words using Parts-of-Speech tag

Table 3 Comparison analysis of semantic oriented abstractive summarizer

System Under Comparison	Model Name	Pyramid Score	Average Precision
Baseline System [22]	Sem-Graph	0.5247	0.72
Baseline System [21]	Abs_GA_SRL	0.5376	0.68
	Abs_SRL	0.2841	0.47
Baseline System [3]	ATS	0.5326	0.69
Baseline System [23]	Sem-graph-both-rel	0.5214	0.72
Proposed System	Abs_GA_JSRL	0.6451	0.73
	Abs_JSRL	0.4301	0.47

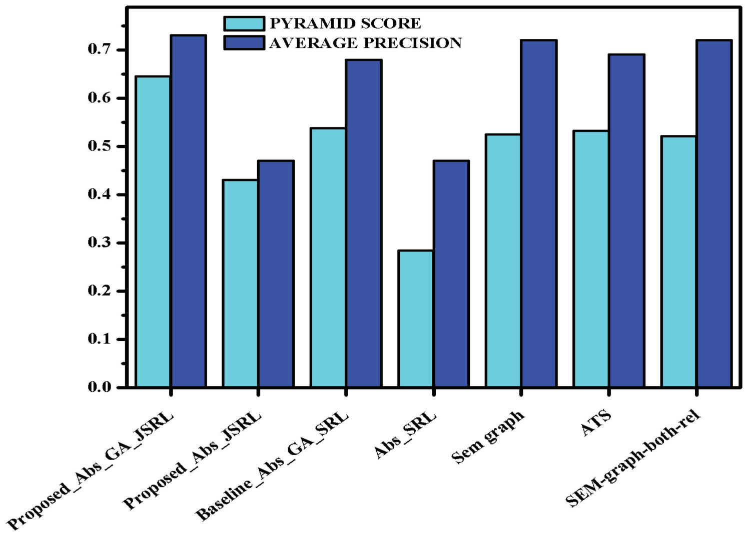


Figure 25 Comparative Analysis of abstractive Summarizer.

gives furthermore additional clarity in summarization creation. In Sem-graph-both-rel system works on multi-document Abstractive Summarization using ranking algorithm for PAS performs precision of 0.72. In Arabic text summarizer (ATS) for single document using RST, abstractive summary is produced. These systems are compared with our proposed system, using average precision of 0.73 and pyramid score is 0.6451.

4.6 Discussion

We live in an era of heterogeneous and expeditious information overload. This in turn demands advanced techniques to find meaningful data in a condensed form which saves time and effort. Text Summarization has become an important and timely tool for aiding a user to quickly understand large volume of information. Abstractive summarization's core functionality lies in its content and structure quality.

Furthermore, Semantic Oriented Abstractive Summarization takes a great leap forward when comparing to syntactic abstractive summarization, since source text understanding phase and the summary generation phase incorporates the semantics of the content thereby delivering more meaningful summary.

5 Conclusion and Future Work

Thus the Semantic Oriented Abstractive Summarizer is successfully developed. This system is able to generate abstractive summaries thereby exhibiting control over content and structure of the summaries generated. Semantic Representation of text is obtained via Joint Model (PSD+SRL) Semantic Role Labelling for content selection. Thus, this system has shown the feasibility of automatic abstractive summarization system can be incorporated by boosting the accuracy of summarization process. Additionally complex language generation techniques, such as information fusion, sentence compression and reformulation can still improve the target language process.

The proposed system can be modeled into domain specific such as medical records summarizer. Moreover, The system can be modeled with speech recognition system for summarizing lengthy orations. Search engines could use the system for summarizing the contents available for results.

References

[1] Achananuparp XH Palakorn, Yang CC (2009) "addressing the variability of natural language expression in sentence similarity with semantic structure of the sentences". In: Proceedings of Advances in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, Vol. 5476, pp. 548–555.

[2] Avanija J, Ramar K (2013) A hybrid approach using pso and k-means for semantic clustering of web documents. Journal of Web Engineering 12, no 3&4 pp: 249–264.

