Multi-document Summarization for News Articles

Rachel Hantz, Yi-Chien Lin, Yian Wang, Tashi Tsering, Chenxi Li

Department of Linguistics
University of Washington
Seattle, WA USA
{hantz1rk,yichlin,wangyian,tashi0,cl91}@uw.edu

Abstract

Automatic text summarization is the process of selecting the most crucial part from a text to create a shortened version. We focus on multi-document summarization of English language news articles. The input is a set of news articles on the same topic, and the output is a summary. We adopt two methods for content selection (CS): Term Frequency-Inverse Document Frequency (TF-IDF) and Integer Linear Programming (ILP). The outputs of the CS are then reordered using two information ordering (IO) methods. We apply Position Ordering for summaries generated with TF-IDF and Cosine Similarity Ordering for those generated with ILP. Lastly, we apply Name Resolution on our ordered summaries as content realization (CR). We evaluate our system on ROUGE-1 and ROUGE-2 scoring, and the experimental results show that ILP + Cosine Similarity Ordering + Name Resolution is the highest performing system.

1 Introduction

Text summarization has been one of the major tasks in the field of natural language processing (NLP). Automatic text summarization is the process of selecting the most crucial part from a text to create a shortened version. The application of text summarization can be particularly helpful for extracting essential information from large amounts of data.

Our final working systems include three core components for summarization: Content Selection (CS), Information Ordering (IO), and Content Realization (CR). For CS, we implement two methods: Term Frequency-Inverse Document Frequency (TF-IDF) and Integer Linear Programming (ILP). Furthermore, we implement and evaluate Position Ordering, Majority Ordering, Cosine Similarity ordering, and Jaccard Similarity ordering for IO through a human rater survey. For CR, we implement Name Resolution.

The datasets used in our system are the AQUAINT (Graff, 2002) and AQUAINT-2 (Vorhees and Graff, 2008) corpora. The datasets are divided into training, development, and evaluation data. For the development of our current system, we develop our algorithm with the training data and evaluate the system performance on the development data when tuning hyperparameters and selecting components. The evaluation data is held out and will only be used to evaluate the performance of our final best systems.

The rest of this report is organized as follows. Section 2 and Section 3 presents a overview of our system architecture and the details our approaches used for data pre-processing, CS, IO, and CR. Section 4 shows the results of our systems, and section 5 provides an analysis of the results of each component. Lastly, a conclusion is provided in Section 6.

2 System Overview

As can be seen in Figure 1, given a document set of English language news articles in xml format, our method extracts sentences as a summary in four steps: pre-processing, CS, IO, and CR. First, we pre-process each article and store them in groups based on their topic ID. A topic ID references which news event the article depicts. Second, we select sentences with either TF-IDF or ILP. Third, we order sentences selected by TF-IDF with Position Ordering and sentences selected by ILP with Cosine Similarity Ordering based on human rater responses. While we also implement Majority Ordering and Jaccard Similarity ordering for IO, the results of these methods were not favorable to human raters. Lastly, we compare the effects of including CR in the form of Name Resolution.

3 Approach

This section defines how we implemented CS, IO, and CR, alongside intermediate hyperparameter

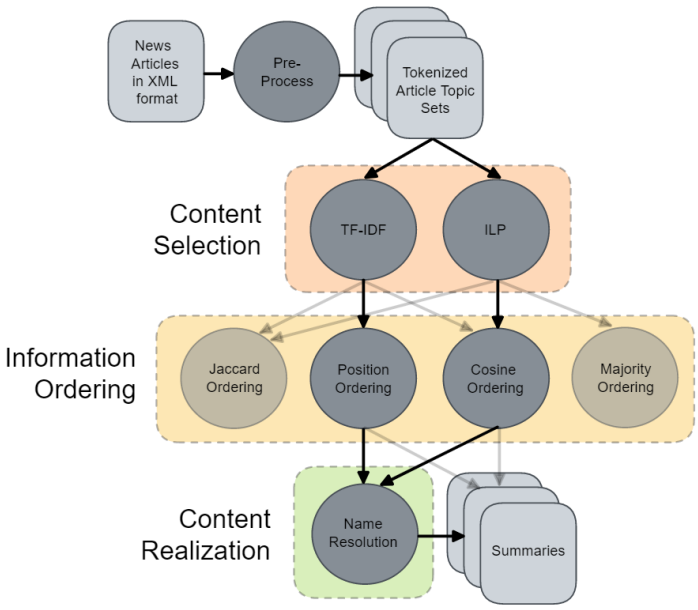

Figure 1: **System Overview.** Components and/or methods not utilized within the best systems are shown greyed-out. Inputs and outputs are shown as rounded rectangles, while system components are shown as circles.

searches. We describe our implementation of the two CS methods, TF-IDF and ILP, and their relevant hyperparamters. Then we explain and compare our implementation of four IO methods. We select two out of the four IO methods to use in our final working system based on a survey of human raters (See Section 4.3 for more details).

3.1 Preprocessing

In order to use the articles that we will summarize, we first needed to pre-process them. We accomplished this with a script containing two steps: processing and tokenization.

Processing takes in the path of input xml file and extracts the document ID. Specifically, the system imported `xml.dom.minidom` to parse the path of input xml file: the system called `getElementsByTagName()` to obtain the elements under docsetA and called `getAttribute()` to obtain each document ID.

