

Contextual Networks and Unsupervised Ranking of Sentences

Hao Zhang
Dept. of Computer Science
University of Massachusetts
Lowell, USA
hao_zhang@student.uml.edu

You Zhou
Dept. of Computer Science
University of Massachusetts
Lowell, USA
you_zhou@student.uml.edu

Jie Wang
Dept. of Computer Science
University of Massachusetts
Lowell, USA
wang@cs.uml.edu

Abstract—We construct a contextual network to represent a document with syntactic and semantic relations between word-sentence pairs, based on which we devise an unsupervised algorithm called CNATAR to score sentences, and rank them through a bi-objective 0-1 knapsack maximization problem over topic analysis and sentence scores. We show that CNATAR outperforms the combined ranking of the three human judges provided on the SummBank dataset under both ROUGE and BLEU metrics, which in term significantly outperforms each individual judge’s ranking. Moreover, CNATAR produces so far the highest ROUGE scores over DUC-02, and outperforms previous supervised algorithms on the CNN/DailyMail and NYT datasets. We also compare the performance of CNATAR and the latest supervised neural-network summarization models and compute oracle results.

Index Terms—contextual network, topic analysis, T5 sentence similarity, bi-objective 0-1 knapsack

I. INTRODUCTION

Ranking sentences (or segments of text) for a given article may be used, for example, as an oracle to build a hierarchical-reading tool to allow readers to read the article one layer of sentences at a time in a descending order of significance, as a selection criterion to construct a better search engine, or as a base for constructing a summary.

We present an unsupervised algorithm called CNATAR (Contextual Network And Topic Analysis Rank) to rank sentences for a given article, which works as follows:

Step 1: Construct a contextual network (CN) to represent semantic and syntactic relations between sentences in the article by leveraging dependency trees and contextual embeddings of words to form weighted edges between word-sentence pairs.

Step 2: Devise an unsupervised algorithm called CNR (Contextual Network Rank) to score nodes of the underlying CN using a biased PageRank algorithm w.r.t. the underlying article structure, and then score a sentence by summing up node scores for nodes containing the said sentence with a BM25 normalizer.

Step 3: Carry out topic analysis using Affinity Propagation [1] based on T5 sentence similarity, and rank sentences by approximating a bi-objective 0-1 knapsack maximization problem to select sentences with the largest scores and topic diversity using the Within-Cluster Sum of Square metric and dynamic programming.

We show that CNATAR outperforms the combined ranking of all human judges over the SummBank dataset in all categories under both ROUGE and BLEU measures, and substantially outperforms each judge’s individual ranking. Moreover, CNATAR is efficient with an average running time of about 0.7 seconds for each document in SummBank on a commonplace CPU desktop computer. We also evaluate CNATAR on other datasets for abstractive summaries, including DUC-02, CNN/DailyMail (CNN/DM in short), and NYT. We show that CNATAR outperforms all previous algorithms on DUC-02; and outperforms all previous unsupervised algorithms and the supervised model REFRESH [2] on CNN/DM and NYT trained on these datasets. We then compare performance of CNATAR and the two latest supervised BERT-based models BERTSum [3] and MatchSum [4].

II. RELATED WORK

Early sentence-ranking algorithms typically score sentences in connection to text summarization. Recent unsupervised methods include CP₃ [5], Semantic SentenceRank (SSR) [6], BES (BERT Extractive Summarizer) [7] and PacSum [8].

CP₃ models a document as a bipartite graph between words and sentences and uses Hyperlink-Induced Topic Search [9] to score sentences that maximizes sentence importance, non-redundancy, and coherence.

SSR introduces semantic relations overlooked by early unsupervised algorithms to construct word-level and sentence-level semantic graphs. It uses article-structure-biased (ASB) PageRank to score words and sentences separately, and then combines them to generate the final score for each sentence. SSR ranks sentences based on their final scores and topic diversity through semantic subtopic clustering. In so doing, SSR offers higher ROUGE scores on the DUC-02 dataset than CP₃ and the previous unsupervised algorithms, and significantly outperforms each judge’s individual ranking on the SummBank dataset, but still falls short of the combined ranking of the three judges.

BES clusters sentence embeddings generated by BERT with K-means, and ranks a sentence by the Euclidean distance between the sentence and the centroid of the underlying cluster. PacSum builds a complete graph based on dot products

of sentence embeddings, attempting to capture influence of any two sentences to their respective importance by their relative positions in the document.

