Ling575 Summarization System

D4: Improved system

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

Overview

Improvements in content selection

- Method 1: TF-IDF
- Method 2: ILP

Information ordering (IO)

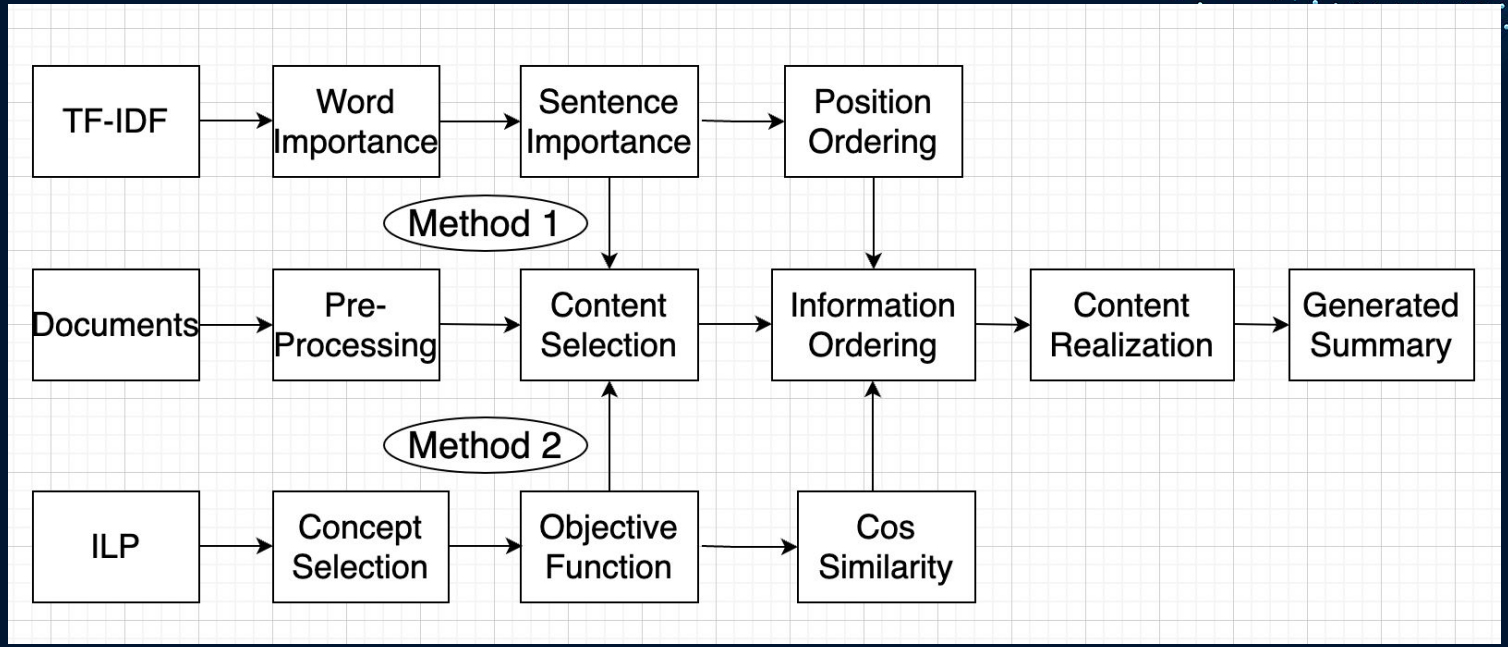
- Position Ordering (TF-IDF)
- Majority Ordering (ILP)
- Ordering with cosine similarity and jaccard similarity (Both)

IO human rating report

WorkSplit

- Baseline evaluation - Yian
- TF-IDF improvement & Position Ordering - Tashi & Chenxi
- ILP improvement & Majority Ordering - Rachel & Yi-Chien
- Cosine similarity & Jaccard similarity - Yian
- Updated Evaluation & Pipeline - Yi-Chien
- Report & Slides - Everyone

System Architecture



TF-IDF Recap

TF-IDF in D3

- Built the model
- Testing hyperparameters:
  - Cosine similarity
  - Sentence length

TF-IDF update overview

Content selection

- Harmonize input and output sentences
- Literature review for D3 hyper parameters
- Implement bigrams
- Attempted solution for linking word

Information ordering

- Position Ordering (only within TF-IDF)

Fixing output

Pre-processing out come:

```
156  TECHNICAL PROBLEMS : Peter Trigg
157
158  Phone : ( 212 ) 499-3332 .
```

D3 output:

```
5559  that would have had a minimal impact on trade in the goods .                    27  43  0.013033
5560  TECHNICAL PROBLEMS : Peter Trigg  Phone : ( 212 ) 499-3332 .                    31  35  0.013014
5561                                                  By James Brooke .               3   4   0.012677
```

D4 output:

```
5804                By Michael M. Weinstein .          4  5  0.007174  80
5805       TECHNICAL PROBLEMS : Peter Trigg            5  5  0.006826  97
```

TF-IDF update overview

Content selection

- Harmonize input and output sentences
- Literature review for D3 hyper parameters
- Implement bigrams
- Attempted solution for linking word

Information ordering

- Position Ordering (only within TF-IDF)

Literature on hyperparameters

Cosine Similarity

- Removing sentences with cos_sim 0.75 before clustering - .R and Arutchelvan (2022)

Sentence Length

- Penalizing sentences < 12 words - Teufel and Moens (2002)

