Paraphrastic Fusion for Abstractive Multi-Sentence Compression Generation

Mir Tafseer Nayeem
University of Lethbridge
Lethbridge, AB, Canada
mir.nayeem@uleth.ca

Yllias Chali
University of Lethbridge
Lethbridge, AB, Canada
chali@cs.uleth.ca

ABSTRACT

This paper presents a first attempt towards finding an abstractive compression generation system for a set of related sentences which jointly models sentence fusion and paraphrasing using continuous vector representations. Our paraphrastic fusion system improves the informativity and the grammaticality of the generated sentences. Our system can be applied to various real world applications such as text simplification, microblog, opinion and newswire summarization. We conduct our experiments on human generated multi-sentence compression datasets and evaluate our system on several newly proposed Machine Translation (MT) evaluation metrics. Our experiments demonstrate that our method brings significant improvements over the state of the art systems across different metrics.

KEYWORDS

Sentence Fusion, Multi-Sentence Compression, Lexical Paraphrasing, Abstractive Compression Generation

1 INTRODUCTION

The task of automatic document summarization aims at finding the most relevant informations in a text and presenting them in a condensed form. A good summary should retain the most important contents of the original document or a cluster of documents, while being non-redundant and grammatically readable. There are two types of summarizations: extractive summarization and abstractive summarization. Extractive summarization systems select the salient (important) sentences from the source document without any modification to create a summary. On the other hand, abstractive summarization methods, which are still a growing field, are highly complex as they need extensive natural language generation to rewrite the sentences. The abstractive techniques which is traditionally used are sentence compression, fusion and lexical paraphrasing. However, in case of multi-document summarization where source documents usually contain similar information, the extractive methods would produce redundant summary or biased towards specific source document.

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3133106

Multi-sentence compression (MSC) can be a useful solution for the above problem. It usually takes a group of related sentences and produces an output sentence through merging the sentences about the same topic [2]. On the other hand, Lexical paraphrasing aims at replacing some selected words with other similar words while preserving the meaning of the original text. A good lexical substitution for a target word needs to be semantically similar to the target word and compatible with the given context [8]. For example, the sentence "Jack composed these verses in 1995" could be lexically paraphrased into "Jack wrote these lines in 1995" without altering the sense of the initial sentence.

2 RELATED WORKS

Most of the previous **MSC** approaches rely on syntactic parsers to build dependency tree [5]. Unfortunately, syntactic parsers are not available for all the languages. As an alternative, word graph-based approaches that only require a **POS** tagger and a list of stopwords have been proposed first by Filippova [4]. A directed word graph is constructed in which nodes represent words and edges represent the adjacency between words in a sentence. Hence, compressed sentences are generated by finding k-shortest paths in the word graph. Boudin and Morin [3] improved Filippova's approach by re-ranking the fusion candidate paths according to keyphrases to generate more informative sentences. However, grammaticality is sacrificed to improve informativity in these works.

Banerjee [1] proposed an abstractive multi-document summarization system using Filippova's sentence fusion approach [4] combined with Integer Linear Programming sentence selection. Following Banerjee's work, several recent approaches have been proposed with slight modifications. Multiword Expressions (MWE) was exploited in [14] to produce more informative compressions. Recently, [15] use syntax factor along with the Banerjee's model to generate compressions. However, all the above mentioned systems try to produce compressions by copying the source sentence words, no paraphrasing is involved in the process.

3 PARAPHRASTIC SENTENCE FUSION : MODEL

3.1 Sentence Abstraction Techniques

Most of the previous works rely only on one of the following techniques for abstracting sentences. Instead, in this paper we take the first step towards finding a joint representation for sentence abstraction using sentence fusion and lexical paraphrase rather than treating these two independently.

(1) Sentence Compression
- [*ACD*~~*EFA*~~*GED*] ⇒ [*ACDGED*]

- Deletion of unimportant words from the input sentence.
- Mainly used for summarizing a sentence or headline generation.