Recent supervised methods construct neural-network models to perform sequence scoring/labeling. REFRESH [2] a sentence with a CNN encoder and scores sentences using LSTM that globally optimizes the ROUGE metric with reinforcement learning. BERTSum [3], a fine-tuned BERT embeddings for sentences from the input document, scores sentences with a summarization-specific layer trained on a labeled dataset such as CNN/DM. MatchSum [4], another variant of BERT, produces an embedding for the input document and an embedding of the best summary candidate that is most similar to the document embedding. A summary candidate is formed with a desired number of sentences selected from a number of sentences with high scores produced by other models such as BERTSum. These supervised models are trained on the CNN/DM and NYT datasets, all imposing a small upper bound on the size of an input sequence due to the difficulty on handling a long sequence. BERTSum, for example, imposes an input sequence of upto 512 tokens (about 30 sentences on average) and drops the remaining text after the first 512 tokens. Needing a large labeled dataset to train a supervised neural-network model also imposes a major roadblock for languages without such labeled datasets.

III. CONTEXTUAL NETWORKS

Let D denote a document of m sentences and let $\langle S_1, S_2, \dots, S_m \rangle$ be the original sequences of sentences in D . Compute a dependency tree for each sentence and then replace each pronoun in a sentence with its original mention using a coreference resolution tool. A dependency tree for a sentence [10] is an undirected tree rooted at the main verb, connecting other words according to grammatical relations (see Fig. 1 for an example). Compute a contextual embedding $e(w, S)$

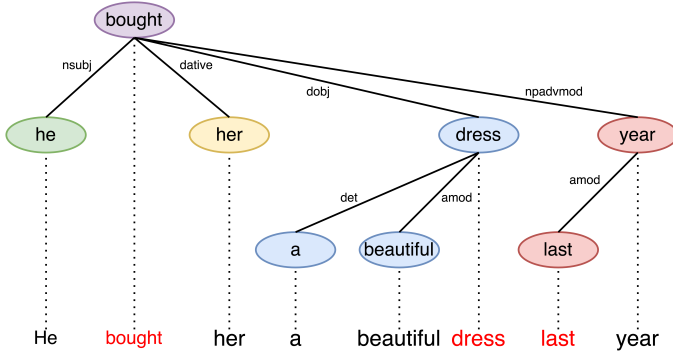


Fig. 1: A dependency tree for “He bought her a beautiful dress last year.”

for with $w \in S$. Next, mark non-content words (stop words) using a stopwords filter. Stop words include determiners, prepositions, postpositions, coordinating conjunctions, copulas, and auxiliary verbs. For each sentence S , let w and w' be two content words in S . If there are stopwords $\sigma_1 \dots, \sigma_r$ such that $w, \sigma_1, \dots, \sigma_r, w'$ forms a path on the dependency tree

T_S , then add a new connection of w and w' in T_S . Finally, remove stop words and replace every content word with its lemma using a lemmatizer.

In what follows, unless otherwise stated, by “word” it means its lemma. For each word w in S , let N_w be the set of direct neighbors of w on T_S . Two words x and y are said to be *syntactically related* if either they are neighbors (i.e., $x \in N_y$) or they share a common third neighbor w (i.e., $x \in N_w$ and $y \in N_w$). This relation captures the structure of subject-verb-object in the same sentence such that any two of these words are syntactically related.

Let $\langle w_1, w_2, \dots, w_n \rangle$ be the original sequence of words in D . By comparing w_i with w_j we mean to compare the words at locations i and j . Denote by (w_i, S_k) the i -th word in the k -th sentence. When i is given, it is straightforward to determine k , which can be expressed with a function $h(i)$. Namely, $w_i \in S_{h(i)}$ for all i . If $i \neq j$, then $(w_i, S_{h(i)})$ and $(w_j, S_{h(j)})$ are different entities even if $w_i = w_j$ and $h(i) = h(j)$.

Construct a weighted, undirected multi-edge graph $G_D = (V_D, E_D)$ with $V_D = \{v_i \mid v_i = (w_i, S_{h(i)}), 1 \leq i \leq n\}$. Let $v_i, v_j \in V_D$ with $i \neq j$. E_D is constructed below: (1) *Semantic edges inside or across sentences*. Connect v_i and v_j if the cosine similarity of $e(v_i)$ and $e(v_j)$ is at least δ (a hyperparameter; it is reasonable to set $\delta = 0.7$). (2) *Syntactic edges inside the same sentence*, namely, $h(i) = h(j)$. Connect v_i and v_j if w_i and w_j are syntactically related on the dependency tree $T_{S_{h(i)}}$. (3) *Syntactic edges across sentences*, namely, $h(i) \neq h(j)$. Connect v_i and v_j if there is a third node $v_q = (w_q, S_{h(q)})$ with $h(q) = h(i)$ and $q \neq i$ such that $w_q = w_j$, v_q and v_j are semantically connected as in (1), and v_q and v_i are syntactically connected as in (2); or the mirror condition (i.e., swap i with j in the above condition) is true. Fig. 2 illustrates this construction. In other words, we

