D4

Anna Batra, Sam Briggs, Junyin Chen, Hilly Steinmetz
Department of Linguistics, University of Washington
{batraa, briggs3, junyinc, hsteinm}@uw.edu

Abstract

This deliverable contains a skeleton of the paper and a list of team members. This is an example citation (Jurafsky and Martin, 2022). According to Jurafsky and Martin (2022), this is another example citation.

1 Introduction

2 Engines

3 System Overview

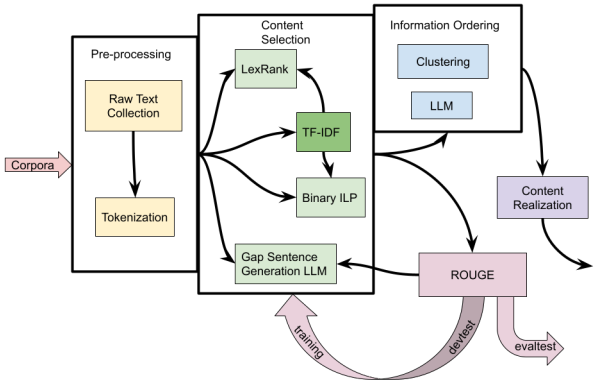

Figure 1: System Architecture

4 Approach

4.1 Data Pre-processing

4.1.1 Collecting raw text

We used the data from the TAC 2009, 2010, 2011 Shared Task data. To process the given XML files to retrieve the set of document names in docSetA, we used xml.etree.ElementTree. We then figured out which corpus data path matched each document in docSetA.

In order to read in these AQUAINT and AQUIANT2 files that are organized differently, we used lxml.etree as this can parse non-XML compliant files. We found there are three different organization methods that were used, and we made sure to process each kind uniquely. From these files, we parsed the the headline, time, and raw text paragraphs.

4.1.2 Tokenization

After getting the raw text, we used two different tokenization methods: spaCy and NLTK.

From spaCy 2.0 we used the English model tokenizer “en_core_web_sm". We decided on spaCy 2.0 since it fits the python version (Python 3.6) on Patas (the virtual machine we used). We decided not to tokenize the raw text into sentences first. Instead, we just ran a word tokenizer on the raw text. The spaCy tokenizer utilizes a transformer under the hood. We then realized that there was a sentence tokenizer, but since this tokenization method uses transformers on the CPU, it actually takes a really long time to run. Therefore, we decided to leave it as it is and turn to a rule-based tokenizer explained in the next method.

For NLTK, we used the English model tokenizer “tokenizers/punkt/english.pickle". We first ran a sentence tokenizer, and then for each tokenized sentence, we then ran a word tokenizer. This gave us a list of sentences, where each sentence contains tokenized words. The NLTK tokenizer is significantly faster than spaCy’s tokenizer and gives us the correct output we need per D2.

4.1.3 Sentence Embeddings

We wanted to investigate whether semantic information can improve the performance of various algorithms used generate summaries. We compared the performance of the LexRank and topic clustering algorithms by utilizing TF-IDF, Word2Vec (Mikolov et al., 2013), and DistilBERT (Sanh et al., 2019) to generate document vectors. For the pretrained word2vec model, we used the continuous

bag of words (CBOW) model trained on the Google News corpus.

Creating vectors from each document involve different pooling methods. The TF-IDF vector was constructed by using either calculating TF-IDF for each document, with IDF values obtain either from the document set or data set. Word2vec vectors were created by averaging each word vector (obtained from the pretrained model). DistilBERT vectors were created by obtaining the final hidden states of the pretrained model, and averaging these final states.