[3] Azmi AM, Altmami NI (2018) An abstractive arabic text summarizer with user controlled granularity. Information Processing and Management 54(6):903–921.

[4] Bakaev SHVK Maxim, Gaedke M (2019) Auto-extraction and integration of metrics for web user interfaces. Journal of Web Engineering 17, no 6&7: 561–590.

[5] Barros C, Lloret E, Saquete E, Navarro-Colorado B (2019) Natsum: Narrative abstractive summarization through cross-document timeline generation. Information Processing and Management.

[6] Che W, Liu T (2010) Using word sense disambiguation for semantic role labeling. Tech. rep., In Universal Communication Symposium (IUCS), 4th International, pp. 167–174. IEEE.

[7] Chen YC, Bansal M (2018) Fast abstractive summarization with reinforce-selected sentence rewriting. arXiv preprint arXiv:180511080.

[8] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. Journal of Machine Learning Research 12(Aug):2493–2537.

[9] Del Corro L, Gemulla R (2013) Clausie: clause-based open information extraction. In: Proceedings of the 22nd international conference on World Wide Web, ACM, pp 355–366.

[10] DUC (2002) Document understanding conference(duc) dataset. In: http://duc.nist.gov/data.html

[11] Gatt A, Reiter E (2009) Simplenlg: A realisation engine for practical applications. In: Proceedings of the 12th European Workshop on Natural Language Generation, Association for Computational Linguistics, pp. 90–93.

[12] Genest PE, Lapalme G (2010) Text generation for abstractive summarization. In: TAC.

[13] Genest PE, Lapalme G (2011) Framework for abstractive summarization using text-to-text generation. In: Proceedings of the Workshop on Monolingual Text-To-Text Generation, Association for Computational Linguistics, Stroudsburg, PA, USA, MTTG '11, pp 64–73, URL http://dl.acm.org/citation.cfm?id=2107679.2107687

[14] Greenbacker CF (2011) Towards a framework for abstractive summarization of multimodal documents. In: Proceedings of the ACL Student Session, Association for Computational Linguistics, pp. 75–80.

[15] GUO Y, PENG Y (2018) Semantic emotion-topic model in social media environment. Journal of Web Engineering 17, no 1&2 p: 073–092.

[16] Hou JL, Chen YJ (2013) Development and application of optimization model for customized text summarization. In: Computer Supported Cooperative Work in Design (CSCWD), IEEE 17th International Conference on, IEEE, pp. 246–250.

[17] Jadhav N, Bhattacharyya P (2014) Dive deeper: deep semantics for sentiment analysis. ACL p 113.

[18] Jiang JJ, Conrath DW (1997) Semantic similarity based on corpus statistics and lexical taxonomy. arXiv preprint cmp-lg/9709008.

[19] Kallimani JS, Srinivasa K, et al. (2011) Information extraction by an abstractive text summarization for an indian regional language. In: 7th International Conference on Natural Language Processing and Knowledge Engineering, IEEE, pp. 319–322.

[20] Karwa S, Chatterjee N (2014) Discrete differential evolution for text summarization. In: International Conference on Information Technology (ICIT), IEEE, pp. 129–133.

[21] Khan A, Salim N, Kumar YJ (2015) A framework for multi-document abstractive summarization based on semantic role labelling. Applied Soft Computing 30:737–747.

[22] Khan A, Salim N, Kumar YJ (2015) Genetic semantic graph approach for multi-document abstractive summarization. In: Fifth International Conference on Digital Information Processing and Communications (ICDIPC), IEEE, pp. 173–181.

[23] Khan A, Salim N, Farman H, Khan M, Jan B, Ahmad A, Ahmed I, Paul A (2018) Abstractive text summarization based on improved semantic graph approach. International Journal of Parallel Programming 46(5):992–1016.

[24] Knight K, Marcu D (2000) Statistics-based summarization-step one: Sentence compression. AAAI/IAAI 2000:703–710.

[25] Kupiec J, Pedersen J, Chen F (1995) A trainable document summarizer. In: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp. 68–73.