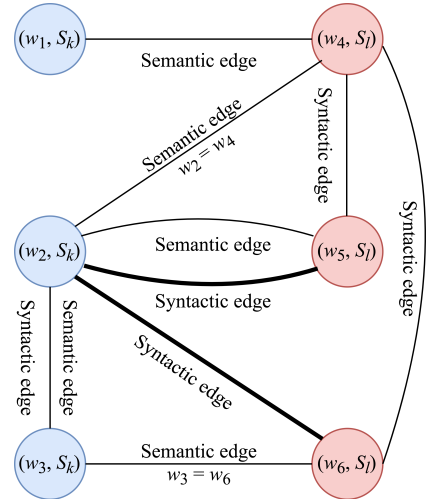


Fig. 2: Two syntactic edges between S_k and S_l .

transform a syntactic relation between w_q and w_i inside $S_{h(i)}$ to a syntactic relation between v_i and v_j if w_q w.r.t. $S_{h(i)}$ is semantically close to w_j w.r.t. $S_{h(j)}$. Note that requiring

$w_q = w_j$ is critical, for it will result in undesirable syntactic relations if $w_q \neq w_j$ (see Remark 1 below).

To construct a contextual network for D , compute edge weights and merge multiple edges. If v_i and v_j are connected by a syntactic edge, let its initial weight be 1, and normalize it by the total number of syntactic edges. If they are connected by a semantic edge, let its initial weight be the cosine similarity of $e(v_i)$ and $e(v_j)$, and normalize it by the summation of all the initial weights of the semantic edges. If v_i and v_j are connected by both a syntactic edge and a semantic edge, then merge the two edges to one edge and let its new weight be the summation of the corresponding syntactic weight and the semantic weight.

Remark 1. To see why we must require $w_q = w_j$ when constructing syntactic edges across sentences consider, for example, the following two sentences: S_1 : A dove with an olive branch in its mouth is a common symbol of world peace. S_2 : Doves, comparing with pigeons, are smaller and slenderer, while pigeons are larger. Fig. 3 depicts the correct syntactic relations by our construction. If, however, we allowed $w_q \neq w_j$, then because the cosine similarity of $e(\text{“dove”}, S_1)$ and $e(\text{“pigeon”}, S_2)$ is greater than the threshold value of $\delta = 0.7$, these two nodes would be connected by a semantic edge, implying that “dove” in S_1 and “large” in S_2 would be syntactically connected, which is undesirable.

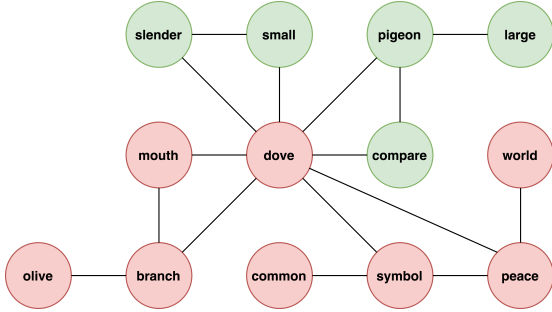


Fig. 3: Syntactic relations via dependency trees on S_1 (red) and S_2 (green) mentioned in Remark 1.

Remark 2. Co-occurrences of words are previously used to capture syntactic relations between words, where two words are related if they co-occur in a small window of successive words. However, this method may falsely relate unrelated words and miss related words. For example, if two adjacent words in the same sentence fall in different sub-trees of its dependency tree, then they are unrelated from the syntactic point of view, but they could be made related because they co-occur. Co-occurrence also fails to capture related words that do not co-occur within a small window. Fig. 4 depicts the syntactic relations of words in the above sentences S_1 and S_2 with a window size of 3, which includes undesirable syntactic edges between “pigeon” and “small”, “pigeon” and “slender”, and “mouth” and “symbol”; yet misses desirable syntactic edges between “dove” and “small”, “dove” and “slender”, “dove” and “peace”, among other things. Our construction of

syntactically related words through dependency trees resolve these issues.