4.2 Content Selection

4.2.1 TF-IDF

To obtain the importance of an n-gram in a given document set, we used the "term-frequency, inverse document frequency ($tf \cdot idf$) metric. To calculate, we used a few different formulas with a few different parameters. The parameters were as follows:

- Document_level: Whether to treat each sentence as a document, or the entire docset as a document.
- N-gram: Whether to treat each term as a unigram, bigram, or trigram. Padding was incorporated here for both start of sentence and end of sentence tokens, using nltk.util.ngrams.
- Eliminate_punctuation: Whether to include punctuation or not.
- Casing: Whether to lowercase all letters, or maintain original capitalization
- log: Whether to use logged equations or not (see equations below)
  - log_base: If logged equations are used, what base to use
- smoothing: Whether to smooth tf and idf, or not
  - tf_delta: Which δ_1 to use or tf fif smoothing
  - idf_delta: What δ_2 to use for idf if smoothing

Given all the training data D, a n-gram t, and a document set $d \in D$, we calculated the term-frequency, inverse document frequency ($tf \cdot idf$) as follows:

One difference than normal tf-idf is that we used tf at a different level of document than idf. For LexRank we used a sentence level document, while allowing the idf to span over the entire dataset. For ILP, we used a docset level document, while allowing the idf to span over the entire dataset else. This will help to make frequent words insignificant and help located the more important words for the sentence/docset.

Using the logarithmically scaled, add δ_1 smoothed tf, and we used an add δ_2 smoothed idf to weight each term in the document set (Seki, 2003).

We let:

$$f_{t,d} = |\{t \mid t \in d, d \in D\}|$$
$$n_t = |\{d \mid t \in d, d \in D\}|$$

Figure 2: If logged, we calculate as follows:

$$\mathrm{tf} \cdot \mathrm{idf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, d, D)$$
$$\mathrm{tf}(t, d) = \log(\delta_1 + f_{t,d})$$
$$\mathrm{idf}(t, D) = \delta_2 + \log\left(\frac{N}{\delta_2 + n_t}\right)$$

Figure 3: If not logged, we calculate as follows:

$$\mathrm{tf} \cdot \mathrm{idf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, d, D)$$
$$\mathrm{tf}(t, d) = \delta_1 + f_{t,d}$$
$$\mathrm{idf}(t, D) = \delta_2 + \frac{N}{\delta_2 + n_t}$$

If not smoothed, δ_1 and δ_2 effectively become 0.

4.2.2 Binary Linear Programming

We used Luo et al. (2018) as a guide to design our summarizer of a document set. For notation, we use y_j for sentence j and z_i for concept i in the document set. We use $A_{i,j}$ to denote the indicator function $\mathbb{1}_{z_i \subseteq y_j}$, i.e. $A_{i,j}$ = 1 if concept z_i appears in sentence y_j, 0 otherwise. We use the weight w_i where weight i is the corresponding weight for "concept" i. We also have a maximum term summary length L.

We then formulated the optimization problem as follows:

$$\begin{aligned}
\text{maxmimize} \quad & \sum_i w_i z_i \\
\text{Subject To} \quad & \sum_j A_{i,j} y_j \geq z_i \\
& A_{i,j} y_j \leq z_i \\
& \sum_j l_j y_j \geq L \\
& y_j \in \{0, 1\} \\
& z_i \in \{0, 1\}
\end{aligned}$$

For each "concept" z_i, we tested unigrams, bigrams, and trigrams. For the corresponding weight w_i for each concept, we used the tf-idf score of the unigram z_i as calculated in 4.2.1.

We found that the best combination of hyperparameters to pass in to our calculation of tf-idf was to use calculate tf-idf for unigrams, using a logged tf-idf with δ_1 close to 0 (in our case $\delta_1 = 0.01$), a δ_2 close to 1 (our best case uses $\delta_2 = 0.7$). We also found that eliminating punctuation and lowercasing all tokens yielded the best results. We also found that removing sentences with less than around 20 tokens (after being tokenized by nltk.word_tokenize) yielded the best Rouge scores.

4.2.3 LexRank

We also implemented the LexRank algorithm described in Erkan and Radev (2004). LexRank is an adaption of the PageRank algorithm (Page et al., 1999), and was proposed as an alternative to centroid-based approaches. LexRank leverages relationships between documents by creating a weighted graph that connects sentences. Relating the sentences to one another has the advantage of (1) dampening the effect of high IDF scores of rare words (when using TF-IDF vectors) and (2) formalizing a preference for more informative (or more connected) sentences .