Tokenization takes in a document ID and outputs a file of desired format needed for our later tasks. After locating the xml document in corpora, we used `xml.etree.ElementTree` to create a tree for the xml document. On the root node, we obtained the text content of article sections by matching the tag name (e.g., `node.tag == HEADLINE`). For tokenization, we used the sentence tokenizer `sent_tokenize()` from the `NLTK`[1] library to split paragraphs into sentences and the word tokenizer `word_tokenize()` to split each sentence into tokens. The tokenization schema produces output files for each article. Each file starts with headline, date, etc. and has a single tokenized sentence per line. Paragraphs are separated with a single blank line. Files are then grouped according to their topic ID.

3.2 Content Selection

For our multi-document summarization system, we implement two CS methods: TF-IDF and ILP. Descriptions of the two methods are detailed in the following subsections.

3.2.1 Method 1: TF-IDF

The first method we implemented was TF-IDF. The objective of TF-IDF is to weigh the importance of tokens based on their frequency in the foreground and background. A token that occurs frequently in the foreground and less so in the background will have a high TF-IDF score; the higher the score is, the more important a token is.

Our model of TF-IDF consists of two parts: calculating TF-IDF scores of all the tokens and selecting sentences for summary based on the

[1] https://www.nltk.org/

scores. For this particular task, we set all the documents in development test directory as background and the ten articles to be summarized under the same topic as foreground. For TF, we used the raw term frequency, the number of times a token has appeared in the foreground. Note that for the term here, we decided to remove the punctuation and stop words in NLTK library. The formula of IDF is (1), (N) stand for the total number of document and ($df(w)$) is the number of documents containing term (w). L2 normalization (Euclidean normalization) is applied when calculating the TF-IDF score. Note that for the term here, we decided to remove the punctuation and stop words in NLTK library.

$$idf(w) = 1 + ln[(N+1)/(df(w)+1)] \quad (1)$$

We used python library Sicikit-Learn, which provides a simple implementation of the TF-IDF method through the TfidfVectorizer class.[2] We store Topic ID, sentence, word count (excluding stop words), word count (including stop word) and TF-IDF score each as corresponding column and populated a Pandas dataframe. Then we sort TF-IDF score in ascending order for sentences within each document set to select the sentences with with highest score.

$$s_w = \frac{\sum_{i=1}^{n} TFIDF(w_i)}{n} \quad (2)$$

In the second part of sentence selection, the score of each sentence (s_w) is the average TF-IDF score of the tokens (w_i) in the sentence. When including sentences into summary, we explored many hyperparameters to compare the model performance, as discussed in section 3.3.

3.2.2 Method 2: Integer Linear Programming

We also leveraged ILP for our multi-document summarization system, akin to previous implementations (Gillick and Favre, 2009). ILP optimizes a function that is subject to various constraints. This function and these constraints are comprised of decision variables that must be integer valued.

Using the python package PuLP (Mitchell et al., 2011, PuLP) – a python linear programming API – we selected sentences that contained the most important (highest weighted) concepts from a set of multiple documents.[3] Thus, sentences (s_j) and concepts (c_i) were our two binary decision variables.

Concepts could be defined as one of the following: named entities, stemmed unigrams, stemmed bigrams, or stemmed skipgrams. Skipgrams could take any size integer degree and could be at most two tokens. Additionally, all concepts, excluding named entities, could have punctuation and/or stop words removed. Unigrams always had stop words removed. For any bigram or skipgram, there could only be at most one stop word. We utilized NLTK's RegexpTokenizer to remove punctuation, SnowballStemmer (with ignorestopwords=True) to stem words, bigrams, and skipgrams functions. For named entities, we used the spaCy[4] NER pipeline component.

Each concept had a weight equaling the number of articles it appeared in. Only concepts found in at least three documents were retained. Using each concept, we maximized our objective function (3): the weighted (w_i) sum of each concept.

$$\sum_i c_i w_i \quad (3)$$

We had three required constraints: An overall length constraint (4) where the sum of each included sentence's length (l_j) could not be more than 100 whitespace delimited tokens. A coverage constraint (5) where each included concept was found in at least one included sentence. And a second coverage constraint (6) where no concepts that weren't included could be found in any included sentences. o_{ij} denotes the binary presence of c_i in s_j.

$$\sum_j s_j l_j \leq 100 \quad (4)$$

$$\sum_j s_j o_{ij} \geq c_i \; \forall i \quad (5)$$

$$s_j o_{ij} \leq c_i \; \forall i, j \quad (6)$$

Two constraints were optional: A minimum sentence length constraint (7) where each individual included sentence was required to be a minimum sentence length (m). And a theme constraint (8) where sentences could be clustered into themes, and each theme (t_k) could only be included at most one time among the selected sentences. p_{jk} denotes the binary presence of t_k in s_j. We used the python

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] https://coin-or.github.io/pulp/

[4] https://spacy.io/

library Scikit-Learn's TfidfVectorizer class and KMeans[5] clustering algorithm to denote each sentence with one of a set of k themes. Details of the k-means clustering algorithm can be found in Section 3.4.2.

$$l_j s_j \geq m s_j \tag{7}$$

$$\sum_j s_j p_{jk} \leq 1 \; \forall k \tag{8}$$

Using the GLPK solver within PuLP, the ILP model returned which sentences should be selected to maximize the weighted sum of the concepts included. Each included sentence was then printed to a single output file.