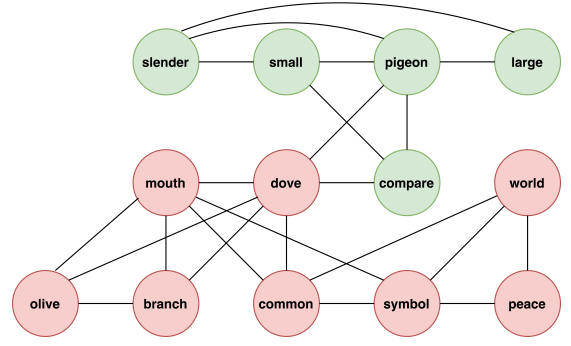


Fig. 4: Syntactic relations through co-occurrences with a window size of 3 between words in S_1 and S_2 in Fig. 3.

IV. SENTENCE RANKING

Article structures also play a role in ranking sentences [6], which may be classified into four types based on locations where words tend to be more important: (1) *Rectangle*. Words are of the same importance in any part of the article. Narrative articles are typically of this type. (2) *Inverted pyramid*. Words toward the beginning of the article tend to be more important. News articles are typically of this type. (3) *Pyramid*. Words toward the end of the article tend to be more important. Argumentative articles are typically of this type. (4) *Hourglass*. Words toward the beginning and the end of the article tend to be more important. Research papers are typically of this type.

Let $LW(i) > 0$ denote the location weight of the i -th word (to be constructed later) with $\sum_{i=1}^n LW(i) = 1$. CNR computes the score of node v_i over the contextual network, denoted by $\text{score}(v_i)$, using the following article-structure-biased (ASB) PageRank:

$$\text{score}(v_i) = 0.85M(v_i) + 0.15LW(i), \text{ where}$$

$$M(v_i) = \sum_{v_j \in \text{Adj}(v_i)} \frac{wt(v_i, v_j) \cdot \text{score}(v_j)}{\sum_{v_k \in \text{Adj}(v_j)} wt(v_j, v_k)},$$

and $wt(u, v)$ is the edge weight of (u, v) . It then scores a sentence S_k by summing up the scores of all the nodes v_i with $h(i) = k$ and normalizing the sum by a BM25 normalizer:

$$\text{score}(S_k) = \frac{\sum_{i:h(i)=k} \text{score}(v_i)}{1 - \beta + \beta \left(\frac{|S_k|}{\text{avsl}} \right)},$$

where $|S_k|$ is the number of words contained in S_k , $\text{avsl} = \sum_{j=1}^m |S_j|/m$ is the average sentence length of the document, and $\beta \in [0, 1]$ is a hyperparameter for the purpose of penalizing sentences that are longer than average and rewarding sentences that are shorter than average. Since the ratio of a sentence length over the average sentence length for a given document is often larger than 2 or smaller than 1/2, an appropriate value of β should be near the first quadrant and we choose $\beta = 0.2$.

Next, we define $LW(i)$ so that it does not abruptly change weight from location i to location $i+1$. For the rectangle structure we simply use a uniform distribution with $LW(i) = 1/n$. For the inverted pyramid structure, we use a slow decreasing quadratic curve to assign location weight for the i -th word by

$$LW(i) = \frac{6(\gamma-1)(i-n)^2}{(n-1)n(2n\gamma-n-\gamma)} + \frac{a(n-1)^2}{\gamma-1},$$

where $\gamma = LW(1)/LW(n)$ is a hyperparameter (e.g., let $\gamma = 5$). The pyramid structure is a mirror image of the inverted pyramid, where the location weight of the i -th word equals the weight for the $(n-i+1)$ -th word in the inverted pyramid structure. For the hourglass structure, we again use a quadratic curve defined by $LW(i) = ((i-n/2)^2 + 1) / \sum_{i=1}^n ((i-n/2)^2 + 1)$, with the minimum value in the middle of 1 and n .

Sentence ranking should reflect the topics covered by the article. A topic clustering algorithm partitions sentences into topic clusters based on a sentence similarity measure. Let $F(D')$ denote the distribution of topic covered by a subset $D' \subseteq D$, and L the maximum $|D'|$ allowed. The sentence ranking problem can be modeled as the following bi-objective 0-1 knapsack maximization problem:

$$\begin{aligned} & \text{Maximize } \sum_{k=1}^m \text{score}(S_k) \cdot x_k \text{ and } F(\{S_k \mid x_k = 1\}), \\ & \text{subject to } \sum_{k=1}^m x_k = L \text{ and } x_k \in \{0, 1\}, \end{aligned}$$

where $x_k = 1$ if S_k is selected, and 0 otherwise. A ranking of sentences can be achieved by starting L from 1, incremented by 1 each time, until $L = |D| - 1$.