(2) Sentence Fusion
- $[ACD\sout{EFA}GED] + [\sout{CDEF}BADE] \Rightarrow [ACDGEDBADE]$
- Involves the merging of two or more sentences into one.
- Reduces redundancy in the final generated summary.

(3) Lexical Paraphrase
- $[A\sout{CD}EFA\sout{GE}DHB] \Rightarrow [A\mathbf{B}GEFA\mathbf{BC}DHB]$
- Replaces complex words with simple words to make the sentence easier to understand.

3.2 Sentence Fusion

3.2.1 Word Graph Construction. In order to generate a one sentence representation from a cluster of related sentences we use the word-graph approach of [4]. Let $S = \{s_1, s_2, ..., s_n\}$ is a set of related sentences, we construct a graph $G = (V, E)$ by iteratively adding sentences to it. The vertices are the words along with the parts-of-speech (POS) tags and directed edges are formed by simply connecting the adjacent words in the sentences. Once the first sentence is added, words from the other related sentences are mapped onto a node in the graph provided that they have exactly the same lower cased word form and the same POS tag. Each sentence is connected to dummy start and end nodes to mark the beginning and ending of the sentences.

In Filippova's approach [4], punctuation marks are not considered. To generate well-punctuated compressions which in turn represents complete sentences, we considered a fourth step for adding punctuation marks in the graph following [3]. Consider the following two sentences,

S1: The video was made on Feb. 19-20, 2003.

S2: The morning after the video was made, she said, three social workers came and interviewed them.

As we can see, the two input sentences contain similar information, but differs in sentence length, syntax, and the detail of information. After constructing the word-graph, we can generate K-Shortest paths from dummy start node to end node in the word graph. For example, we can generate paths like the following:

Ex1: The morning after the video was made on feb. 19-20 , 2003.

The main challenge of sentence fusion is to rank sentences generated from the K-Shortest paths (K is usually ranges from 50 to 200 according to the literature [4] [3]) that are grammatically correct and contain the most important information. Hence, we design a candidate re-ranking strategy which is generated from K-Shortest paths based on grammaticality and informativeness.

3.2.2 Candidate Ranking. We rank the fused candidates by applying **TextRank** algorithm [9] which involves constructing an undirected graph where candidates are vertices, and weighted edges are formed by connecting candidate sentences by a similarity metric. TextRank determines the similarity based on the lexical overlap between two sentences. However, this algorithm has a serious drawback: If two sentences are talking about the same topic without using any overlapped words, there will be no edge between them. Instead, we use the continuous skip-gram model introduced by [10] to measure the semantic similarity along with entity overlap. We take the pre-trained word embeddings[1] [10] of all the non stopwords in a sentence and take the weighted vector sum according to the term-frequency (TF) of a word (w) in a sentence (S). Where, E is the word embedding model and $idx(w)$ is the index of the word w. More formally, for a given sentence S, the weighted sum becomes,

$$S = \sum_{w \in S} TF(w, S) \cdot E[idx(w)]$$

Then we calculate the cosine similarity between the sentence vectors obtained from the above equation to find the relative distance between S_i and S_j. We also calculate $NESim(S_i, S_j)$ by finding the Named Entities present in S_i and S_j using NLTK Toolkit, then calculating their overlap.

$$CosSim(S_i, S_j) = \frac{S_i \cdot S_j}{||S_i||\ ||S_j||}$$

$$NESim(S_i, S_j) = \frac{|Entity(S_i) \cap Entity(S_j)|}{min(|Entity(S_i)|\ ,\ |Entity(S_j)|)}$$

$$Sim(S_i, S_j) = \lambda \cdot NESim(S_i, S_j)\ +\ (1 - \lambda) \cdot CosSim(S_i, S_j) \quad (1)$$

The overall similarity calculation involves both $CosSim(S_i, S_j)$ and $NESim(S_i, S_j)$ where, $0 \leq \lambda \leq 1$ decides the relative contributions of them to the overall similarity computation. We use the similarity function described in Equation (1) by empirically setting $\lambda = 0.3$.