Stop Words and Punctuation

- Consistent with pre-processing outcome

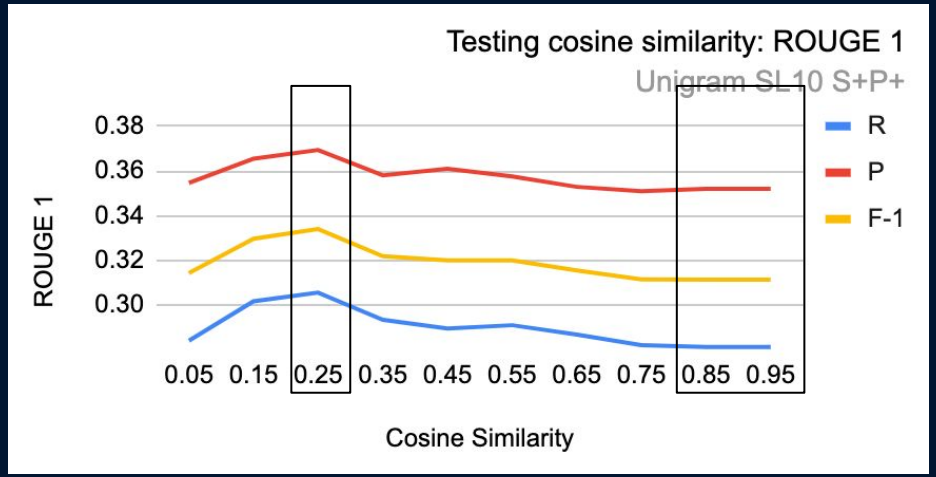
☐ Cos_sim
☐ Length
☐ SW & P

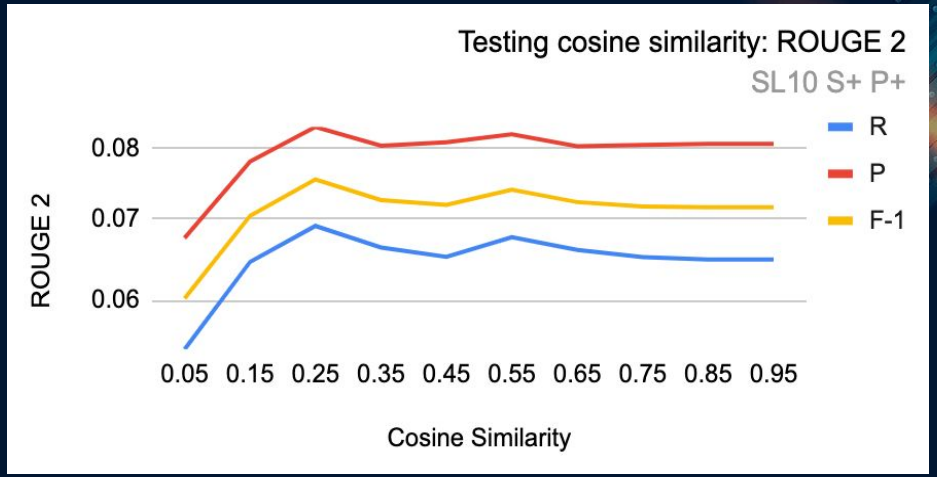
TF-IDF: Testing Cosine Similarity

Unigram Sentence Length 10 SW+P+

Start from 0.75, increment 0.1

Cos_sim 0.25 yields best score





- [x] Cos_sim
- [ ] Length
- [ ] SW & P

TF-IDF: Testing SW & P

Fixating cosine similarity 0.25

Testing SW+/- & P+/-

- Eg. SW+ P+ including stop words and punctuation

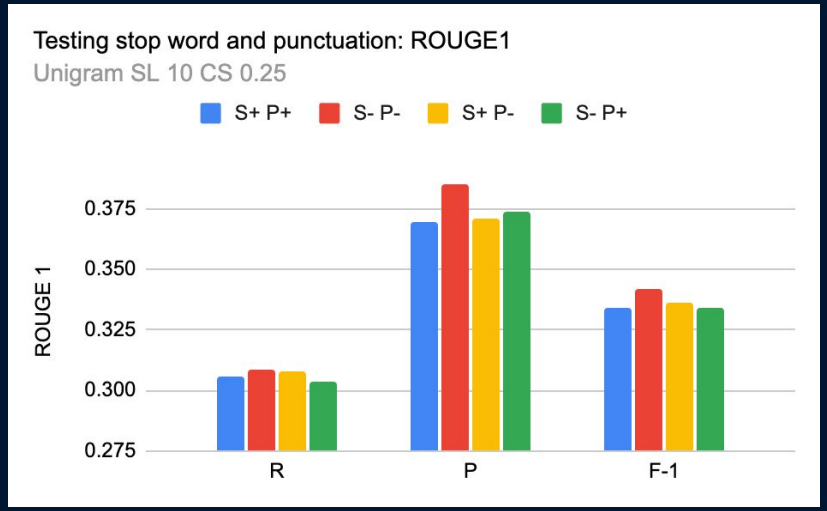
Sentence Length 10 & 8

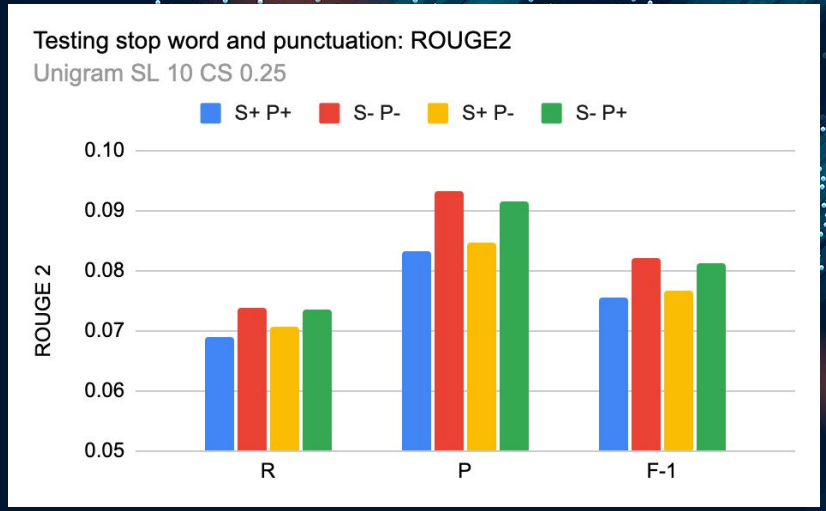
SW- P- yields best score (length 10), but long sentences in summary