CNATAR approximates the bi-objective 0-1 knapsack problem as follows: Suppose that D is partitioned into K topic clusters of sentences D_1, \dots, D_K . Define a topic diversity function F by dividing L into K numbers $L_j = \lfloor (W_j / \sum_{\ell=1}^K W_\ell) L \rfloor$, where W_j is the Within-Cluster Sum of Square (WCSS) [11] for cluster D_j , which is the squared average distance of all the points within D_j to the cluster centroid. Thus, $\sum_{j=1}^K L_j \leq L$. Divide the bi-objective 0-1 knapsack into K 0-1 knapsack problems over each D_j with length bound L_j for $1 \leq j \leq K$. That is, maximize $\sum_{S_k \in D_j} \text{score}(S_k) \cdot x_k$, subject to $\sum_{S_k \in D_j} x_k = L_j$ and $x_k \in \{0, 1\}$, where the L_j sentences in D_j with the highest scores form the maximum solution. Rank sentences in all solutions according to their scores. Let $L' = \sum_{j=1}^K L_j$. If $L' < L$, then select a remaining sentence with the highest score, rank it after the selected sentences, and increase L' by 1. Repeat until $L' = L$.

Remark 3. Sometimes we may need to select sentences such that the total number of words contained in them do not exceed a certain limit L'_j . The constraint of the 0-1 knapsack becomes $\sum_{S_k \in D_j} x_k \cdot |S_k| \leq L'_j$. Using dynamic programming we can obtain a maximum solution to this version of the j -th 0-1 Knapsack problem in $O(|D_j|L'_j)$ time, which is feasible in practice since $|D_j|$ and L'_j would be small. We will need

to use this version of 0-1 knapsack later when we deal with the DUC-02 dataset.

V. IMPLEMENTATION AND EVALUATION

We preprocess documents with spaCy [12] to split the text into sentences, and resolve coreference within a sentence using the NeuralCoref pipeline [13]. We then generate, for each sentence, a dependency tree with spaCy, and generate contextual embedding using BERT-Large [14] for each word in the sentence. Next, we identify stopwords with spaCy's stopword list and replace each content word with its lemma using spaCy's lemmatizer. To generate a contextual embedding for each word w.r.t. to a sentence, we sum up, the corresponding vector representations in the last 4 layers of BERT-Large to form a contextual embedding of the word, to take the advantage of more syntactic information at the lower layers more semantic information at the higher layers [15]. Finally, we use Affinity Propagation (AP) [1], an exemplar-based clustering algorithm, to cluster sentences using a pretrained T5 similarity [16] to compute sentence similarities. T5 similarity takes two sentences as input and returns a similarity score between 1 and 5. AP dynamically determines the number of topic clusters. The major components and dataflows of CNATAR are shown in Fig. 5.

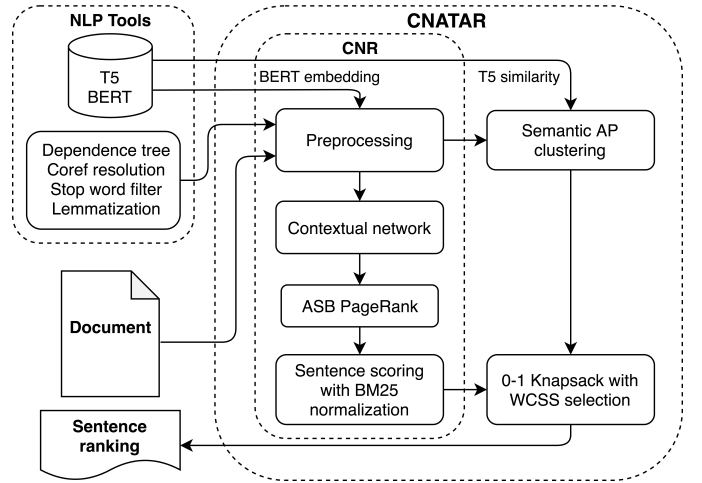


Fig. 5: CNATAR components and dataflows

Datasets. SummBank [17] is the most suitable dataset for evaluating sentence-ranking algorithms. Three human judges rank sentences for each of the 200 news articles written in English individually in categories of top 5%, and 10% to 90% with an increment of 10%. A combined sentence ranking of all judges, denoted by CMB-HR, is also provided on each article. DUC-02, CNN/DM, and NYT are other datasets for evaluating single-document summarization algorithms, consisting of one or more human-written abstractive summaries for each article as the gold standard. Each summary in DUC-02 consists of upto 100 words, while each summary consists of an average of 3 sentences in CNN, and 4 sentences in DM and NYT. These datasets, although not ideal for evaluating sentence

TABLE I: Comparison of CMB-HR, CNATAR, CNR, SSR, PacSum, and BES against all judges over SummBank