After we have our graph, we can run the **TextRank** algorithm on it. This involves initializing a score of 1 for each vertex, and repeatedly applying the TextRank update rule until convergence. The update rule is:

$$Rank(S_i) = (1 - d) + d * \sum_{S_j \in N(S_i)} \frac{Sim(S_i, S_j)}{\sum_{S_k \in N(S_j)} Sim(S_j, S_k)} Rank(S_j)$$

Where, $Rank(S_i)$ indicates the importance score assigned to sentence S_i. $N(S_i)$ is the set of neighboring sentences of S_i.

3.2.3 Grammatical Quality. We compute grammatical quality of a fused sentence candidate using a 3-gram (trigram) language model same as [1], which assigns probabilities to sequence of words in a generated candidate. Suppose that a candidate contains a sequence of m words $\{w_1, w_2, w_3, ...w_m\}$. The score GQ (Gramatical Quality) assigned to each candidate is defined as follows:

$$GQ(w_1, ..., w_m) = \frac{log_2 \prod_{t=3}^{m} P(w|w_{t-1}w_{t-2})}{N} \quad (2)$$

The scores are normalized by N, the word length of the candidates. In our experiments, we used a 3-gram model that is trained on the English Gigaword corpus [2].

Finally, we re-rank the K candidate fusions and find the N-best sentence fusion using the following equation which balances the grammaticality and the informativeness. The score of a candidate sentence fusion c is given by (where, we empirically set $\alpha = 0.6$)

$$score(c) = \alpha \cdot Rank(c) + (1 - \alpha) \cdot GQ(c) \quad (3)$$

[1] https://code.google.com/archive/p/word2vec/

[2] Available : http://www.keithv.com/software/giga/ (We used the 64K NVP vocabulary version)

3.3 Lexical Substitution in Context

3.3.1 Target Word Identification for Substitution . We first label the words in all N-best candidates using Part-Of-Speech (POS) tagging. We then filter out the named entities and take only the nouns and verbs for possible substitution candidates.

3.3.2 Substitution Selection. The PPDB 2.0[3] [12] provides millions of lexical, phrasal and syntactic paraphrases which comes into packages of different sizes (going from S to XXXL)[4]. For instance, we can gather $S = \{gliding, sailing, diving, travelling\}$ lexical substitution set for the target word ($t = flying$) from PPDB 2.0. We hardcoded the model to select substitutes with the same POS tag and that are not a morphological variant (*e.g. fly, flew, flown*).

3.3.3 Substitution Ranking. Word embeddings are low dimensional vector representations of words such as word2vec [10] that recently gained much attention in various semantic tasks. Word2vecf[6] is an extension of word2vec to produce syntax-based word embeddings. They show that these embeddings tend to capture functional word similarity (as in *manage* → *supervise*) rather than topical similarity (as in *manage* → *manager*). We use the word and context vectors released by [8] which was shown to perform strongly on lexical substitution task. These embeddings contain 600d vectors for 173k words and about 1M syntactic contexts. Their measure *addCos* for estimating the appropriateness of a substitute s from the substitution set S, for the target word t in the set of the target word's context elements $C = \{c_1, c_2, ..., c_n\}$, is defined as follows,

$$addCos(s|t, C) = \frac{cos(s, t) + \sum_{c \in C} cos(s, c)}{|C| + 1}$$

Finally, we select the best substitution **s** according to maximum **addCos** scores over **0.7** and replace it with the target word **t**.

4 EXPERIMENTAL SETUP

Our system first takes a set of related texts as input and preprocesses them which includes tokenization, Part-Of-Speech (POS) tagging, and Lemmatization. We use **NLTK** toolkit to preprocess each sentence to obtain a more accurate representation of the information. We generate 50 shortest paths from start to end nodes from each cluster in the graph using the K-shortest path algorithm [3]. Paths shorter than eight words or that do not contain a verb are filtered out. To ensure pure abstractive compression generation, we remove the paths that have $cosineSimilarity \geq 0.9$ to any of the original sentence in the cluster. We then select 3-Best candidates from K paths for lexical substitution. For fair evaluation, we also select the 3-best candidates for the baseline systems that we compare with our model.