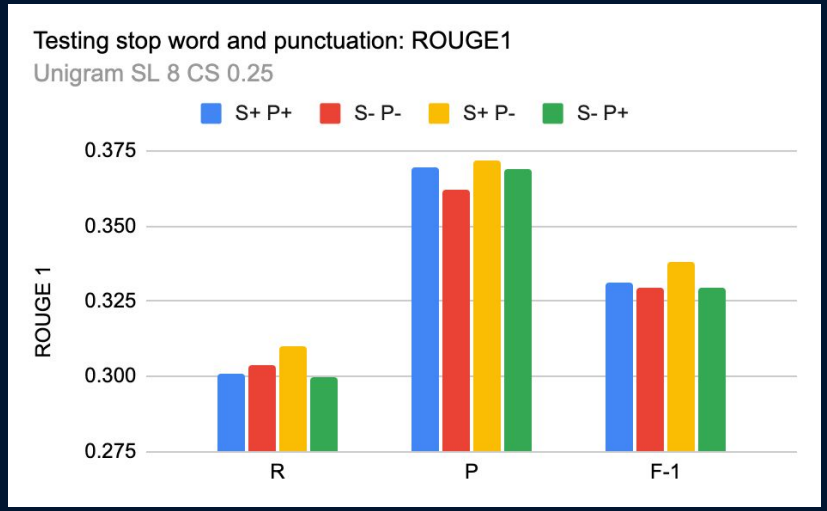
SW+ P- yields best score (length 8)

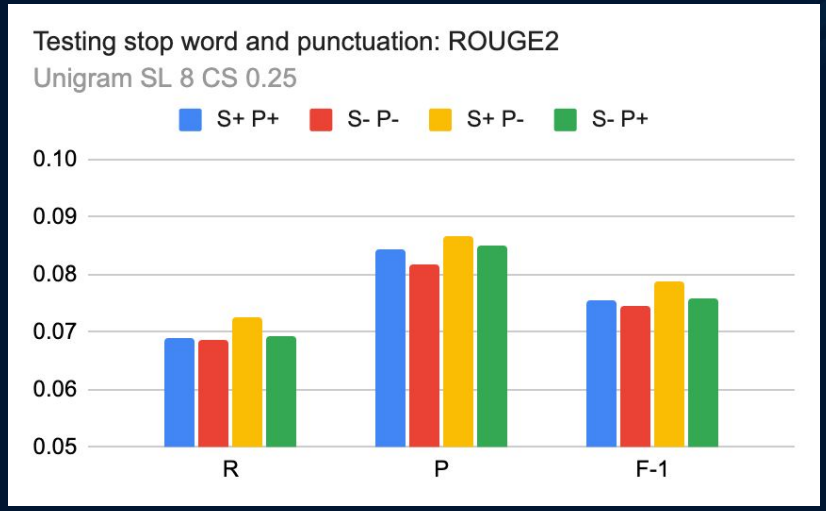
Choose SW+ P-

☑ Cos_sim
☐ Length
☑ SW & P


Testing stop word and punctuation: ROUGE1
Unigram SL 10 CS 0.25
S+ P+
S- P-
S+ P-
S- P+
ROUGE 1
0.375
0.350
0.325
0.300
0.275
R
P
F-1


Testing stop word and punctuation: ROUGE2
Unigram SL 10 CS 0.25
S+ P+
S- P-
S+ P-
S- P+
ROUGE 2
0.10
0.09
0.08
0.07
0.06
0.05
R
P
F-1


Testing stop word and punctuation: ROUGE1
Unigram SL 8 CS 0.25
S+ P+
S- P-
S+ P-
S- P+
ROUGE 1
0.375
0.350
0.325
0.300
0.275
R
P
F-1

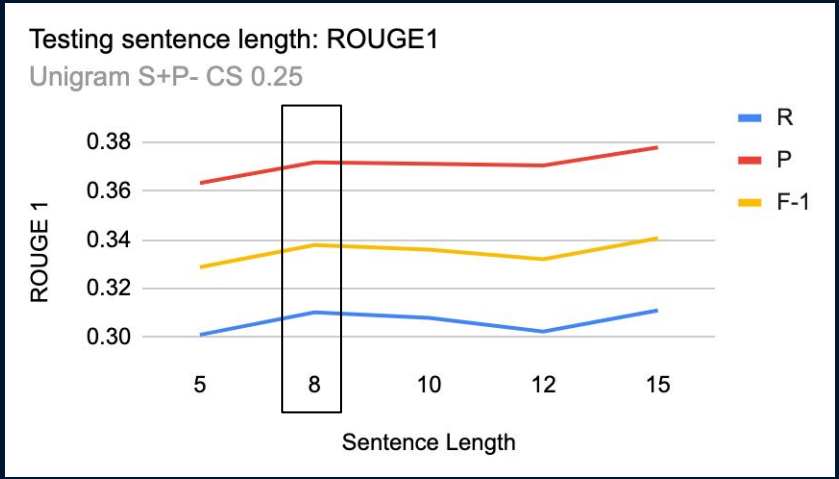

Testing stop word and punctuation: ROUGE2
Unigram SL 8 CS 0.25
S+ P+
S- P-
S+ P-
S- P+
0.10
0.09
0.08
0.07
0.06
0.05
R
P
F-1