3.3 Hyperparameter Search

In this section, we describe the hyperparameters we experimented with for each CS method and how we select the hyperparameters used in our final systems.

3.3.1 TF-IDF

Abundant literature has addressed the selection of features when calculating the sentence score. In our TF-IDF model, we tested four hyperparameters: cosine similarity, stop words and punctuation, sentence length, and finally unigrams and bigrams. When adding sentences to the summary, we exclude sentences that are too similar and sentences that are too short. When calculate sentence length, we also explored the the model performance including and excluding stop words and punctuation. Lastly, we tested out both unigrams and bigrams to compare the performance.

For cosine similarity, .R and Arutchelvan (2022) chose 0.75 as their criterion when eliminating similar sentences before clustering; we used it as our starting point and tested out different cosine similarities with fixed sentence length 10, including stop words and punctuation. The best performance is when cosine similarity is set to 0.25. Next, our test results of stop words and punctuation revealed that when punctuation is excluded, including or excluding stop words yield similar results, both better than other combinations. We decided to include stop words; otherwise the sentences would be long and do not resemble a typical summary. As for sentence length, it is represented in some literature as the ratio of current sentence length to the longest sentence in the article (Neto et al., 2002). We used the raw count of tokens, similar to Teufel and Moens (2002)'s method of penalizing sentences shorter than 12 words. Finally, we found that using bigrams in TF-IDF model, as expected, tend to have higher ROUGE2 score and lower ROUGE1 score. Though, overall, unigrams have higher performance, so we use unigrams for our system.

When looking at the summaries generated by the TF-IDF model, we found that the summaries contain sentences that begin with linking words. These sentences are incomplete in meaning on their own. One way to solve this issue is to also include the preceding sentence, regardless of the sentence score, into the summary (Mohd et al., 2020). However, our experimentation with this proposed solution demonstrated that the inclusion of the previous sentence slightly reduced the rouge score. This result suggests that further investigation is needed to determine the optimal strategy for addressing incomplete sentences in summary generation.

3.3.2 ILP

Our manual hyperparameter search for ILP took place in three steps: (1) Stop Words and Punctuation, (2) Minimum Sentence Length and Concepts, (3) Number of Themes. We describe below why we elect to run ILP under the following configuration: Remove only punctuation, minimum sentence length of 5, degree 1 skipgrams, with no theme constraint.

F1, precision, and recall are highest when stop words are retained and punctuation is removed. This can be seen in Table 1, where bigrams are used as concepts and minimum sentence length is set to 0.

N		-SW+P	+SW-P	-SW-P	+SW+P
1	R	.36314	**.37443**	.36749	.37307
	P	.30704	**.33893**	.3188	.318
	F1	.33249	**.3556**	.34005	.34301
2	R	.07588	**.09304**	.08339	.08521
	P	.08999	**.1029**	.09633	.10019
	F1	.08226	**.09766**	.08908	.09201

Table 1: ROUGE-N scores of the ILP content selection based on stop word (SW) and punctuation (P) removal (bigrams, minimum sentence length = 0). For each type of score, the highest score is denoted in **bold**.

ROUGE-1 F1 is highest when using degree 1 skipgrams as concepts with a minimum sentence

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

length of 5. A minimum sentence length of 5 is also used by Gillick and Favre (2009) in their ILP pre-processing. Degree 2 skipgrams also perform similarly here, but the ROUGE-2 F1 scores are strongest in degree 1 skipgrams. Named entities perform worst in all cases. This can be seen in Table 2 where stop words are retained and punctuation is removed.

N	m	NE	Uni	Bi	SG-1	SG-2
1	0	.3205	.3388	.3556	.3603	.3611
	5	.3216	.3412	.3543	**.3620**	.3608
	8	.3196	.3394	.3545	.3430	.3597
2	0	.0764	.0816	.0977	.0988	.0972
	5	.0761	.0810	.0959	**.0994**	.0974
	8	.0757	.0813	.0964	.0991	.0974

Table 2: ROUGE-N F1 scores of the ILP content selection based on minimum sentence length (m) and concept type (stop words retained, punctuation removed). The highest F1 scores generated with different combinations of sentence length and concept type are denoted in **bold**.

F1 is highest when we do not apply the theme constraint. Although, including the theme constraint does not drastically decrease F1, and should still be considered if sentence redundancy remains an issue in selected sentences. This can be seen in Table 3 where we evaluate the theme constraint when using degree 1 skipgrams.

Concept	k	ROUGE-1	ROUGE-2
SG-1	0	**0.36202**	**0.09936**
	5	0.35208	0.09384
	10	0.35438	0.09464
	15	0.35702	0.09482
	20	0.35737	0.09639
	25	0.35793	0.09621

Table 3: ROUGE-N F1 scores of the ILP content selection based on number of themes (k) used in the theme constraint (stop words retained, punctuation removed, minimum sentence length = 5). The highest F1 scores based on the number of themes in each ROUGE metric are highlighted in **bold**.

3.4 Information Ordering

For the development of this system, we implemented four IO methods: *Position Ordering*, *Majority Ordering*, *Cosine Similarity Ordering*, and *Jaccard Similarity Ordering*. Given the scope of the project, we did not run experiments for all combinations of CS and IO methods.[6] Specifically, position ordering is only used in combination with the TF-IDF CS method; majority ordering is only used in combination with the ILP CS method.[7] Cosine similarity ordering and jaccard similarity ordering are used in combination with both CS methods. The following subsections describe the IO methods we implemented.

