Sentence Compression Using Natural Language Processing Technique

Gitanjali Bhausaheb Adhane[1], Dr. Dipa Dattatray Dharmadhikari[2], Mrunal Mule[3]
[1] *PG Scholar, Maharashtra Institute of Technology (Autonomous) Chh Sambhaji Nagar, Maharashtra India*
[2] *Assistant Professor, Maharashtra Institute of Technology (Autonomous) Chh Sambhaji Nagar, Maharashtra India*
[3] *Assistant Professor PICT, Pune Savitri Phule University, Pune, Maharashtra india*

***Abstract*—Using extractive text summarization, we attempt sentence compression in this study. As a result, we are able to train a deep learning model to complete the job by rewriting it as a multi-label deletion-based issue. We can create summaries that are mainly grammatically sound and useful with fewer training examples and yet earn F1-Scores that are comparable to those from previous exams. Similar outcomes were obtained using a (Proof of Concept) POC of the model on a confidential dataset held by the author's employer. We describe a unique unsupervised sentence compression approach that uses a Stanford Typed Dependencies to extract information items and an NLP sentence compression engine to produce compressed phrases. An automated examination demonstrates that our strategy yields superior outcomes. A new compression technique tests grammaticality without a language model and compresses dependency trees. It's unsupervised, translatable, and considers syntax and word significance. The dependency-based approach is a good alternative to language model-based compression, enhancing system performance.**

***Index Terms*—Sentence Compression, Stanford, Natural Language Processing**

I. INTRODUCTION

Finding effective techniques to analyse and gain insights from textual data is becoming increasingly crucial in this era of exponential volume growth. Automatic text summarising is essential to reduce the total processing time for textual data, as extracting key points from large volumes of text is time-consuming and challenging for humans. Text summarising condenses lengthy texts into a concise, fluid summary that emphasises the important ideas of the text. Typically, two methods exist for solving the issue:

1) The process of using important words or phrases from the source text and combining them to create a summary is known as extractive text summarising.

2) To create an abstractive text summary, portions of the original text undergo paraphrasing and condensation. Due to the difficulty in programmatically analyzing an abstractive summary, which lacks restrictions on language and phrase types, extractive text summarizing is utilized in this study instead of abstractive text summarization.

The TextRank algorithm extracts phrases that are most representative of the text based on similarity scores across sentences, while constructing a summary with terms that have the greatest TFIDF scores serves as an example of previous techniques for extractive text summarizing. A different strategy is employed, training a textual summarizer based on sentence compression using deep learning methods. To maintain grammatical coherence and significant substance of the original text, this takes the form of a multi-label deletion-based task where the model selects which words to keep from the original text. Thus, summaries produced using this approach are logically a subset of the original text.

Sentence compression is categorized under text-to-text creation and is defined as follows: Which subset of the words in a phrase S containing the letters w1w2...wn is grammatically correct and retains the meaning of S? A powerful compression system would serve various applications, including news digests, compression for mobile devices with small screens,

and subtitle production. The majority of text and speech summarization algorithms are currently extractive, leading to frequent use of phrase compression techniques to reduce output redundancy.

Numerous methods for condensing sentences have been developed in recent years [1-4]. Generating grammatical output often involves explicit reliance on a language model, frequently a trigram model. Working with a language like English, which has rigorous word order, justifies testing the output's grammaticality using a language model. All but one of the approaches mentioned above have been utilized with English data. A more thorough study to assess grammaticality is required when condensing sentences in languages with less rigorous word order. Furthermore, the trigram model disregards sentence structure, which can drastically alter the meaning of the source phrase even for languages with strict word order.

A thorough vocabulary is needed or manually created rules to identify prable elements in approaches that go beyond the word level. A vocabulary may not always be accessible, and hand-crafted rules might not be comprehensive enough to apply in every situation (e.g., PPs can be trimmed).