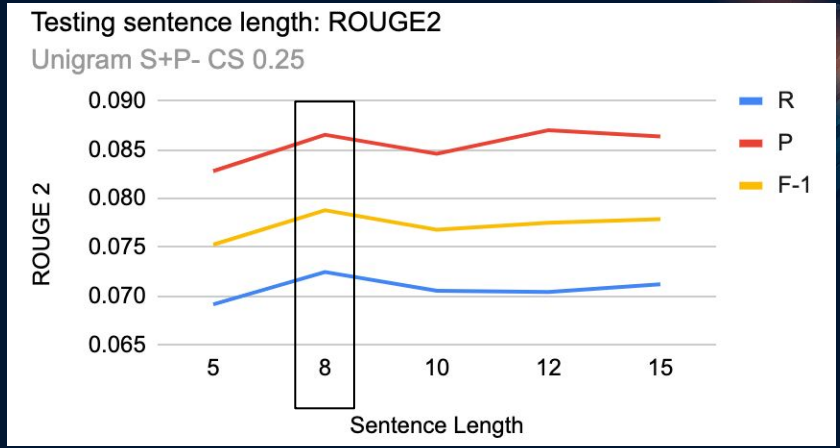
TF-IDF: Testing Sentence Length

Fixating cosine similarity 0.25, SW+ P-

Testing length 5 8 10 12 15

- Potential improvement with large length but long summary
- Choose length 8





✅ Cos_sim
✅ Length
✅ SW & P

TF-IDF update overview

Content selection

- Harmonize input and output sentences
- Literature review for D3 hyper parameters
- Implement bigrams
- Attempted solution for linking word

Information ordering

- Position Ordering (only within TF-IDF)

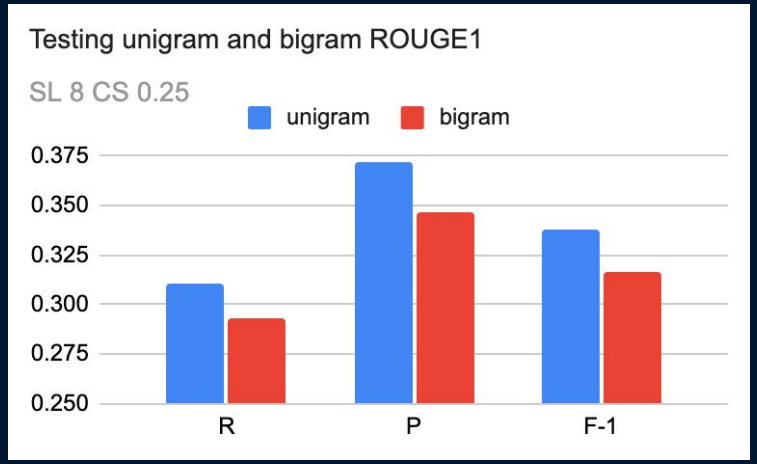
TF-IDF: implement bigram

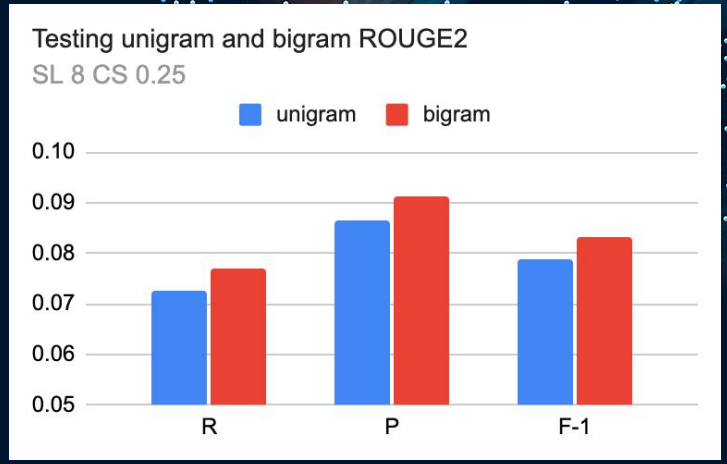
Initialize tfidfvectorizer with bigram

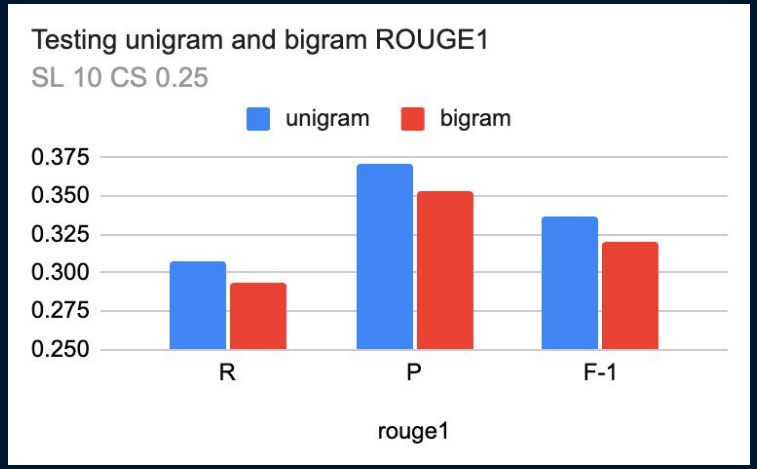
```
# bigram
if args.gram_type == "bigram":
    vectorizer = TfidfVectorizer(token_pattern=r'\b[^\s]+\b', ngram_range=(2, 2))
```