	5%				10%				20%				30%				40%			
	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2
Oracle	62.34	55.56	100.00	100.00	66.79	59.62	100.00	100.00	75.34	68.86	100.00	100.00	80.33	74.75	100.00	100.00	84.45	80.05	100.00	100.00
CMB-HR	51.60	43.50	86.09	80.22	57.53	50.03	91.80	89.32	69.08	63.14	94.75	93.84	75.37	70.42	96.82	96.69	82.03	78.15	98.03	97.98
CNATAR	52.18	44.22	86.54	81.90	58.24	50.86	92.06	89.51	69.36	63.88	94.93	94.21	76.04	71.19	97.01	96.97	82.95	79.02	98.11	98.03
CNR	51.38	41.24	85.09	79.22	57.27	49.74	87.46	83.42	68.51	61.22	91.92	88.86	74.71	69.29	93.63	92.85	80.37	76.02	94.57	94.07
SSR	51.66	43.32	85.29	79.55	57.63	50.18	88.16	83.93	68.53	61.24	92.33	89.12	74.92	69.82	94.06	93.38	80.40	76.22	94.83	94.22
PacSum	48.33	37.59	80.25	72.49	55.77	47.73	84.98	78.42	65.03	57.50	88.21	83.95	70.91	64.52	90.11	88.04	74.66	68.97	91.71	90.01
BES	47.23	36.35	78.72	69.98	53.98	44.46	81.54	73.93	62.70	54.06	85.84	79.06	68.47	59.94	88.79	85.51	72.80	65.54	91.02	88.00
	50%				60%				70%				80%				90%			
	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2
Oracle	87.61	84.20	100.00	100.00	90.88	88.34	100.00	100.00	93.65	91.95	100.00	100.00	96.01	95.06	100.00	100.00	98.55	98.27	100.00	100.00
CMB-HR	84.85	81.73	98.34	98.29	88.34	85.90	98.81	98.69	91.28	89.55	99.17	99.14	94.02	92.59	99.57	99.50	96.78	96.30	99.81	99.74
CNATAR	85.47	82.72	98.47	98.51	88.53	86.14	98.86	98.74	91.52	89.61	99.22	99.27	94.36	92.73	99.57	99.61	96.81	96.49	99.83	99.79
CNR	84.21	79.89	95.86	95.14	87.51	83.85	97.36	97.07	91.23	88.65	97.69	97.18	93.79	92.53	98.18	97.96	96.41	95.86	99.06	98.96
SSR	84.21	79.89	95.86	95.14	87.51	83.85	97.36	97.07	91.23	88.65	97.69	97.18	93.79	92.53	98.18	97.96	96.41	95.86	99.06	98.96
PacSum	77.75	73.31	93.03	91.94	82.35	78.96	94.88	93.96	86.96	84.41	96.29	95.44	91.04	89.50	97.20	96.99	96.71	96.28	99.34	99.01
BES	75.48	69.58	91.27	88.91	80.44	76.91	93.80	92.46	85.83	83.04	95.44	94.89	90.01	88.07	95.94	95.34	95.75	94.75	97.27	96.82

ranking, are used to compare with the latest summarization algorithms in the last 5 years. We follow the standard split of training and evaluating [18] of CNN/DM on supervised algorithms, and use scripts supplied by [19] to obtain non-anonymized version of data. The XSum [20] dataset provides a one-sentence abstractive summary for each article, and so is inappropriate for evaluating sentence-ranking algorithms.

All of these datasets are news articles and so the location weight function for the inverted pyramid structure is applied.

Comparison on SummBank. We compare machine rankings with CMB-HR against each individual ranking as reference and average the ROUGE [21] and BLUE [22] scores over all documents. Both CMB-HR and SSR outperform each individual judge’s ranking using the other two judges’ ranking as reference [6]. A full-range comparison is shown in Table I against all judges under common measures of ROUGE- n (R- n) and BLEU- n (B- n), where $n = 1, 2$. The highest score under each category is shown in boldface. It can be seen that under all categories, CNATAR outperforms CMB-HR and substantially outperforms SSR, PacSum, and BES. SSR slightly outperforms CNR. The oracle results are computed by choosing, for each article and under each percentage category, an individual judge’s selection of sentences that has the highest R-1 score against all three judge’s selections. Because one judge’s selection is always selected, the corresponding BLEU score is 100%. We carry out the same experiments on two 32G-RAM computers, one with an Intel Core i7-8700K CPU and the other an NVIDIA RTX 2080 Ti GPU. The average running time of CNATAR on each document is 0.73 seconds on the CPU machine, and 0.6 seconds on the GPU machine.