3.4.1 Position Ordering

Each sentence in the summary is hashed with its index in the original article. The position of the sentence is the ratio of the index (s_i) to the total number of sentences (n) in the article.

$$p_s = s_i/n \tag{9}$$

The results of position ordering is the sorted outcome of the positions of selected sentences in their original article. Specifically, position ordering is performed only on the sentences selected with TF-IDF instead of all sentences in the original documents.

3.4.2 Majority Ordering

The idea of majority ordering is to classify sentences into themes before CS and to create a final order of the themes based on the order of those themes in the original articles. This IO method consists of two steps: (1) classifying sentences into themes by clustering and (2) constructing a theme graph to get the final order of themes.

First, sentences under the same topic are classified into themes by using the k-means clustering algorithm implemented by the `sklearn` library. We clustered into 10 themes for the present implementation of majority ordering. The algorithm divides the total sentences into clusters by grouping the nearest data points together until the number of clusters reaches the designated number of clusters.

Once each sentence under a topic has its corresponding theme, we constructed a weighted directed graph. The weights on the arcs are represented by the number of input texts which have sentences from a theme occurring before other themes. For example, the arc weight from theme 1 to theme

[6] Doing so lessened the text-heavy options for the survey we conducted for deciding our final IO methods (i.e. not having five IO options for each summary generated with different CS method).

[7] We elected to only implement MO for ILP, since theme clustering was already implemented for the model. This saved computational time, since themes would not need to be re computed.

2 is the number of sentences from theme 1 occurring before sentences from theme 2. On the other hand, the arc weight from theme 2 to theme 1 is the number of sentence from theme 2 occurring before sentences from theme 1. To decide the final order of the themes, we calculate the weight of all themes iteratively. In each iteration, the weight for each theme is obtained by subtracting the total incoming arc weights of the theme from its total outgoing arc weights. The most weighted theme in each iteration is the theme with the highest priority. This theme with the highest priority along with its relevant arcs are then removed from the graph, and the next iteration proceed. Therefore, once the iterations end, we get the final order of the themes in terms of the priority of occurrence.

This final order is then used to reorder the sentences in the output summaries. Sentences belongs to the themes with higher order are prioritized in the reordering process. Note that this IO method is only done after the sentences for an output summary are selected by the ILP model. This MO method looks at the set of sentences that will be in the output summary and order them according to the priority of the themes.

3.4.3 Cosine Similarity Ordering

For the selected sentences in a summary, we obtained permutations of these sentences to generate all possible orderings of a summary. Then, we calculated the cosine similarities between consecutive sentences. To calculate cosine similarities, we first remove stop-words in sentences, then, we use sentence-bert to covert sentences to vectors, and compute cosine similarities between two vectors(10). The score of each ordering is the sum of the added cosine similarities (11).

$$cos(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{10}$$

$$Score(O) = \sum_{i}^{n-1} cos(S_i, S_{i+1}) \tag{11}$$

The ordering with the highest score is the output of this method. Suppose the original summary contains n sentences. There would be n! permutations. Calculating the scores of all possible ordering is an NP hard problem. However, since there is a word limit of 100 words, the original summaries usually contain 3-6 sentences, which makes the permutation number small, so exhaustive search is feasible.

3.4.4 Jaccard Similarity Ordering

Much similar to cosine similarity, we looked at the sum of each order, with similarity between sentences represented by the ratio of the intersection of two neighbour sentences to the their union 12, and output the ordering with the highest score 13.

$$jac(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{12}$$

$$Score(O) = \sum_{i}^{n-1} jac(S_i, S_{i+1}) \tag{13}$$

Stop-words are also removed before the calculation of Jaccard similarity.

3.5 Content Realization

For CR we implemented Name Resolution, which is divided into two parts.

The first part uses `spaCy` to build a find people module. Given a string sentence as input, we loaded the "en_core_web_sm" `spaCy` language model with lemmatization disabled perform named entity recognition. We selected only entities tagged as 'Person', which we will call 'name' for the remainder of the paper. Modified names are retrieved using `spaCy`'s dependency parser by accessing `noun_phrases` through `sentence.noun_chunks`. Then we created two lists: one to append all the unmodified names(e.g. George W. Bush); the other to append all the modified names (e.g. President George W. Bush). The two lists are co-indexed so that given a name instance, it is possible to use its index to find its modified version in the second list.

The second part resolves the names in the summaries that are generated after CS and IO. This part contains three steps. The first step is to call the `find_people` module for each sentence in the summaries generated after CS and IO and extract all the unmodified names (e.g. George W. Bush) and their modified version (e.g. President George W. Bush). Second, for each unmodified name, we trace back to the pre-processed articles in the current topic ID and call the module again to find *all* the modified versions of the name, returning the longest name. Lastly, similar to re-write rules regarding discourse-new and discourse-old named entities developed by Siddharthan et al. (2011), our re-write rules state the following: if a name appears for the first time in the text, replace this name with the longest name found in the second step; for the later occurrences of a name, replace the modified name with the unmodified one.

4 Results

In this section, we present the results for the summaries generated with the CS and IO methods used in this system. We also show the results of the survey we conducted for deciding which IO method to use in our current system.