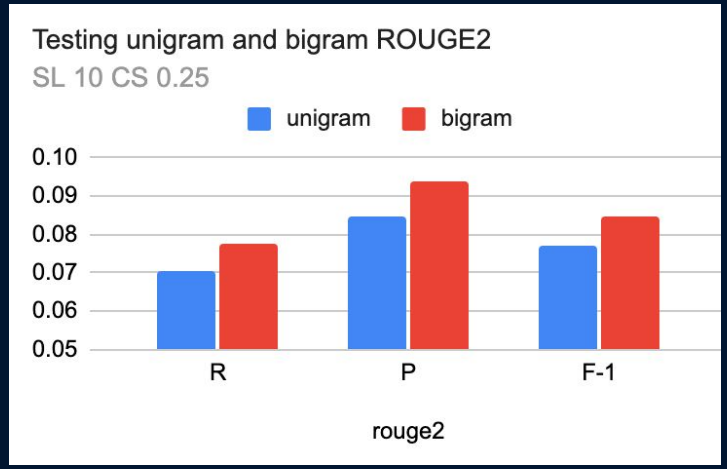
Tokenize sentences in the same format when calculating the sentence score

```
if args.gram_type == "bigram":
    tokenizer = nltk.RegexpTokenizer(r'\b[^\s]+\b')
    tokens_for_bi = tokenizer.tokenize(sentence)
    bigrams = list(nltk.bigrams(tokens_for_bi))
    for i in range(len(bigrams)):
        bigrams[i] = bigrams[i][0].lower() + ' ' + bigrams[i][1].lower()
    score = sent_score(article_index // 10, bigrams, tfidf_dict1, args.gram_type)
```


Testing unigram and bigram ROUGE1
SL 8 CS 0.25
unigram
bigram
0.375
0.350
0.325
0.300
0.275
0.250
R
P
F-1


Testing unigram and bigram ROUGE2
SL 8 CS 0.25
unigram
bigram
0.10
0.09
0.08
0.07
0.06
0.05
R
P
F-1


Testing unigram and bigram ROUGE1
SL 10 CS 0.25
unigram
bigram
0.375
0.350
0.325
0.300
0.275
0.250
R
P
F-1
rouge1


Testing unigram and bigram ROUGE2
SL 10 CS 0.25
unigram
bigram
0.10
0.09
0.08
0.07
0.06
0.05
R
P
F-1
rouge2

TF-IDF update overview

Content selection

- Harmonize input and output sentences
- Literature review for D3 hyper parameters
- Implement bigrams
- Attempted solution for linking word

Information ordering

- Position Ordering (only within TF-IDF)

TF-IDF: solving linking words

Define a list that contain conjunctions and adverb

- Idea inspired by Mohd et al. (2002)
- Check if the selected sentence starts with a linking word
- Append the precedent sentence

```
conjunctions_and_adverbs = ["and", "but", "or", "nor", "yet", "so", "also", "besides", "consequently",
 "hence", "however", "indeed", "instead", "likewise", "meanwhile", "moreover", "nevertheless",
 "nonetheless", "otherwise", "similarly", "still", "subsequently", "then", "therefore", "thus"]
```

```
first_word = sentence.split()[0].lower()
if (first_word in conjunctions_and_adverbs):
    precedent_sent = df.loc[(df['Topic_id']==row[0])&(df['Article_id']==row[1])& (df['sent_index']== row[6]-1)]
    sent = precedent_sent['sentence'].values[0]
    info_order.append(sent)
```

TF-IDF: solving linking words

Result

- Slight decrease in ROUGE score
- Undesirable preceding sentences

```
But some oysters do survive in the bay .
`` The major problems that the Chesapeake Bay agreement was designed to diminish are still there . ''
In 2000 , the foundation rated the bay a 28 .
Collier , now in the Chesapeake Bay Maritime Museum in St. Michaels .
Out on the water , the oysters have kept dying .
This grim portrait does n't mean that there are no oysters left in the Chesapeake .
Once the Chesapeake Bay's premiere species , native oysters have been virtually wiped out due to over-harvesting and disease .
```

```
For centuries , the Chesapeake has been synonymous with oysters .
Here is the progress on a few of the important goals for the bay , based on information from the EPA 's Chesapeake Bay
Now , it stands alone in the shadow of the Kent Narrows Bridge .
But some oysters do survive in the bay .
In 2000 , the foundation rated the bay a 28 .
Collier , now in the Chesapeake Bay Maritime Museum in St. Michaels .
How did the bay get to this condition ?
```

TF-IDF update overview

Content selection

- Harmonize input and output sentences
- Literature review for D3 hyper parameters
- Implement bigrams
- Attempted solution for linking word

Information ordering

- Position Ordering (only within TF-IDF)