Comparison on DUC-02. Table II depicts the comparison results of the algorithms published in the last five years on the DUC-02 dataset, where each of the algorithms extracts sentences of the highest ranks with a total length bounded by 100 words. Among these algorithms, CNN-W2V [23] is a supervised algorithm. In addition, we also provide oracle results by selecting a subset of sentences for each document that maximizes the ROUGE score w.r.t. the benchmark summaries

except the 6 articles with 78 sentences or more. For these 6 articles we use an approximation to avoid combinatorial explosion by selecting the first sentence with the highest R-1 score, then the next to the already-selected sentences with the highest R-1 score until the total number of words exceeds 100. To the best of our knowledge, no oracle results on DUC-02 were published before. It can be seen that CNATAR outper-

TABLE II: Comparison results (%) on DUC-02, where the italic numbers are extracted from the corresponding papers.

Methods	R-1	R-2	R-SU4
Oracle	52.0	29.1	29.2
CNATAR	49.4	25.6	26.7
CNR	49.2	24.8	26.1
SSR	49.3	25.1	26.5
CP ₃	49.0	24.7	25.8
PacSum	48.7	23.3	25.3
CNN-W2V	48.6	22.0	—
BES	48.5	23.3	25.4

forms all previous algorithms, supervised and unsupervised. The three latest supervised models trained on CNN/DM and NYT only produce 3 to 4 sentences for a given document, and so perform poorly on DUC-02, where each summary typically contains more than 4 sentences.

Comparison on CNN/DM and NYT. Table III shows the comparison results of CNATAR, REFRESH [2], BERTSum [3], and MatchSum [4] on CNN/DM, NYT, and SummBank-4, a subset of 156 articles in SummBank that provide the top 4 sentences (the rest of the articles do not provide top 4 sentences because SummBank only rank sentences on certain percentages). All models output 4 sentences. Recall that MatchSum suffers from a combinatorial blowup, to make it feasible to train, we select sentence candidates using the top 5 most important sentences on CNN/DM, top 6 sentences on NYT, and top 9 sentences on SummBank-4. It can be seen that CNATAR outperforms the unsupervised PacSum and the supervised REFRESH while MatchSum achieves the highest ROUGE scores, where R-L stands for ROUGE-L.

On SummBank-4, CNATAR outperforms all the supervised models by a large margin, even if MatchSum has tried all possible candidate outputs for each article in SummBank-4. The oracle results are computed by selecting the first sentence with the highest R-1 score, then select the next sentence to the already selected sentences with the highest R-1 score.

TABLE III: Comparison results (%), where the numbers in italic are taken from the corresponding papers.

Method	CNN/DM			NYT			SummBank-4		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Oracle	53.1	30.2	50.4	60.7	40.8	57.5	82.6	77.2	79.4
SSR	40.9	18.1	37.2	42.1	21.9	37.8	74.4	69.3	74.1
PacSum	40.7	17.8	36.9	41.4	21.6	37.5	69.0	63.1	68.6
CNATAR	41.6	18.9	38.0	42.9	22.7	38.5	77.3	73.4	76.9
REFRESH	41.0	18.8	37.7	42.6	22.3	38.2	71.6	67.3	71.1
BERTSum	43.2	20.2	39.6	45.8	26.0	42.3	72.3	68.8	71.9
MatchSum	44.4	20.8	40.5	46.2	26.2	42.5	73.7	69.3	73.4
CMB-HR	-	-	-	-	-	-	76.6	72.7	76.4

Ablation study. We show that, over SummBank, each mechanism in CNATAR is necessary for achieving its over-all performance. In particular, contextual networks, location weights, and topic-cluster-wise 0-1 knapsack are the most significant components. Table IV depicts the numerical results, where V-NSR denotes a variant of CNATAR without con-

TABLE IV: Results from ablation study.

	10%				40%				70%			
	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2	R-1	R-2	B-1	B-2
CNATAR	58.24	50.86	92.06	89.51	82.95	79.02	98.11	98.03	91.52	89.61	99.22	99.27
V-NSR	56.41	48.68	85.17	80.73	75.36	70.25	92.49	91.08	88.03	85.92	96.57	95.81
V-NLW	57.44	49.94	87.88	83.51	80.28	76.06	94.55	94.08	91.17	88.47	97.42	97.03
V-NBM25	58.07	50.58	91.10	88.95	82.43	77.89	97.33	96.98	91.29	89.32	98.94	98.76
V-BERT	58.20	50.83	92.01	89.45	82.89	78.94	98.04	97.95	91.44	89.49	99.17	99.18
V-WMD	58.13	50.79	91.75	89.32	82.64	77.96	97.67	97.51	91.43	89.46	99.14	99.01
V-RR	57.34	49.85	87.57	83.49	80.38	76.08	94.64	94.13	91.24	88.64	97.70	97.20
V-CS	57.48	50.05	87.96	83.57	80.41	76.18	94.70	94.22	91.28	88.93	98.26	97.34