4.1 Content Selection

Our summarization system is evaluated on the Recall-Oriented Understudy for Gisting Evaluation metrics (Lin, 2004, ROUGE) with the ROUGE python package.[8] Specifically, we use the ROUGE-N to measure the number of n-grams occurring in both system-generated summaries and the human summaries. For the evaluation of our system, we evaluated matching unigrams and bigrams between the system-generated summaries and the human summaries. The metrics included in the evaluation process are: recall, precision, and F1 scores.[9] The results of the two CS methods are presented in the paragraphs that follow.

As shown in the CS + IO column[10] of Table 4, The ILP model outperforms the TF-IDF model in ROUGE-1 and ROUGE-2 score. However, both models only capture around a third of the unigram content occurring in the human summaries.

The recall scores of the ILP model are 0.34492 and 0.09454 for measuring the unigrams and bigrams across the system-generated summaries and the human summaries, showing that this method captures around 34.5% of the unigrams and 9.5% of the bigrams occurring in the human summaries. The precision scores of ILP model are 0.38148 and 0.10488, showing that around 38% of the unigrams and 10.5% of the bigrams in the system-generated summaries are consistent to the human summaries.

Both TF-IDF and ILP results improve upon our baseline system which selects the the first sentences up to 100 whitespace delimited tokens from a random article in the input set.

4.2 Information Ordering

To decide the final IO methods used in our system, we conducted a survey. In the survey, we asked 14 human raters to rate 5 set of summaries for each CS method (i.e.TF-IDF and ILP), 10 in total. Each set contains four summaries produced by different ordering methods. Specifically, for summaries generated with TF-IDF, the human raters were asked to choose the best one from summaries produced with the following IO methods: (1) no ordering (2) cosine similarity (3) jaccard similarity, and (4) position ordering. For summaries generated with ILP, the human raters were asked to choose the best one from summaries produced with the IO methods: (1) no ordering (2) cosine similarity (3) jaccard similarity, and (4) majority ordering. The results of the 14 responses are presented in table 5

	Method	Ave Vote (%)	Best Vote
TF-IDF	No IO	12.84	0
	PO	**45.72**	4/5
	Cos_simi	17.12	0
	Jac_simi	24.26	1/5
ILP	No IO	25.72	1/5
	MO	28.56	2/5
	Cos_simi	**32.82**	2/5
	Jac_simi	12.86	0

Table 5: Human rating of IO methods result. In this table, *Ave vote* refers to the average percentage of votes that each method was chosen as the "best" ordering out of the four given candidates; the IO method with the highest percentage for each CS method is denoted in **bold**. *Best Vote* represents the number of summaries that each IO method is chosen as the best ordering according to the majority vote of the 14 raters on the orderings out of all summaries for a given CS method.

For the TF-IDF model, position ordering has the highest average vote of 45.72%; out of the 5 sets, it produced 4 best summaries out of 5 according to the response. For the ILP model, cosine similarity has the highest average vote of 32.82%; out of the 5 sets, it produced 2 best summaries, the same as majority ordering.

Looking at the performance across two models, cosine similarity ordering performed well with the results produced by the ILP model, whereas it has the lowest performance with the results produced by the TF-IDF model.

4.3 Content Realization

After implementing the name resolution for CR, we conducted a thorough evaluation of our system using TF-IDF and ILP, both before and after the application of CR. As we see in table 4, our results demonstrate that ILP outperforms TF-IDF in terms of overall performance, with a slight reduction in precision and F-1 scores for unigrams and bigrams

[8] https://pypi.org/project/rouge-score/

[9] The scores reported here are the average scores of all system-generated summaries

[10] Recall that IO does not change the sentences selected, only the order of them.

N	Metric	CS + IO		CS + IO + CR		Baseline
		TF-IDF	ILP	TF-IDF	ILP	
1	Recall	0.31028	0.34492	0.31293	**0.34691**	0.23824
	Precision	0.37183	**0.38148**	0.36833	0.37779	0.33036
	F1-score	0.33793	**0.36202**	0.33796	0.36143	0.27436
2	Recall	0.07246	0.09454	0.07350	**0.09514**	0.05136
	Precision	0.08654	**0.10488**	0.08591	0.10353	0.07092
	F1-score	0.07880	**0.09936**	0.07913	0.09908	0.05906

Table 4: ROUGE-N scores when using CS + IO, using CS + IO + CR, and baseline on the development dataset

after the application of CR. Since the decrease in performance was not significant, and we observed that the readability of the summaries was improved substantially, which validates the effectiveness of CR in producing more coherent and understandable summaries.

To further validate our system, we also conducted a test on the evaluation dataset with our best systems and a baseline. TF-IDF is followed with position ordering and CR; ILP is followed with cosine ordering and CR. As shown in Table 6, our findings revealed that, once again, the ILP method exhibited superior overall performance compared to TF-IDF. Interestingly, we observed that our baseline approach yielded impressive ROUGE-scores, with the precision for unigrams even surpassing that of TF-IDF and ILP. However, despite the encouraging results of our baseline, the ILP method remained the optimal choice for our system.