[26] Larsen B (1999) A trainable summarizer with knowledge acquired from robust nlp techniques. Advances in automatic text summarization p. 71.

[27] Liao K, Lebanoff L, Liu F (2018) Abstract meaning representation for multi-document summarization. arXiv preprint arXiv:180605655.

[28] Liu F, Flanigan J, Thomson S, Sadeh N, Smith NA (2018) Toward abstractive summarization using semantic representations. arXiv preprint arXiv:180510399.

[29] Lloret E, Palomar M (2012) Text summarisation in progress: a literature review. Artificial Intelligence Review 37(1):1–41.

[30] Lloret E, Boldrini E, Vodolazova T, Martínez-Barco P, Muñoz R, Palomar M (2015) A novel concept-level approach for ultra-concise opinion summarization. Expert Systems with Applications 42(20): 7148–7156.

[31] Lowd D, Domingos P (2007) Efficient weight learning for markov logic networks. In: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, pp. 200–211.

[32] Luhn HP (1958) The automatic creation of literature abstracts. IBM Journal of research and development 2(2):159–165.

[33] Mehta P, Majumder P (2018) Effective aggregation of various summarization techniques. Information Processing and Management 54(2): 145–158.

[34] Mitkov R (2014) Anaphora resolution. Routledge.

[35] Moawad IF, Aref M (2012) Semantic graph reduction approach for abstractive text summarization. In: Computer Engineering and Systems (ICCES), 2012 Seventh International Conference on, IEEE, pp. 132–138.

[36] Moratanch N, Chitrakala S (2016) A survey on abstractive text summarization. In: International Conference on Circuit, Power and Computing Technologies (ICCPCT), IEEE, pp. 1–7.

[37] Munot N, Govilkar SS (2014) Comparative study of text summarization methods. International Journal of Computer Applications 102(12).

[38] Ng JP, Abrecht V (2015) Better summarization evaluation with word embeddings for rouge. arXiv preprint arXiv:150806034.

[39] Persson J, Johansson R, Nugues P (2009) Text categorization using predicate–argument structures. In: Proceedings of NODALIDA, pp. 142–149.

[40] Porter MF (2001) Snowball: A language for stemming algorithms.

[41] Saggion H, Poibeau T (2013) Automatic text summarization: Past, present and future. In: Multi-source, multilingual information extraction and summarization, Springer, pp 3–21.

[42] Saif H, He Y, Fernandez M, Alani H (2016) Contextual semantics for sentiment analysis of twitter. Information Processing and Management 52(1):5–19.

[43] Salim N, Suanmali L, Binwahlan M (2010) Srl-gsm: a hybrid approach based on semantic role labeling and general statistic method for text summarization. Journal of Applied science, Vol. 110, no. 3, p:166–173.

[44] Shehata S, Karray F, Kamel MS (2013) An efficient concept-based retrieval model for enhancing text retrieval quality. Knowledge and information systems pp. 1–24.

[45] Titov I, Klementiev A (2012) A bayesian approach to unsupervised semantic role induction. In: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, pp 12–22.

[46] Tuffy (0.3) A scalable markov logic network (mln) inference engine. In: http://i.stanford.edu/hazy/tuffy/

[47] Zhang J, Zhou Y, Zong C (2016) Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. IEEE/ACM Transactions on Audio, Speech, and Language Processing 24(10):1842–1853.

Biographies

N. Moratanch is currently pursuing Ph.D. in Anna University, Chennai, Tamil Nadu, India. She has published 4 IEEE papers in International Conferences and one book chapter in Springer. Area of interest towards Data Mining, Natural Language Processing, Information Retrieval and Deep Learning.

S. Chitrakala is a Professor, Department of Computer Science and Engineering at Anna University, Chennai, Tamil Nadu, India. Her research interests include data mining, computer vision, artificial intelligence, web information retrieval and natural language processing, text mining, specifically application of statistical and NLP techniques in big data. Her research contributions have culminated in 108 publications which include 43 international journals and 65 international conferences. She is the reviewer for various journals and international conferences. She is a life member of CSI and life member of Indian Society of Technical Education ISTE, New Delhi.