TF-IDF: Position Ordering

sentence hashed with index in the original article

position is the ratio of the index to the total number of sentences

Sort by position

```
position = {}
for total_snetence, sentence_index, sentence in sents:
    # key:position retio; value:sentence string
    position[int(sentence_index)/int(total_snetence)] = sentence

sorted_position = dict(sorted(position.items()))
position_list=list(sorted_position.values())

return(topic_ID,position_list)
```

$$p_s = s_i / n$$

Method 2: ILP Improvements

Issue 1: **Concept Construction**

```
if args.concept_type == "named_entity":
    doc = nlp(sent)
    sent_concepts = set([entity.text.lower() for entity in doc.ents])
    return sent_concepts

if args.remove_punctuation:
    sent_stemmed = [stemmer.stem(word) for word in tokenizer.tokenize(sent)]
else:
    sent_stemmed = [stemmer.stem(word) for word in sent.strip().split(" ")]

if args.remove_stop_words:
    sent_stemmed = [word for word in sent_stemmed if word not in stop_words]

# stop words are always removed for unigrams
if args.concept_type == "unigrams":
    sent_concepts = {unigram for unigram in sent_stemmed if unigram not in stop_words}

# for bigrams and skipgrams, only 1 stop word is allowed (unless stop words are fully removed)
elif args.concept_type == "bigrams":
    sent_concepts = {bigram for bigram in bigrams(sent_stemmed) if (bigram[0] not in stop_words) or (bigram[1] not in stop_words)}
elif args.concept_type == "skipgrams":
    sent_concepts = {skipgram for skipgram in skipgrams(sent_stemmed, 2, args.skipgram_degree) if (skipgram[0] not in stop_words) or (skipgram[1] not in stop_words)}
else:
    raise ValueError(
```

Concepts can be...

- Named Entities,
- Unigrams,
- Bigrams,
- or Skipgrams

From them, we can remove:

- Punctuation
- and/or Stop Words

Method 2: ILP Improvements

Issue 2: **Sentence Length**

- Sentences must be at a minimum sentence length (*m*)

```
for j, sentence in enumerate(sentences):

    # Retrieves length of each sentence
    l = len(sentence["text"].strip().split())

    if args.min_sent_length != 0:

        # s_j Length Constraint: s_j is above the minimum sentence length
        model += (l * s[j] >= args.min_sent_length * s[j], f"s{j}_length_constraint")
```

$$l_j s_j \geq m s_j$$

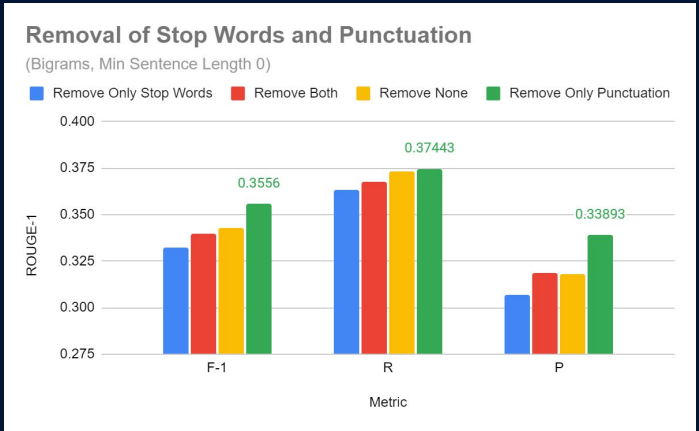
Issue 3: **Redundancy**

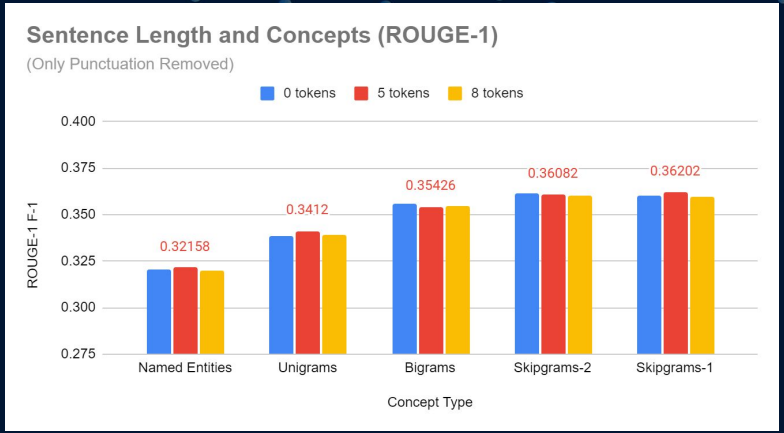
- Cluster sentences into themes (Sklearn Kmeans + TFiDF vectorizer)
- Of the selected sentences, each theme (*k*) can only occur *n* times (presently, only tested when *n*=1)

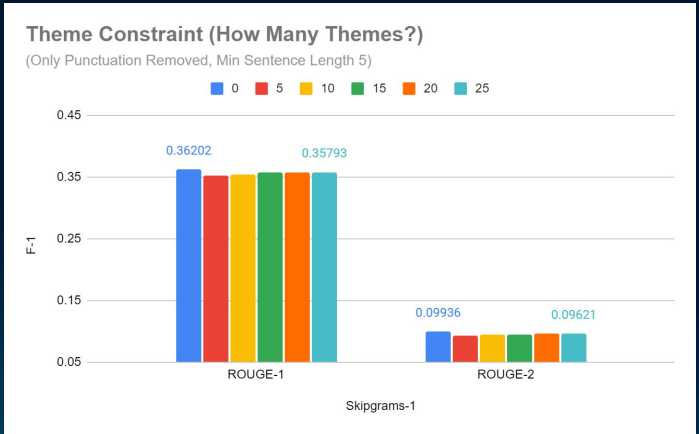
```
if args.theme_constraint:
    for theme in set(themes):
        theme_constraint = []
        for j, sentence in enumerate(sentences):
            theme_constraint.append(s[j] * theme_occurence[(j, theme)])

        # Theme Constraint: theme_t can only occur at most {args.theme_redundancy} time(s) in the selected sentences
        model += (lpSum(theme_constraint) <= args.theme_redundancy, f"theme_{theme}_constraint")
```