The LexRank algorithm treats each sentence as a document. It compares sentence vectors to construct a weighted graph of the relationships between sentences in a document set. Erkan and Radev (2004) obtains sentence vectors using TF-IDF (without smoothing); however, sentence vectors can be obtained using a number of methods (see 4.1.3).

Sentences are compared to one another are related to one another using the cosine similarity measure:

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||_2 \times ||\mathbf{y}||_2}$$

A similarity matrix can then be constructed by calculating similarity scores across all sentences in the document. Unlike Erkan and Radev (2004), we do not calculate the similarity between instances of the same sentence, since we found these high scores to hurt its performance. We believe the decline in performance might pertain to lowering other similarity scores after the matrix is normalized by rows.

Using this similarity measure, we created a similarity matrix between sentences in the document, which also functioned as a weighted graph. Per Erkan and Radev (2004), values with low similarities scores are discarded and self-connections between nodes. The matrix satisfies the properties of a stochastic matrix, allowing us to use the power method to estimate the eigenvalue of the matrix. We initialize the centrality vector as $\boldsymbol{p} = \frac{1}{N}\boldsymbol{p}$, where N is the number of documents. We then apply the following update to $\boldsymbol{p}$:

$$\boldsymbol{p} = [d\boldsymbol{U} + (1 - d)\boldsymbol{B}]^T \boldsymbol{p}$$

where $\boldsymbol{U}$ is a square matrix of size $[N \times N]$ with values equal to $1/N$ and $\boldsymbol{B}$ is the adjacency matrix of the graph.

After finding $\boldsymbol{p}$, the eigenvalue is used to rank the sentences by index. Sentences with the top k values in ranking are extracted and added to the summary text until the summary reaches the maximum length. To reduce redundancy, we use the Jaccard similarity measure (provided by NLTK) to calculate each sentence's similarity to sentences already included in the bibliography (Bird and Loper, 2004).

$$D_j(s_i, s_j) = \frac{|\{w|w \in s_i\} \cap \{w|w \in s_i\}|}{|\{w|w \in s_i\} \cup \{w|w \in s_i\}|}$$

Additionally, sentences that are too long are discarded. We discard long sentences because we believe these sentences are too likely to be too information-dense to be useful for constructing a summary.

Experiments were conducted to investigate whether LexRank performed best with TF-IDF, Word2vec, or DistilBERT vectors. A comparison table can be found in AppendiX B.

4.2.4 Gap sentence generation

We used the gap sentence generation method introduced in Zhang et al. (2019). Based on the finding when Zhang et al. were training the Pegasus based model, we will selecting the top m sentences as gap sentences without replacement from a document based on importance score. The importance score is calculated based on the ROUGE score one sentence get comparing to the remaining sentences in one document as in Algorithm 1.

Algorithm 1 Independent sentence selection

1: $D := \{x_i\}_n \leftarrow$ sentences in all*document*
2: $S := \emptyset$
3: $I \leftarrow$ list contains index from 0 to n
4: **for** $j \leftarrow 1$ to n **do**
5: $\quad s_i := rouge(x_i, D \setminus \{x_i\})$
6: $\quad S := S \cup \{s_i\}$
7: $I := sort(I)$ Based on the value in S

To improve the system, when we are processing the training data, we calculate the ROUGE score based on the average ROUGE score when comparing the selected sentence with each of the gold summary. We only calculate the ROUGE score based on the rest of the documents when we are processing the test and validation data.

We only mask the top thirty percent of the sentences as Zhang et al. finds out achieves relatively high performance without sacrifice training efficiency.

4.3 Information Ordering

4.3.1 Topic Clustering

This algorithm tries to order topics in the order that they are most likely to appear in a document. The intuition being that similar sentences form topics, and topics must be ordered in the original documents in a cohesive manner. This algorithm tries to recreate this cohesive topic order. This algorithm is heavily modified from the 'Augmented Ordering Algorithm' presented in ("Barzilay et al., 2002)

Similarity between sentences is determined by the similarity of sentence embeddings. To group topics, we group similar sentences over a whole document set. We calculate sentence as calculated in section 4.1.3 using Word2Vec embeddings for each word in a sentence.