N	Metric	CS + IO + CR		Baseline
		TF-IDF	ILP	
1	Recall	0.32825	**0.36769**	0.29203
	Precision	0.38140	0.39139	**0.39313**
	F1-score	0.35214	**0.37873**	0.33294
2	Recall	0.08051	**0.11572**	0.07610
	Precision	0.09366	**0.12296**	0.10465
	F1-score	0.08641	**0.11909**	0.08748

Table 6: ROUGE-N scores of the best systems on the evaluation dataset

5 Discussion

In this section, we provide analyses for the results of our current system. We break down our discussion based on each component: CS, IO, and CR.

5.1 Content Selection

The two methods we implement for CS yield similar ROUGE scores, as is shown in Table 4. However, the content of the summaries they generated are most often very different.

We can make several observations by taking a look at the first four summaries generated by each CS method (Topic IDs D1001-D1004; see Appendix D). Rarely, do the two CS methods select the same exact sentence. Of the four summaries, only two sentences match between the two CS methods; within a single summary there is at most one matching sentence. Both methods similarly select sentences with location headings in D1001 (i.e. . *"COLO-SCHOOL-SENIORS (Littleton , Colo.)..."* and *"LITTLETON , Colo. (AP)..."*). This information is not desirable, and could be post-processed if desired.

A major difference we see is that each CS method prioritizes different pieces of information. For example, in summary D1004, TF-IDF selects information about which villages a tsunami had destroyed (*"...wiped out three villages and had almost completely destroyed another"*; *"...the Nimas village near the Sissano lagoon , the Warapu village and the Arop village had been wiped out and the Malol village had almost been completely destroyed ."*). Whereas, for the same topic ID, ILP selects quantitative statistics as important (*"...located about 600 kilometers (375 miles) east of the northeastern tip of Australia . "*; *"23-foot (seven-meter) wall of water"; "...the population in the area affected by the tsunami was 8,000 to 10,000 people ."*) Moreover, in summary D1003, while we see that both methods select for statistics, no two statistics are the same between summaries. While our CS methods can extract important information, it is clear that they operationalize "important" differently. Depending on the CS method chosen, one will gain different summarized insights.

We also notice that, in the case of D1002, that ILP selects many named persons as concepts (*Amadou Diallo, State Supreme Court Judge John*

P. Collins, Kenneth Boss, Sean Carroll, Edward McMellon, Richard Murphy), while TF-IDF selects the main names (*Diallo, Saikou Amad Diallo*). Recall that ILP utilizes degree 1 skipgrams, which can span at most three tokens, while TF-IDF utilizes unigrams. Perhaps, the wider token scope of ILP concepts allows it to pick up on longer names (*Diallo* vs. *Amadou Diallo*–the victim in the articles). Since we have used Name Resolution as our CR method, this difference in name selection can have downstream effects.

5.2 Information Ordering

For TF-IDF model, position ordering has the highest human voting. We first assumed that it was due to the fact that many sentences in the summary were from one article, causing positioning to be the most effective ordering feature. However, when looking at the example where position ordering has the highest voting percentage, we found that five sentences in the summary were from four different articles. Hence, another explanation is needed to account for the results. After examining the data and the TF-IDF model, we conclude two reasons why position ordering significantly ourperformed ordering with cosine similarity and jaccard similarity.

First, the previously mentioned four articles that produced five sentences roughly follow the same writing style and logic. Majority of them start out by the discovery of the trace of giant pandas, then segue into a general discussion of pandas and their endangered situation, and end with the nation's protective measure to save them. Therefore, despite being drawn from different articles, the position of the sentence is still a strong indicator of its sequence in the summary. Second, since we have already used cosine similarity to remove similar sentences in the summary, the remaining ones do not have many words in common. This will cause jaccard similarity and cosine similarity between sentences to be very low, making ordering with these two methods not as informative as position ordering.

When applying IO to the ILP selected sentences, we saw a much more uniform distribution of which IO method was preferred on average (as compared to the TF-IDF selected sentences). We can infer some reasoning behind this. In the ILP objective function, each concept, regardless of frequency in the selected sentences, is only counted once towards the weighted sum. Thus, having repeated concepts across sentences, while certainly acceptable, is not ultimately beneficial to the ILP solution. Perhaps, this leads sentences to be quite conceptually distinct, and it is hard for sentences to flow smoothly together, regardless of the IO method chosen. If this is the case, we would expect a uniform distribution from averaged rater evaluations, which is exactly what we see.

Overall, the strength, or lack thereof, of CS can impact the fluidity of a certain IO method as well. A recipe is only as good as the ingredients used. It is possible that CS method acts as a confounding variable when interpreting IO method ratings.

Lastly, our rater sample is small at 14 people, and many survey items had results for each ordering option. While we are able to to see a majority choice for each survey option, it's important to note that proper ordering is a subjective choice that varies from human to human.