4.1 Dataset

We conducted experiments on the human generated sentence fusion dataset released by [7]. This dataset consists of 300 English sentence pairs taken from newswire clusters accompanied by human-produced sentence fusions rewrites collected via Amazon's Mechanical Turk service. We filtered the sentences which have no main verbs. The resulting set contains 296 pairs of sentences.

[3]http://paraphrase.org/

[4]For our experiment, we use the XXL lexical one

4.2 Evaluation Metric

We evaluate our system automatically using various automatic metrics. We also introduce some new automatic evaluation metrics.

BLEU [11] is the most commonly used metric for Machine Translation evaluation. BLEU relies on exact matching of n-grams and has no concept of synonymy or paraphrasing. We used the implementation provided in NLTK[5] considering upto 4-gram matching.

SARI [16] a recently proposed metric which compares **S**ystem output **A**gainst **R**eferences and against the **I**nput sentence. SARI computes the arithmetic average of n-gram precision and recall of three rewrite operations: addition, copying, and deletion which correlates well with human references.

METEOR-E [13] uses a combination of both precision and recall in METEOR metric. Furthermore, the alignment is based on exact token matching, followed by WordNet synonyms, stemmed tokens and then look-up table paraphrases. Recently, an augmented version of METEOR using distributed representations named **METEOR-E**[13] has been released[6].

Compression Ratio is a measure of how terse a compression is and is expressed in the following equation. A compression ratio of zero implies that the source sentence is fully uncompressed.

$$Compression\ Ratio\ (CR) = \frac{\#tok_{del}}{\#tok_{orig}}$$

Copy Rate We define copy rate as how many tokens are copied to the abstract sentence from the source sentence without paraphrasing in the following equation. Copy rate of 100% means no paraphrasing is involved in the process.

$$Copy\ Rate = \frac{|S_{orig} \cap S_{abs}|}{|S_{abs}|}$$

Grammaticality We define grammaticality as the parsing problem, if the sentence is successfully parsed, then it has valid grammar; if not, then it doesn't. We didn't use any statistical parser because, the parser will still return a parse for a sentence with bad grammar as it uses the statistics to make the best guess possible. Instead, we use a chart parser to parse a sentence, given a CFG (Context-Free Grammar) which is implemented in NLTK Toolkit.

4.3 Baseline Systems and Results

We compare our system with Filippova (2010) [4], Boudin and Morin (2013)[7] [3] and Banerjee et al. (2015)[8] [1]. Table 2 shows the output generated by the baseline and our system. We report our system's performance compared with the baselines in terms of different evaluation metrics in Table 1. We get slightly higher score in terms of **SARI** because of the multiple human abstractive rewrites along with input sentence. Copy Rate score of other baseline systems clearly indicates the fact that they are doing completely compression, no paraphrasing is involved. Moreover, we also get higher score in **METEOR-E** metric because of the lexical substitution operation. In our experiments, we used LM (Language Model) which tends to choose longer sentences. Therefore, we get lower compression ratio than Filippova (2010). However, we achieve higher

[5]https://github.com/nltk/nltk/tree/develop/nltk/translate

[6]https://github.com/cservan/METEOR-E

[7]https://github.com/boudinfl/takahe

[8]https://github.com/StevenLOL/AbTextSumm

Table 1: Comparison with baselines across different automatic evaluation metrics (the scores are averaged)