We then use sentence embeddings to create topic clusters to group similar sentences. To group sentences, we used K-means clustering. To run K-means clustering, we used sklearn.cluster.KMeans with 8 clusters, and the following parameters:

```
kmeans = KMeans(
    n_clusters=8, init='k-means++',
    n_init=10, max_iter=300,
    tol=0.0001, verbose=0,
    random_state=None, copy_x=True,
    algorithm='lloyd')
```

To order topic clusters, we used the 'fractional ordering' of the sentences, namely, let d be a document with length n, then the fractional ordering f of sentence i at position i in d is:

$$f(i, d) = \frac{i}{n}$$

For example, the first sentence in the given document is always $\frac{1}{n}$, the second $\frac{2}{n}$, etc. We divide by the number of sentences in the document to try to normalize both short and long documents.

We then ordered each topic cluster t_j with m sentences by their respective average fractional ordering, namely, for each sentence j in the given topic, the average fractional ordering f_{avg} is then:

$$f_{avg}(t_j) = \frac{\sum_j f(j, d)}{m}$$

We then order the sentences in the given summary based on which topic cluster they appear in and the fractional ordering of that topic. In other words, if $f_{avg}(t_j) < f_{avg}(t_i)$, then any sentence that appeared in the summary and in topic cluster t_j would appear before all the sentences that appeared in the summary and in topic cluster t_i.

It is possible that multiple sentences appear in the given summary and the same topic cluster. In this case, we order the sentences by their own fractional ordering. In other words, for sentence i and sentence j in documents d and d' respectively, if $f(i, d) < f(j, d')$ we put sentence i before sentence j in the ordered summary.

In Table, 1 we can see an error analysis of clustering. We can see that the sentences seem to flow a bit better by maintaining the original order of the documents, though the sentences all come from different documents of the same topic.

4.3.2 Zero-shot Learning

We trained a zero-shot learning language model based on the implementation of Reorder-BART

Table 1: Error Analysis for summary ordering using Topic Clustering

docset	Unordered	Ordered
D1001-A	The school wanted to make sure there was enough to eat since students couldn't leave campus for lunch and get back in. So many forms of community, rippling outward from Columbine High and across the planet, have come together since last week's violence that it was difficult to tell. Graham praised the Columbine community for uniting under the pain of a tragedy that could have torn it apart. But Wells said he is more interested in simply trying to have fun and move beyond the tragedy that put his life on hold.	So many forms of community, rippling outward from Columbine High and across the planet, have come together since last week's violence that it was difficult to tell. Graham praised the Columbine community for uniting under the pain of a tragedy that could have torn it apart. The school wanted to make sure there was enough to eat since students couldn't leave campus for lunch and get back in. But Wells said he is more interested in simply trying to have fun and move beyond the tragedy that put his life on hold.
D1002-A	Several of the officers are said to have told associates that they continued firing because Diallo did not fall even after they had unleashed the fusillade. Police officers in criminal trials have often asked for a judge to decide their case, fearing that juries would be unsympathetic. While the trial date would come nearly a year after Diallo's death on the night of Feb. 4, it is not unusual in such high-publicity cases. They are accused of firing 41 times at Amadou Diallo while searching for a rape suspect on Feb. 4.	Several of the officers are said to have told associates that they continued firing because Diallo did not fall even after they had unleashed the fusillade. Police officers in criminal trials have often asked for a judge to decide their case, fearing that juries would be unsympathetic. While the trial date would come nearly a year after Diallo's death on the night of Feb. 4, it is not unusual in such high-publicity cases. They are accused of firing 41 times at Amadou Diallo while searching for a rape suspect on Feb. 4.