5.3 Content Realization

Our method of name resolution produced some good re-written results. For example, in a summary that addresses a disaster happening in Papua New Guinea, more information is given to a person who gave a speech on the disater: *'Dalle'* was re-written to *'chairman Dickson Dalle'*. From the poorly re-written summaries, we attribute the undesirable results to four issues: (1) over-recognition and under-recognition: spaCy sometimes identifies a non-name as a person's name – a false positive (e.g. *Anorexia, Jilin,* etc.) – or fails to identify a person's name – a false negative (e.g. *Pohl*). (2) Co-referencing issues: spaCy sometimes treats full names and the first name as two different people (e.g. *Hunter* and *Alex Hunter*). (3) Inaccurate extraction: when finding modifications, the find people module logic does not check for names being used as modifiers. Thus, when finding modifications, the script may falsely return a noun phrase where the name is used as a modifier, and the modified person is not the original un-modified name (e.g. *Golkar* returning *Golkar spokesperson Marwah Daud Ibrahim*). (4) Unwanted clipping: in very rare cases, where a name can refer to multiple entities, the rules clip modifiers for later instances, resulting in vagueness (e.g. *Mrs. Ramsey* clipped to Ramsey. We do not know if "*Ramsey*" is *Mr.* or *Mrs. Ramsey*). With more time, we would examine all the summaries, calculate the ratio of good re-

written results, and likely implement approaches to solve the four mentioned issues. In the future, we might consider methods such as using a different package/different model to extract names to potentially improve the outcome.

5.4 Ablation Study

The purpose of an ablation study is to assess the effectiveness of a system by removing specific components and examining the impact of their absence on the system's overall performance (Meyes et al., 2019). Since our best system consists of three parts: CS, IO, and CR, we will remove one part at a time and see the resulting ROUGE Score. As is shown in Table 7, the system with CS removed has a significantly lower ROUGE score, suggesting CS contributes most to the overall performance. IO also contributes to the final ROUGE score. As for CR, it increases recall but decreases precision and F1-score. We still keep CR in our best system because it can improve readability.

6 Conclusion

We have implemented and evaluated two methods for CS: TF-IDF and Integer Linear Programming. Our current results show that ILP has the best performance for returning unigrams and bigrams that appear in human generated summaries. Additionally, our current results improve upon our baseline.

We also implemented four different versions of IO methods. Through evaluation via a small sample of human raters, we learned that Position Ordering works best for TF-IDF and Cosine Similarity Ordering works best for ILP.

Finally, we implemented an initial version of name resolution for CR. Our analysis of the name resolution method showed some positive re-written results, but also identified issues with over/under-recognition, co-referencing, inaccurate extraction, and unwanted clipping. To improve the method, we need to address these issues by examining all summaries, calculating the ratio of good results, and potentially using a different package/model for name extraction. Overall, there is room for improvement to increase accuracy and effectiveness.

References

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18, Boulder, Colorado. Association for Computational Linguistics.

D Graff. 2002. The aquaint corpus of english news text. philadelphia, pa: Linguistic data consortium.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Richard Meyes, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. 2019. Ablation studies in artificial neural networks. *arXiv preprint arXiv:1901.08644*.

Stuart Mitchell, Michael J. O'Sullivan, and Iain Dunning. 2011. Pulp : A linear programming toolkit for python.

Mudasir Mohd, Rafiya Jan, and Muzaffar Shah. 2020. Text document summarization using word embedding. *Expert Systems with Applications*, 143:112958.

Joel Neto, Alex Freitas, and Celso Kaestner. 2002. Automatic text summarization using a machine learning approach. volume 2507, pages 205–215.

Michael Paul, ChengXiang Zhai, and Roxana Girju. 2010. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 66–76.

Senthamizh Selvan .R and K. Arutchelvan. 2022. Automatic text summarization using document clustering named entity recognition. *International Journal of Advanced Computer Science and Applications*, 13.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445.

Ellen Vorhees and David Graff. 2008. *AQUAINT-2 information-retrieval text: research collection*. Linguistic Data Consortium.

Appendix A: Work Split

For D1: Rachel went through the tutorial of git; Yi-Chien and Yian showed the basics of overleaf; Tashi and Chenxi wrote up the submission pdf.

For D2: Yian did the coding part, with some help from Rachel and Yi-Chien; Yi-Chien posted tutorial on setting up Anaconda environment on Patas for the group; Tashi made the slides for presentation; Chenxi wrote up the report D2. Rachel will be presenting in class.

N	Metric	No CS	No IO	No CR	Complete System
1	Recall	0.23906	**0.34732**	0.34492	0.34725
	Precision	0.32811	0.37744	**0.38148**	0.37796
	F1-score	0.27400	0.36139	**0.36202**	0.36164
2	Recall	0.05162	0.09520	0.09454	**0.09530**
	Precision	0.07043	0.10334	**0.10488**	0.10368
	F1-score	0.05904	0.09899	**0.09936**	0.09922

Table 7: Ablation study on development dataset. In this table, a Complete System means CS (ILP method) + IO (Cosine similarity) + CR. No CS means using baseline instead of ILP method. No IO means CS + CR. No CR means CS + IO.

For D3: Rachel implemented ILP method; Chenxi and Tashi implemented TF-IDF method; Yi-Chien implemented evaluation script and the pipeline. Yian tested and analyzed the result. Report and slides were contributed among everyone.

For D4: Rachel and Yi-Chien improved ILP method and implemented majority ordering within the ILP frame; Chenxi and Tashi improved TF-IDF method and implemented position ordering within the TF-IDF frame; Yian implemented Cosine similarity and Jaccard similarity ordering. Report, slides, and survey were contributed to by everyone.

For D5: Rachel and Chenxi worked on the find people model and name resolution respectively. Yi-Chien, Tashi, and Yian worked on the evaluation. Yian performed the ablation study.