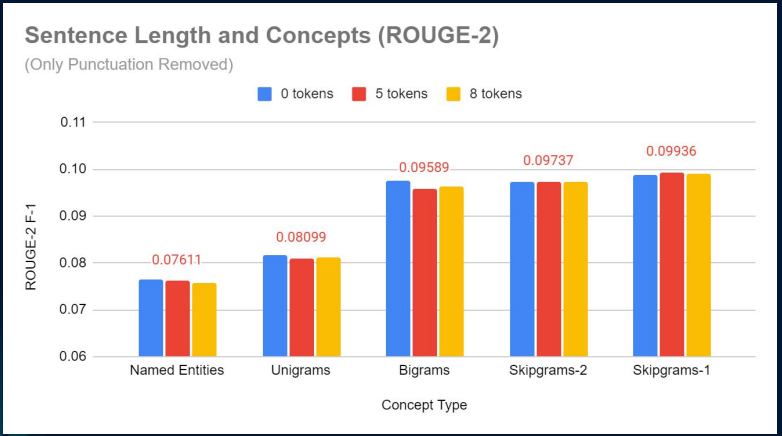
$$\sum_j s_j p_{jk} \leq 1 \forall k$$

ILP - Hyperparameter Search









Best Configuration

ROUGE-1 F1
0.36202

ROUGE-2 F1
0.09936

Remove Only Punctuation

Sentences of at least 5 tokens

Skipgrams-1

No theme constraint

Time ~1m15.083s

Majority Ordering (MO)

Main Steps:

- Form themes from the input documents
- Order the themes

MO: Forming themes

- Form the input sentences into clusters with the k-means clustering algorithm (with the KMeans module from the sklearn library)
  - Form *k* clusters
  - Minimize the variances within the clusters
  - Each sentence belongs to a theme
- The “themes” are represented with the cluster labels
  - Sentences within the same theme should contain information about similar scenarios

MO: Ordering themes

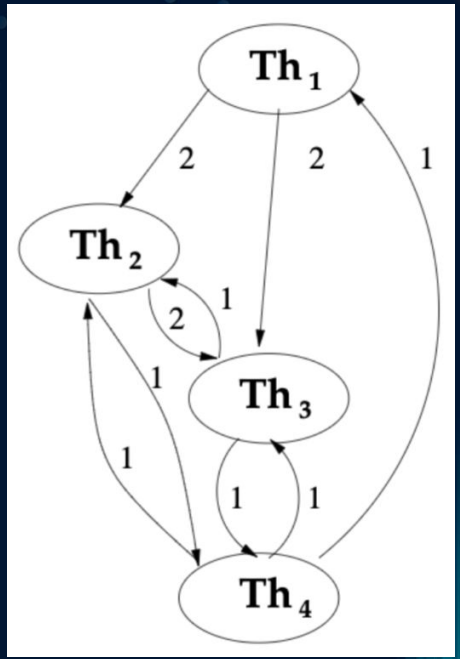
- Build a weighted directed graph
  - Arc weights:

    the number of input texts which have sentences from a theme occurring before other themes

- Order the themes:
  - Iterate through all themes a decide a theme most prioritized in each iteration
  - Theme weights:

    Sum(outgoing weights) - Sum(incoming weights)

  - The theme with the highest theme weight is the theme with highest priority in that iteration



IO - Cosine & Jaccard Similarity

Input: un-ordered summaries generated from TF-IDF and ILP

Output: reordered summaries

Assumptions: adjacent sentences have high cohesion

Two methods to measure cohesion:

- Cosine similarity
- Jaccard similarity

Information Ordering: Cosine similarity

Sentence is represented as mean of its word vectors. The similarity between two sentences is defined as the cosine of the angle between the vectors

Coherence of a text T is defined as the sum of cosine similarity scores of adjacent sentences.

$$sim = \frac{A \cdot B}{||A||||B||}$$

$$\text{coherence}(\text{T}) = \sum_{i}^{n-1} cos(S_i, S_{i+1})$$

Information Ordering: Jaccard similarity

The similarity between two sentences is defined as the amount of word overlap normalized by the union of the sets of words present in the two sentences.

Coherence of a text T is defined as the sum of Jaccard similarity scores of adjacent sentences.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$\text{coherence}(\mathrm{T}) = \sum_{i}^{n-1} jac(S_i, S_{i+1})$$

D3 vs. D4 Results Comparison

Baseline

ROUGE-**1** F1: **0.27436**
ROUGE-**2** F1: **0.05906**

TF-IDF	Metric	D3	D4
Rouge-1	Recall	0.30895	**0.31028**
	Precision	0.36750	**0.37183**
	F1-score	0.33475	**0.33793**
Rouge-2	Recall	0.07160	**0.07246**
	Precision	0.08529	**0.08654**
	F1-score	0.07761	**0.0788**

ILP	Metric	D3	D4
Rouge-1	Recall	0.30750	**0.34492**
	Precision	0.35340	**0.38148**
	F1-score	0.32734	**0.36202**
Rouge-2	Recall	0.07539	**0.09455**
	Precision	0.08723	**0.10488**
	F1-score	0.08049	**0.09936**

IO Human Rating

Human rating on best ordering

- First 5 topics
- 5 items each, 10 in total
- Each item choice of 4
- Single choice

	TF-IDF	ILP
1	No IO	No IO
2	PO	MO
3	Cos_sim	Cos_sim
4	Jac_sim	Jac_sim

https://docs.google.com/forms/d/e/1FAIpQLScP9t5CpnNyG5_1bmHNvLzAvQLHvDdQuDfRXk-sLG1UpgYeDQ/viewform