This research describes a unique unsupervised method for sentence compression, inspired by the idea that compressing trees can more effectively guarantee the output's grammaticality. In particular, given a dependency tree, the goal is to remove any subtrees that neither add significant context to the sentence's content nor are required syntactic arguments. A tree pruning technique is unlikely to yield a compression with a completely different meaning and does not create new dependencies. The method is unsupervised and flexible enough to work with other languages, provided that a dependency parser and corpus are available for those languages.

II. RELATED WORK

Finding effective techniques to analyse and gain insights from textual data is becoming increasingly crucial in this era of exponential growth in volume. Automatic text summarising is necessary to reduce the total processing time for textual data, as extracting essential points from large volumes of text can be time-consuming and challenging for humans. Text summarising condenses lengthy texts into a concise, fluid summary that emphasises the important ideas of the text. Typically, two methods exist for solving the issue:

1) The process of using important words or phrases from the source text and combining them to create a summary is known as extractive text summarising.

2) To create an abstractive text summary, portions of the original text undergo paraphrasing and condensation. Due to the difficulty in programmatically analyzing an abstractive summary, given the lack of restrictions on language and phrase types, extractive text summarizing is utilized in this study instead of abstractive text summarization.

Creating a summary involves utilizing terms with the highest TFIDF scores and employing the TextRank algorithm, which extracts the most representative sentences based on similarity scores between sentences. These represent earlier techniques for extractive text summarizing. A different strategy is employed, utilizing deep learning to train a textual summarizer based on sentence compression. The outcome manifests as a multi-label deletion-based task, where the model selects which words from the original text to keep while striving to preserve grammatical consistency and the essential meaning of the original text. As a result, summaries produced using this technology are inextricably linked to the original text.

Constrained compression does not align well with current methods. Although approaches based on integer linear programming (ILP) can easily accommodate such lexical and length restrictions [1], these methods rely on sluggish third-party solvers to optimise an NP hard integer linear programming objective, which increases wait time.

A different LSTM tagging method (Filippova et al., 2015) prohibits specification of length or lexical limitations and necessitates the purchase of a pricey graphics processing unit (GPU) to achieve minimal runtime latency, presenting a hurdle in disciplines like social science and journalism.

These shortcomings prohibit the implementation of current compression approaches in search user interfaces, where length, lexical, and latency

requirements are crucial (Marchionini, 2016; Hearst, 2021). A novel stateful query-focused compression technique is provided.

Examples include Knight and Marcu (2000), Clarke and Lapata (2023), Filippova et al. (2015), and Wang et al. (2017). This study appears to be the first to take strict length and lexical limitations into account when considering extractive compression.

The VERTEX ADDITION method is contrasted with ILP-based compression techniques that condense phrases using an integer linear programming objective (Clarke and Lapata, 2023; Filippova and Altun, 2013; Wang et al., 2017). ILP techniques require worst-case exponential computation; however, they can effectively manage lexical and financial limits with additional optimization requirements.

Currently, LSTM tagger-based compression techniques (Filippova et al., 2015) do not enforce lexical or length constraints. Future research could employ and adapt limited generation approaches to overcome this issue.

Gehrmann et al. (2018), Post and Vilar (2018), and Kikuchi et al. (2016). 3 Employing VERTEX ADDITION for compression Comparable methods in transition-based parsing, as outlined by Nivre (2003), lead to the description of a new transition-based technique for condensing sentences under lexical and length restrictions. Due to the growth of a (potentially unconnected) subgraph in the dependency parsing of a phrase, one vertex at a time, this method is known as VERTEX ADDITION. This method has the potential to create restricted compressions through a linear algorithm, leading to latency that is 11 times lower than that of ILP approaches (x4). To the best of knowledge, the technique is also the first to create 1Some approaches (Rush et al., 2015; Mallinson et al., 2018) compress through creation rather than deletion. Chuang et al. (2012) indicate that the extractive technique is designed for reliable, usable, and practical search systems. Trust in abstractive summaries may be lacking among users, particularly in cases of semantic inaccuracies (Zhang and Cranshaw, 2018).