(RE-BART) by Chowdhury et al.. RE-BART is a fine-tuned model based on BART by to identify a coherent order for a given set of shuffled sentences. We shuffle the sentences of each input document set based on the Gap sentence generation content selection method without masking. We mark the index of the sentences at the beginning of each sentences. The model takes the sets of shuffled sentences with sentence-specific markers as input and generates a sequence of position markers of the sentences in the ordered text. We trained the model using Huggingface transformer library Wolf et al. (2020). We trained the model using the PyTorch framework with a NVIDIA A100 GPU. We trained the model with batch size of 4 and epoch of 4.

The result, however, is disappointing. The model we trained for our experiment failed to generate a sequence of position makers as output. Instead, the model directly generates the ordered text as the output where each sentences are compressed and no longer contain the same level of information compare to the original sentences. We will further investigate the cause of the undesired output in the next delivery.

4.3.3 ROUGE score ranking

Due to the input size limitation for the majority of the language model, we have to truncate the input text to 1024 tokens. After we mask the important sentence, we then use the ROUGE score ranking calculated for the gap sentence generation to discard sentences that are ranked in the low thirty percent. We keep the ordering of the remainder of the sentences. Discarding unimportant sentences based on ROUGE score helps including more important sentences from multiple documents. When discarding, we calculate the token length for each added sentences and stop when adding additional sentence will cause the token size to exceed 1024 tokens. This make sure we have full sentences for the input sequence.

To improve the system, we experiment on multi-

ple parameters, such as what the percentage of the sentences should we discard. We also calculate the ROUGE score ranking based on either the average score based on the gold summary or based on all of the remaining sentences in all provided documents.

4.4 Content Realization

4.4.1 Large Language Model Training

We trained our model based on "google/pegasus-large" model, as the "google/T5-small" model does not produce complete sentences. We trained the model with batch size of 6 and epoch of 12. We utilized the Huggingface's transformers library for the experiment. The experiments are conducted in PyTorch framework using NVIDIA Tesla A100 GPU.

To improve the system, we tested difference combination of the training arguments, such as different epoch and difference batch sizes.

5 Results

	ROUGE1	ROUGE2
Binary ILP (D4)	0.33697	0.07437
Binary ILP (D3)	0.12085	0.01533
LexRank (D4)	0.21925	0.05966
LexRank (D3)	0.13720	0.02341
GSG LLM (D4)	0.26419	0.05367
GSG LLM (D3)	0.21037	0.06214

Table 2: ROUGE Recall Scores

5.1 Large language model result

Results can be found in Table 3.

6 Discussion

We perform a casual error analysis for the summaries based on docset D1006, which are shown in Table 4.

The improved Binary ILP method makes leaps in achieving a better summary. The old summary seemed to give important facts concerning the FDA and withdrawing Vioxx, but seemed to have a lot of more "unimportant facts" that didn't help the reader get a clear idea of what the article is about. An example of this is mentioning about rewriting abstract conclusions, and the amount of teleconferences that were gone to. The improved summary gives a clear picture of the latest update on Vioxx and the new findings of effects its gives.

The improved LexRank method no longer produce unncessary information such as website address. The improved method successfully mention Vioxx, where the old method did not. However, the improved method still failed to catch one of the core story point, that Vioxx is recalled by the company. The improved method does capture that Vioxx has potential cardiovascular risks. Interestingly, the using TF-IDF vectors resulted in better performance than using word2vec or DistilBERT sentence vectors. A comparison is shown in Table 5 in Appendix B.

The improved GSG LLM methods successfully captured the topic and the time of the event. The improved system also successfully included background information, such as when the FDA had approved the drug, and what the drug is been used for. Lastly, the improved system includes futher discussion about the event, which the gold summary does not have.

These are our future steps for the next step of the project:

- Incorporate TF-IDF as a content selection method.
- Make additional tokenization function to remove phone numbers and city names.
- Find a way to incorporate supervised learning techniques to leverage training data.
- Improve clustering by testing different vector methods and different K's for number of clusters.
- Possibly continue hyper-parameter tuning for ILP, LexRank, and large language model.
- Investigate why the zero-shot information ordering produce undesired output
- Train the language model with other pretrained models to compare the results.