textual networks but using co-occurrences to capture weaker syntactic relations between words as in SSR [6], V-NLW denotes a variant without location weight functions, and V-NBM25 a variant that replaces the use of a BM25 normalizer with the standard normalizer of sentence length. Moreover, V-BERT and V-WMD denote two variants that replace the T5 similarities with, respectively, the cosine similarity of BERT embedding, and similarities based on Word Mover’s Distance [24] as in SSR. Finally, V-RR and V-CS denote two variants that replace the cluster-wise 0-1 knapsack with, respectively, round-robin selections from clusters as in SSR and proportional selections based on cluster size.

VI. CONCLUSIONS AND FINAL REMARKS

CNATAR ranks sentences based on context networks and topic analysis, and achieves the state-of-the-art results. Our construction of contextual networks, however, only takes advantage of a few recent NLP tools. More NLP tools may be leveraged, including part-of-speech tags, role labeling, and sentiment analysis. Using these extra language features,

it is expected a more appropriate weight can be computed when merging two edges in constructing a contextual network, instead of assigning an equal weight as in the current construction. Topic diversity also plays an important role in ranking sentences, and so it would be interesting to investigate a better mathematical formulation for the diversity function and explore other topic clustering algorithms.

REFERENCES

- [1] D. Dueck, “Affinity propagation: clustering data by passing messages,” Ph.D. dissertation, University of Toronto, 2009.
- [2] S. Narayan, S. B. Cohen, and M. Lapata, “Ranking sentences for extractive summarization with reinforcement learning,” in *Proc. of NACCL 2018, Vol. 1*, pp. 1747–1759.
- [3] Y. Liu, “Fine-tune BERT for extractive summarization,” *arXiv preprint arXiv:1903.10318*, 2019.
- [4] M. Zhong, P. Liu, Y. Chen, D. Wang, X. Qiu, and X. Huang, “Extractive summarization as text matching,” *arXiv preprint arXiv:2004.08795*, 2020.
- [5] D. Parveen, M. Mesgar, and M. Strube, “Generating coherent summaries of scientific articles using coherence patterns,” in *Proc. of EMNLP 2016*, pp. 772–783.
- [6] H. Zhang and J. Wang, “An unsupervised semantic sentence ranking scheme for text documents,” *Integrated Computer-Aided Engineering*, no. 28, pp. 17–33, 2021.
- [7] D. Miller, “Leveraging BERT for extractive text summarization on lectures,” *arXiv preprint arXiv:1906.04165*, 2019.
- [8] H. Zheng and M. Lapata, “Sentence centrality revisited for unsupervised summarization,” *arXiv preprint arXiv:1906.03508*, 2019.
- [9] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” in *Proc. of SODA 1998*, pp. 668–677.
- [10] R. A. Hudson, *Word grammar*. Blackwell Oxford, 1984.
- [11] J. A. Hartigan and M. A. Wong, “Ak-means clustering algorithm,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
- [12] M. Honnibal and I. Montani, “spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing,”
- [13] T. Wolf, “State-of-the-art neural coreference resolution for chatbots,” *Blog post*, 2017.
- [14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [15] G. Jawahar, B. Sagot, and D. Seddah, “What does BERT learn about the structure of language?” in *Proc. of ACL 2019*.
- [16] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv preprint arXiv:1910.10683*, 2019.
- [17] D. Radev *et al.*, “Summbank 1.0 ldc2003t16. web download,” Philadelphia: Linguistic Data Consortium, 2003.
- [18] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, “Abstractive text summarization using sequence-to-sequence rnns and beyond,” *arXiv preprint arXiv:1602.06023*, 2016.
- [19] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” *arXiv preprint arXiv:1704.04368*, 2017.
- [20] S. Narayan, S. B. Cohen, and M. Lapata, “Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization,” *arXiv preprint arXiv:1808.08745*, 2018.
- [21] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81.
- [22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *Proc. of ACL 2002*, pp. 311–318.
- [23] Y. Zhang, M. J. Er, and M. Pratama, “Extractive document summarization based on convolutional neural networks,” in *Proc. of IECON 2016*, pp. 918–922.
- [24] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, “From word embeddings to document distances,” in *Proc. of ICML 2015*, pp. 957–966.