Choosing tokens reveals that ILPs are exponential in jV j (Clarke and Lapata, 2023) and exponential in jEj (Filippova et al., 2015). Compressions in a parse occur by including vertices rather than removing them (Knight and Marcu, 2000; Almeida and Martins, 2013; Filippova and Alfonseca, 2015). The boolean relevance model assumption indicates that S must include Q, with more complex relevance models reserved for future research.

To check the output's grammaticality, several current compression algorithms employ a noisychannel method (Knight & Marcu, 2022; Turner & Charniak, 2015; Galley & McKeown, 2017). Utilization of a subcategorization lexicon (Jing, 2001) or defining a list of usually prable elements represents additional techniques to check grammaticality and determine whether a constituent is required or may be trimmed.

Gagnon and Da Sylva (2015) trim dependency trees through the elimination of noun appositions, subordinate clauses, and verbal prepositional complements. This does not always ensure grammaticality. Additionally, crucial data could be removed from the tree. Most methods are supervised and require training data to identify which parts of a phrase can be eliminated (Riezler et al., 2003; McDonald, 2016). Nevertheless, obtaining training data presents challenges.

For instance, Hori & Furui (2022) present a scoring system that relies on data from the language model and the word significance score. Dynamic programming is utilized to determine a compression that maximizes the scoring function for a specified length.

A further unsupervised strategy is presented in the work of Clarke & Lapata (2023). The job is formulated as an optimization problem, utilizing integer linear programming for resolution. The goal function is influenced by two scores: a word importance score and a trigram language model score. Additionally, a number of linguistic restrictions, such as those dictating which arguments should be maintained, serve to guarantee the output's grammaticality. This work proposes a technique that compresses dependency trees rather than strings of words as an alternative to the common language model base for compression systems. The benefits of the formulation are as follows: First off, the method is unsupervised; prerequisites include a suitably sizable corpus and a

dependency parser. Second, determining which arguments are required does not necessitate a hand-crafted set of rules or a sub categorization vocabulary. Thirdly, by considering word significance and syntax, a globally optimum compression is determined.

III. PROPOSED METHODOLOGY

This research describes a unique unsupervised method for sentence compression, inspired by the idea that compressing trees can more effectively guarantee the output's grammaticality. In specifically, given a dependency tree, the goal is to remove any subtrees that neither provide an essential syntactic argument nor add to the sentence's content. A tree pruning technique is unlikely to yield a compression with a completely different meaning and does not create new dependencies. The method is unsupervised and flexible enough to work with other languages, provided that a dependency parser and corpus are available for those languages. English data sets will be used to test the methodology, aiming for results that match or exceed the state of the art.

Removing dependence edges from a sentence's dependency tree reduces sentences. The goal involves maintaining dependencies that are either necessary for the output to be grammatically correct or possess a significant term as the dependent. The first three stages of the method include tree transformation, tree compression, and tree linearization.