7 Conclusion

The system created for D4 was a substantial improvement over D3. We successfully incorporated improvements to existing systems, and implemented new information ordering algorithms. Still, many of of our methods (such as ILP and LexRank) are unsupervised, meaning that we are leaving training data is going unused–a potential area for future exploration.

ROUGE-on	epoch	Discard	Combine Masking	ROUGE1	ROUGE2
single	6	50%	True	0.21037	0.06214
multi	12	50%	True	0.26419	0.05367
multi	12	30%	True	0.24330	0.04773
multi	12	30%	False	0.24263	0.05343

Table 3: Large langauge model ROUGE Recall Scores with different parameters

8 Appendix A: Workload distribution

8.1 D1 Workload

- Anna Batra set up the Github repository, turned in D1
- Junyin Chen got the team together and set up a communication channel
- Sam Briggs set up the Overleaf file and sent out a when-to-meet to schedule weekly meetings
- Hilly Steinmetz edited the Overleaf file to prepare it for D1.

8.2 D2 Workload

- Anna Batra and Sam Briggs wrote test code to test the file structure of the output docSets, created the outline for the presentation, and updated the report.
- Junyin Chen wrote code for tokenizing documents in the docSets using spaCy, PR reviewed the code to merge with Hilly's, cleaned up the code, and created slides for the pre-processing section.
- Hilly Steinmetz wrote the code for the preprocessing steps before tokenization, such as locating paths for AQUAINT and AQUAINT2 files. Hilly also wrote code for tokenizing documents using NLTK.

8.3 D3 Workload

- Anna Batra and Sam Briggs wrote the code to create a json file to easily access our data for the rest of the project. They also wrote the code for the TF-IDF and Linear Programming content selection methods. The Linear Programming information ordering and content realization was also written by them. They also drew the system architecture.
- Junyin Chen wrote the code to create JSON file writer which contains doc_id, text for summarization, the gold standard summarization based on doc_id for both docsetA and docsetB. The writer help cache the JSON file for easier access. He also wrote the code for Gap sentences generation content selection method, truncate the input text based on ROUGE score for information ordering, and write the code for training a large language model for content realization. He also performs quick error analysis.
- Hilly Steinmetz wrote the code for the LexRank method and debugged issues with the original XML document parser.

Everyone worked on the paper and presentation slides for the parts we explicitly worked on.

8.4 D4 Workload

- Anna Batra and Sam Briggs incorporated ngrams into TF-IDF and helped test the IDF to work over the entire data. They also worked on improving ILP and wrote the clustering information ordering method. They updated the system architecture.
- Junyin Chen wrote and trained zero shot information ordering language model. He also improved pre-processing scripts for data parsing, such as using different ROUGE score and comparing with different documents. He done multiple experiments with information ordering model and summarizing model with multiple parameter combination.
- Hilly Steinmetz created an interface to generate vectors using TF-IDF, Word2vec, or DistilBERT. He also worked on creating a new TF-IDF class that takes various inputs to modify its behavior (e.g., smoothing). Lastly, he implemented the improvements to LexRank.

Everyone worked on the paper and presentation slides for the parts we explicitly worked on.

9 Appendix B: Code repository and additional software and data used in your system

9.1 Source Code

The repository for our project can be found on Github at github.com/LING-575-Summarization/Summarization.

9.2 Packages

We used the following packages for our system:

9.2.1 Pre-processing

- lxml.etree (for processing AQUAINT, AQUAINT2, TAC files)
- xml.etree.ElementTree (for processing docSetA file lists)
- spaCy 2.0 (for word tokenization on paragraphs)
  - English model “en_core_web_sm"
- NLTK
  - English model 'tokenizers/punkt/english.pickle' (for sentence and word tokenization)
  - nltk.util.ngrams (for TF-IDF ngrams)

9.3 Content Selection

- PuLP (for binary ILP)
- rouge-score
- Datasets
- Evaluate
- Pytorch
- Transformers
- Anaconda (for virtual environment)

9.4 Information Ordering

- sci-kit learn (for Topic Clustering)

9.5 LexRank Experiments

Table 5: LexRank Experiments (ROUGE F-Scores)

Vector	R-1	R-2
TF-IDF	0.231	0.060
Word2Vec	0.175	0.036
DistilBERT	0.178	0.042

References

R. "Barzilay, N. Elhadad, and K." McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17(17):35–55.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Somnath Basu Roy Chowdhury, Faeze Brahman, and Snigdha Chaturvedi. 2021. Is everything in order? a simple way to order sentences.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Dan Jurafsky and James H. Martin. 2022. *Speech and Language Processing*, 3rd edition (draft) edition. Online.