Appendix B: Packages Used in the Systems

Link to the code repository on github:
https://github.com/rhantz/575_summarization
Off-the-shell tools used in code:

- `xml` to parse the path of input file and the xml document in corpora
- `nltk` for sentence and word tokenization, stemming, and uni/bi/skipgramming
- `spaCy` for Named Entity Recognition
- `os` for system operation on Patas
- `PuLP` for Linear Programming
- `GLPK` as the ILP solver
- `TfidfVectorizer` from `sklearn` for TF-IDF matrix
- `KMeans` from `sklearn` for clustering sentences into themes for majority ordering
- `Pandas` for building dataframe
- `rouge_score` for evaluation

Appendix C: Problems and Solutions

Problem 1:
Description: documents in AQUAINT corpora are not rooted, causing parsing to fail.
Solution: created a root node by inserteing <tag> at the start and appending <\tag> in the end.

Problem 2:
Description: code fails to run on Patas for some group members
Solution: set Anaconda on Patas to ensure people have the same environment.

Problem 3:
Description: Anaconda environment with multiple dependencies became tricky to export and re-install from other's branches
Solution: Create new environment file from scratch with clear dependencies

Problem 4:
Description: ROUGE implementation on patas was inaccessible due to missing perl file.
Solution: Use python rouge-score package instead

Problem 5:
Description: comparing cosine similarities between sentences for ILP would become a quadratic programming task.
Solution: Implement theme constraint that leverages clustering instead.

Problem 6:
Description: Our implementation of Name Resolution is only initial and could use deeper evaluation and improvements.
Solution: Detail our error analysis and propose what we would do with additional time

Appendix D: Content Selection Sample

ID	TF-IDF	ILP
1	Authorities believe Columbine students Eric Harris and Dylan Klebold carried out the massacre and then killed themselves . Many wore Columbine clothing ; everything in sight – even portable toilets – was festooned with blue Columbine memorial ribbons . COLO-SCHOOL-SENIORS (Littleton , Colo.) _ A look at the situation of seniors and others at a now-closed Columbine High . Columbine on Saturday , fingers laced together as they cried for the children . The community outpouring has touched some Columbine students . **She said the school was allowing students to stay home .**	**She said the school was allowing students to stay home .** There are the communities that existed already , like Columbine students and Columbine Valley residents . LITTLETON , Colo. (AP) – Students returned to classes Thursday at Chatfield High School , but the bloodbath at rival Columbine High haunted the halls . LITTLETON , Colo. _ Vice President Al Gore on Sunday bid farewell to Columbine High School 's dead , issuing a ringing challenge to parents and schools before a crowd of 70,000 . For Use By Clients of the New York Times News Service
2	Diallo 's father , Saikou Amad Diallo , arrived here Wednesday from the West African nation of Guinea and said he was anxious to see the officers not only charged , but brought to trial . Where was he when the four plainclothes officers began shooting ? Through their lawyers , the officers have said they thought Diallo had a gun . The officers were immediately suspended from the force . Off-duty police officers and more than 25 of Diallo 's friends and relatives packed the small courtroom for the 45-minute arraignment .	The four police officers have yet to officially explain their actions . They are accused of firing 41 times at Amadou Diallo while searching for a rape suspect on Feb. 4 . State Supreme Court Judge John P. Collins set bail at $ 100,000 for each officer , which was expected to be met . Officers Kenneth Boss , Sean Carroll , Edward McMellon and Richard Murphy pleaded innocent in a Bronx courtroom to second-degree murder . The officers in the Diallo case did not testify before the grand jury . NEW YORK _ The Rev .
3	The Baishuijiang State Nature Reserve is home to more than 100 giant pandas . Giant pandas , said to have been around during the time of dinosaurs , are cited as a " national gem " of China . The nature preserve , occupying about 220,000 hectares , has 102 giant pandas living wild . If the situation worsens all the pandas in the reserve will have to be transferred elsewhere . In Sichuan and Shaanxi provinces , two other habitats of giant pandas , arrow bamboo was also found blooming .	China has 163 giant pandas in captivity . Currently more than 1,500 giant pandas live wild in China , according to a survey by the State Forestry Administration . The giant panda is one of the world 's most endangered species , with an estimated 1,000 living in the mountainous regions of Sichuan , Shaanxi and Gansu provinces . The bank at southwest China 's Giant Panda Protection and Research Centre in the Wolong Nature Reserve in Sichuan province will be completed this year , the China Daily said . And the blooming area has continued to expand .
4	Authorities at Aitape in the West Sepik province , on Papua New Guinea 's northwest coast , said the tsunami that hit the coast west of Aitape on Friday night had wiped out three villages and had almost completely destroyed another . Dalle said the Nimas village near the Sissano lagoon , the Warapu village and the Arop village had been wiped out and the Malol village had almost been completely destroyed . **Australia said it will provide transport for relief supplies and a mobile hospital to Papua New Guinea .**	**Australia said it will provide transport for relief supplies and a mobile hospital to Papua New Guinea .** The capital of Port Moresby is located about 600 kilometers (375 miles) east of the northeastern tip of Australia . The 23-foot (seven-meter) wall of water hit Friday night without warning following an earthquake about 12 miles (30 kilometers) off the coast of Papua New Guinea in the Pacific Ocean . Robert Igara , the government 's chief secretary , said the population in the area affected by the tsunami was 8,000 to 10,000 people .

Table 8: Summaries before CR for TF-IDF and ILP (Topic IDs D1001-D1004. Sentences in **bold** are the same between summaries.