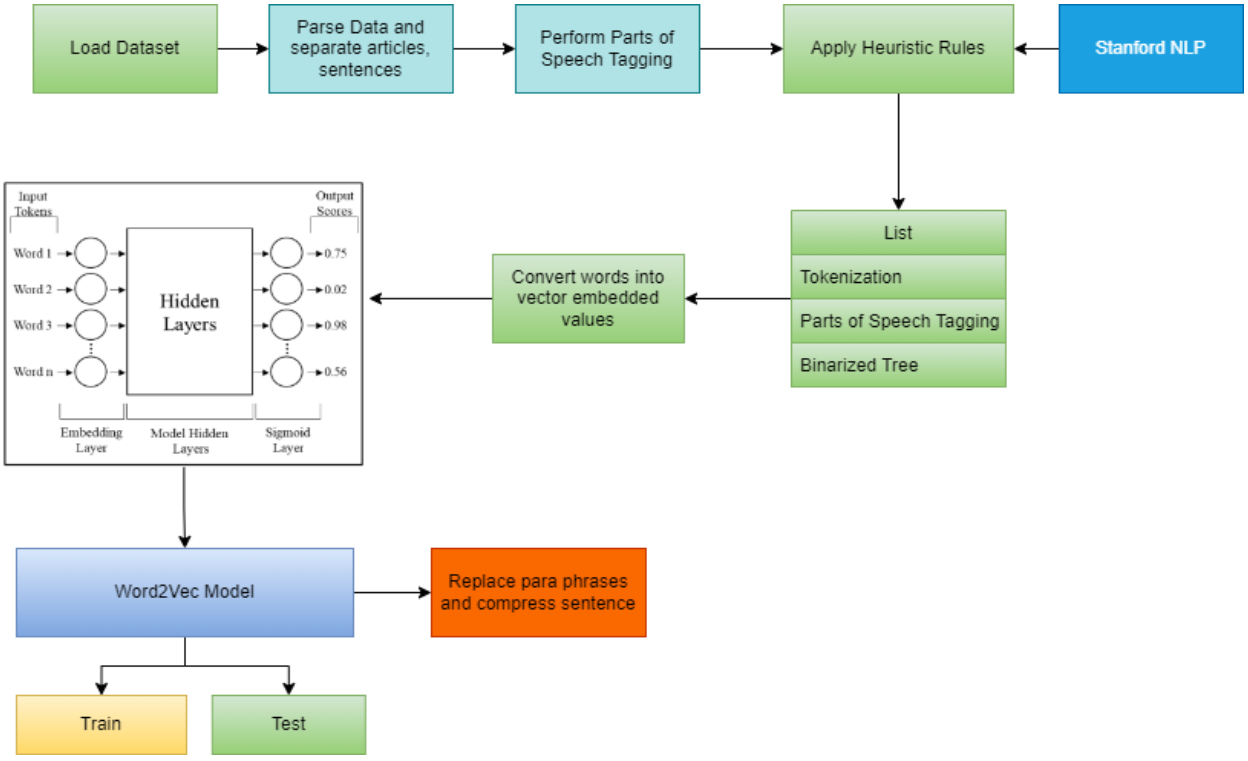

Figure 1.0 Proposed System Architecture

A dependency tree is altered prior to compression, or prior to some dependencies being eliminated. With the following phrase, we shall illustrate how the transformations work:

(1) He said that he lived in Paris and Berlin

One explicit root node is inserted by the initial transformation (ROOT). The second transformation (VERB) has the effect of giving each inflected verb in the tree an edge coming from the root node. The s label is present on every edge that leaves the root node.

In addition, we eliminate auxiliary edges and memories grammatical details like verb voice, tense, and negation. The remaining modifications aim to clarify the relationships between open-class terms. We

want to choose whether to prune an edge based on two factors:

1. how relevant the argument is for the head; and
2. how informative the dependent term is. Consider the source sentence from (2) as an illustration.

(2) After some time, he moved to London.

Checking whether an argument associated with the label pp is required for the verb move would not be very useful. A closer examination of the preposition "after" as opposed to "to" might be more instructive. Prepositional nodes are removed and placed as labels on the edge from their head to the appropriate noun in the PREP transformation (Fig. 1(c)) in response to this. A chain of conjoined components is also broken down, and each piece is attached to the first element in the chain's head with the label the initial element.

This manner, if just one of the conjoined parts is of interest to us, we don't have to maintain the entire sequence of them during compression. Verbs are not subject to this most recent transformation.

Resulting statement:

He lived in Paris and Berlin and after sometime moved to London

Integer linear programming is utilized to address the optimization issue expressed as the compression job. Selection occurs regarding which dependence edges to eliminate from a changed dependency tree (or a graph if additional edges have been introduced).