Wencan Luo, Fei Liu, Zitao Liu, and Diane Litman. 2018. A novel ilp framework for summarizing content with high lexical variety. *Natural Language Engineering*, 24(6):887–920.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Yohei Seki. 2003. Sentence extraction by tf/idf and position weighting from newspaper articles. *National Institute of Informatics: Proceedings of the Third NTCIR Workshop*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System*

Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *CoRR*, abs/1912.08777.

Table 4: Error analysis for docset D1006

gold	On Sept 30, Merck voluntarily recalled the pain killer Vioxx, used by almost 2 million, after clinical trials for its use in colon cancer showed unacceptable rates of stroke/heart attack. Results corroborated earlier warnings that had not resulted in recalls by the Food and Drug Administration (FDA). As a COX inhibitor, Vioxx was safer for digestive tracts, important for arthritis patients. Merck's advertising campaigns did not clearly warn about side effects. The case highlighted concerns about drug manufacturers' advertising and FDAs role in insuring safety of drugs on the market. Safety of other COX inhibitors is now a concern."
ILP D4	Merck officials said last week its latest research showed an increased risk of heart attack and other cardiovascular complications in patients who took Vioxx for at least 18 months. Heavily advertised as an arthritis drug, Vioxx was pulled from the market last week after its maker said a study showed it doubled the risk of heart attack and stroke. But some doctors say this group of drugs may work in a way that increases the risk of heart problems for some patients, and they point to this latest information as additional reason for concern.
ILP D3	FDA urged to weigh in Then I gave her the facts , " he said . said Lopez-Mendez , medical director of rehabilitation services at Winter Haven Hospital . Graham presented his findings in France Aug. 25 , but already had encountered the resistance from supervisors . The FDA said that Graham decided to revise his abstract conclusion . Over the next three days , Kim and his researchers convened three teleconferences with about 15 outside medical experts to get their advice . " This morning Merck is announcing a voluntary worldwide withdrawal of Vioxx , " Gilmartin began .
LexRank D4	With Vioxx, researchers had been warning about the drug's possible cardiovascular risks since 2000, only a year after it was approved by the FDA . Data from a company study found then that users had four times as many heart attacks and strokes as those who used another painkiller . But the data was not definitive, and Merck, which even critics say is one of the most responsible drug companies, repeatedly reassured the medical and financial communities that Vioxx was safe.
LexRank D3	T̈hat's the tragedy here . " And if courts determine that Merck was negligent, the company will pay a heavy price in compensation. It is in the insurance industry's interest, the FDA's interest and the federal government's interest–because the federal government is a major provider of health insurance–either to require drug companies to conduct such comparative tests or to set up a neutral agency to do so. FDA: http: //www.fda.gov/
GSG LLM D4	Merck recalled Vioxx in September 2004 after a study showed that it doubled the risk of heart attacks and strokes in older people taking it for at least three years. The drug had been approved by the FDA in 1999 for arthritis. Merck had promoted Vioxx as a way to lower blood pressure and cholesterol, but the study showed that it increased the risk of heart attacks and strokes. Merck's decision to withdraw Vioxx from the market raised questions about aggressive marketing of the drug before its long-term safety had been proven.
GSG LLM D3	Vioxx should cast scrutiny on at least two problems inherent in the nation's system for assessing and monitoring drug safety. The number of warning letters has dropped precipitously since the Bush administration took power, from 82 in 2000 to 24 in 2003. The FDA could help solve these problems not only by enforcing its own rules but by requiring doctors and hospitals to report adverse events" when patients use drugs.