Model	BLEU	SARI	METEOR-E	Compression Ratio	Copy Rate	Grammaticality(%)
Filippova (2010) [4]	40.6	34.6	0.31	**0.57**	99.8	58.2%
Boudin and Morin (2013) [3]	**44.0**	37.2	0.36	0.42	99.9	65.8%
Banerjee et al. (2015) [1]	42.3	36.5	0.34	0.45	99.8	71.4%
Paraphrastic Fusion	42.5	**37.4**	**0.43**	0.41	**76.2**	**73.5%**

Table 2: The output generated by the baseline and our system (the paraphrased words are marked bold)

Input Sentences	Bush, who initially nominated Roberts to replace retiring Justice Sandra Day O'Connor, tapped him to lead the court the day after Rehnquist's death. President Bush initially nominated Roberts in July to succeed retiring Justice Sandra Day O'Connor.
Filippova (2010)	president bush initially nominated roberts to replace retiring justice sandra day o'connor .
Boudin and Morin (2013)	bush , who initially nominated roberts in july to succeed retiring justice sandra day o'connor , tapped him to lead the court the day after rehnquist 's death .
Banerjee et al. (2015)	bush , who initially nominated roberts to replace retiring justice sandra day o'connor , tapped him to lead the court the day after rehnquist 's death .
Paraphrastic Fusion	president bush initially **recommended** roberts in july to **accomplish** retiring justice sandra day o'connor , tapped him to **run** the court the day after rehnquist 's death .

grammaticality percentage. As expected, we get little lower **BLEU** score compared to Boudin and Morin (2013) for two main reasons (1) We tried to balance between gammaticality and informativity (2) **BLEU** works well on surface level lexical overlap.

5 CONCLUSION AND FUTURE WORK

In this work, we designed a new abstractive compression generation model which can jointly perform sentence fusion and paraphrasing using skipgram word embedding model. In future, we will apply our model to abstractive multidocument summarization system where documents usually contain related set of sentences.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their useful comments. This research is supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and the University of Lethbridge.

REFERENCES

[1] Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document Abstractive Summarization Using ILP Based Multi-sentence Compression. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 1208–1214.

[2] Regina Barzilay and Kathleen R. McKeown. 2005. Sentence Fusion for Multidocument News Summarization. *Comput. Linguist.* 31, 3 (Sept. 2005), 297–328.

[3] Florian Boudin and Emmanuel Morin. 2013. Keyphrase Extraction for N-best Reranking in Multi-Sentence Compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, 298–305.

[4] Katja Filippova. 2010. Multi-sentence Compression: Finding Shortest Paths in Word Graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 322–330.

[5] Katja Filippova and Michael Strube. 2008. Sentence Fusion via Dependency Graph Compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 177–185.

[6] Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 302–308.

[7] Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-Efficient Creation of an Accurate Sentence Fusion Corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, 317–320.

[8] Oren Melamud, Omer Levy, and Ido Dagan. 2015. A Simple Word Embedding Model for Lexical Substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Association for Computational Linguistics, Denver, Colorado, 1–7.

[9] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP 2004*, Dekang Lin and Dekai Wu (Eds.). Association for Computational Linguistics, Barcelona, Spain, 404–411.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*. Curran Associates Inc., USA, 3111–3119.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 311–318.

[12] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, 425–430.

[13] Christophe Servan, Alexandre Berard, zied elloumi, Hervé Blanchon, and Laurent Besacier. 2016. Word2Vec vs DBnary: Augmenting METEOR using Vector Representations or Lexical Resources?. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, 1159–1168.

[14] Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond K. Wong, and Fang Chen. 2016. An Efficient Approach for Multi-Sentence Compression. In *Proceedings of The 8th Asian Conference on Machine Learning (Proceedings of Machine Learning Research)*, Vol. 63. PMLR, The University of Waikato, Hamilton, New Zealand, 414–429.

[15] Dung Tran Tuan, Nam Van Chi, and Minh-Quoc Nghiem. 2017. *Multi-sentence Compression Using Word Graph and Integer Linear Programming*. Springer International Publishing, Cham, 367–377.

[16] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics* 4 (2016), 401–415.