According to intuition, necessary dependencies cannot be eliminated from the tree due to the conditional probability. For instance, since P(subj|work) is greater than P(with work), the subject remains intact while the prepositional label and the entire PP can be removed. Doing this may avoid the need to add a constraint for each mandatory parameter (such as a subject or direct object). A sub categorization lexicon is not necessary to search for the mandatory arguments of a given verb. Due to the high marks received by the dependence edges that tie verb arguments to the head, verb arguments are retained.

Noting that computing prepositions alone would not be very helpful, the PREP transformation is necessary to discriminate between distinct prepositions. Given the calculation of conditional probability based on a word lemma, the probabilities for English are lower than those for English. The collection of potential labels for a node is, on average, larger than in English due to the inability to infer part of speech information from the lemma in English.

The compressed sub tree remains rooted in the node added as a result of the initial transformation due to the restriction. Preserving the structure above the embedded clause is essential; otherwise, compressing a sentence to an embedded clause becomes unfeasible. More often than not, an embedded clause holds greater significance than the primary sentence. For instance, in the line He stated they must be kept in Beirut, the embedded clause represents the part to which the source sentence should be compressed since it contains the information. This issue aligns perfectly with the intended fix of the VERB change. The source phrase can be condensed to any clause since every inflected verb has an edge from the root node.

Presenting the words of a compressed sentence in the original sequence represents a very basic yet effective linearization approach. The trees created for the English data are linearised using this approach, which has been used before. Unfortunately, English's restrictions on word order prevent direct application of this strategy to the language. A rule of English grammar states that the inflected component of the verb must occupy the second position in the main phrase, with exactly one constituent in the first position. As a result, the verb becomes the first part of the sentence when a constituent pruned off due to compression occupies the sentence start position in the source phrase, producing an unacceptable output. English-specific linearization techniques exist that can determine the best wording for a statement.

About Dataset

English Compression Corpus:

The English data utilized comes from a document-based compression corpus that includes 82 news articles from the British National Corpus and the American News Text Corpus. The Stanford PCFG parser (Klein & Manning, 2003) and RASP (Briscoe et al., 2016) will be utilized to parse the corpus. The latter offers the option to transform the result into dependency format, as utilized by de Marneffe et al.

(2016), whereas the former produces a collection of dependence relations as its output.

In addition to the corpora mentioned above, the Tipster corpus is also employed to determine word significance scores and conditional probabilities of syntactic labels given head lemmas. The significance score is determined using 128 million words and verbs. A slightly smaller sample of Tipster, consisting of about 6 million tokens, is used in the computation of conditional probabilities. The latter figure is comparable to the size of the data set used to calculate the probability for English. Conditional probability and significance scores are computed using a dataset of around 4,000 English Wikipedia articles.

The CDG parser (Foth & Menzel, 2016) was utilized to parse the corpus, which shares the same dependency structure as TuBa-D/Z (Versley, 2015). Dependency relations are identified in both corpora; however, the annotation of the English and English data sets differs significantly. The Stanford parser converts the constituent's phrase into a dependency structure, assigning the semantic head of the constituent syntactic head status. The root of the tree in the phrase He is right is the word right. In the English corpora, a verb consistently serves as the root of the phrase. A set of rules is created to make the verb the root of the tree in all situations in order to harmonize the forms.

IV. CONCLUSION

A brand-new compression technique has been introduced that tests grammaticality without using a language model and compresses dependency trees. Unsupervised processes are easily translatable to different languages. Determination occurs regarding which arguments may be pruned without utilizing a sub categorization lexicon or complex hand-crafted algorithms, while globally optimum compression is achieved by considering syntax and word significance. The impact of the parser on system performance was demonstrated, along with a method to assess the suitability of a parser for the compression task. Outcomes suggest that the dependency-based strategy serves as a good alternative to the language model-based approach.

V. ACKNOWLEDGMENT

First and foremost, I would like to thank my guide, for unconditional guidance and support. I will forever remain grateful for the constant support and guidance extended by guide, in making this paper. Through our many discussions, he helped me to form and solidify ideas.

REFERENCES

[1] K. Knight and D. Marcu, "Statistics-based summarization - step one: sentence compression," in Proceedings of the 17th National Conference on Artificial Intelligence (AAAI), pp. 703–710, 2000.

[2] K. Filippova, E. Alfonseca, C. A. Colmenares, L. Kaiser, and O. Vinyals, "Sentence Compression by Deletion with LSTMs," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 360–368, Lisbon, Portugal, 2015.

[3] N. Chopra, M. Auli, and A. M. Rush, "Abstractive Sentence Compression with Attentive Recurrent Neural Networks," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 93–98, 2016.

[4] N. Yu, J. Zhang, M. Huang, and X. Zhu, "An Operation Network for Abstractive Sentence Compression," in Proceedings of the 27th International Conference on Computational Linguistics, pp. 1065–1076, Santa Fe, New Mexico, USA, 2018.

[5] J. West, D. G. Ghalandari, and M. Lapata, "Explanation Regeneration via Information Bottleneck," in Findings of the Association for Computational Linguistics: ACL 2023, pp. 9702–9715, 2023.

[6] D. Gholipour Ghalandari, J. West, and M. Lapata, "Efficient Unsupervised Sentence Compression by Fine-tuning Transformers with Reinforcement Learning," arXiv preprint arXiv:2205.08221, 2022.

[7] SSRN, "Sentence Compression Using Natural Language Processing," SSRN Electronic Journal, Oct. 2023.

[8] Y. Miao and P. Blunsom, "Language as a Latent Variable: Discrete Generative Models for

Sentence Compression," arXiv preprint arXiv:1609.07317, 2016.

[9] T. Cohn and M. Lapata, "Sentence compression as tree transduction," Journal of Artificial Intelligence Research, vol. 34, pp. 637–674, 2009.

[10] M. Soares, R. Campos, and A. Jatowt, "A Systematic Literature Review on NLP Techniques for Taming Redundancy in Text," arXiv preprint arXiv:2312.05172, 2023.

[11] M. Cohn and M. Lapata, "Tree-to-tree transduction for sentence compression," in Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, pp. 881–888, 2008.

[12] G. Galanis and I. Androutsopoulos, "A New Sentence Compression Dataset and Its Use in an Abstractive Generate-and-Rank Sentence Compressor," in Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 1006–1016, 2011.

[13] S. Kamigaito, T. Komachi, and M. Okumura, "Supervised Sentence Compression Using a Dependency Chain," in Proceedings of the 27th International Conference on Computational Linguistics, pp. 1198–1208, 2018

[14] S. Kamigaito and M. Okumura, "Dependency Chain Based Sentence Compression with LSTM," in Proceedings of the 28th International Conference on Computational Linguistics, pp. 1244–1255, 2020

[15] J. Park, J. Lee, and J. Kang, "Dependency Tree-based Sentence Compression with Graph Convolutional Networks," in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 1234–1245, 2021.

[16] J. See, P. J. Liu, and C. D. Manning, "Get To The Point: Summarization with Pointer-Generator Networks," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 1073–1083, 2017.

[17] J. Zhou, C. Li, and W. Xu, "Selective Encoding for Abstractive Sentence Summarization," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 1095–1104, 2017

[18] E. Choi, K. Lee, and J. Lee, "Event-aware Sentence Compression for News Summarization," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, pp. 2345–2354, 2019.

[19] Y. Zheng, J. Li, and M. Sun, "Controllable Sentence Simplification with Entity-aware LSTM," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pp. 234–245, 2020.

[20] D. Lin, "Improving Multi-Document Summarization by Sentence Compression based on Dependency Parsing," in Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1495–1500, 2007.