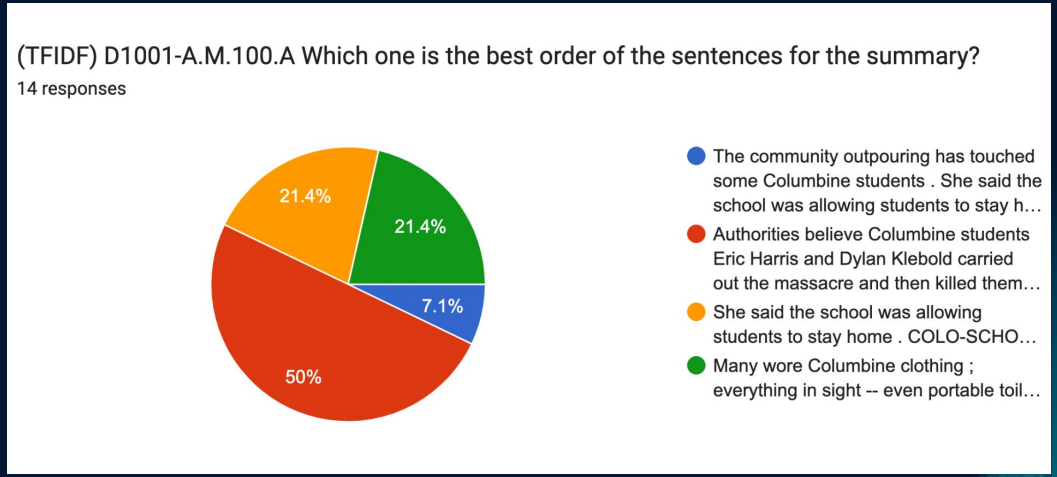
IO Human Rating

Collected 14 responses

Sample from one response

Evaluate IO methods

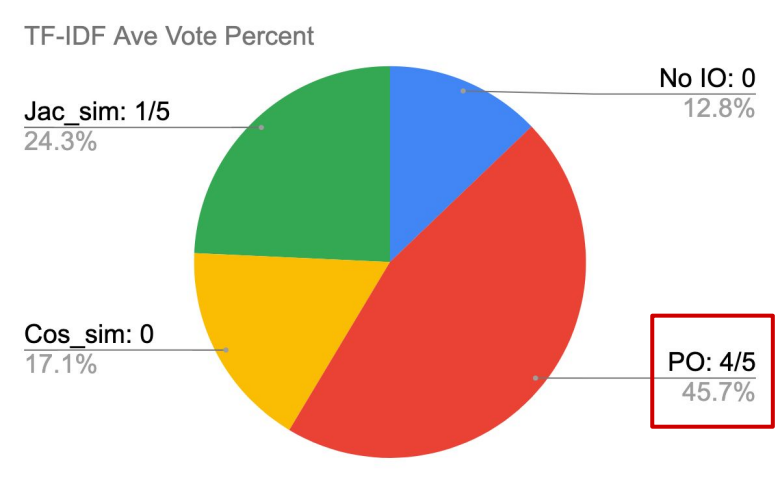
- Ave vote percentage
- Best votes

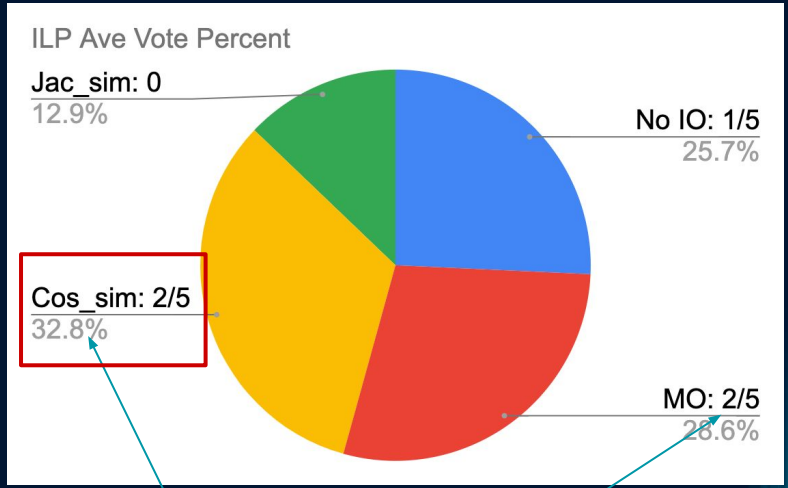


IO Human Rating

Results from 14 responses

- Cos_sim & Jac_sim perform very differently across models
- PO for TF-IDF
- Cos_sim for ILP





Ave Vote Percent

Best Votes

Analysis on Ordering

TF-IDF:

- Position ordering might outperform other method if most sentences are from the same article. (not the case)
- Jac & cos_sim are not informative when similar sentences are filtered

ILP:

- ILP counts concepts only once towards the weighted sum → Having repeated concepts across sentences is not beneficial.
- Conceptual connections and a smooth flow between sentences might be hard with ILP. It makes sense then, that the distribution of IO method ratings for ILP is uniform (not one method is overwhelmingly preferred)

IO Human Rating

Pos	Top_id	Jac order	Sentence
4/9	5	4	The Baishuijiang State Nature Reserve is home to more than 100 giant pandas .
7/11	1	2	Giant pandas , said to have been around during the time of dinosaurs , are cited as a `` national gem '' of China .
9/14	9	3	The nature preserve , occupying about 220,000 hectares , has 102 giant pandas living wild .
12/18	2	5	If the situation worsens all the pandas in the reserve will have to be transferred elsewhere .
11/14	9	1	In Sichuan and Shaanxi provinces , two other habitats of giant pandas , arrow bamboo was also found blooming .

IO Human Rating

Jac order	Sentence
1	In Sichuan and Shaanxi provinces , two other habitats of giant pandas , arrow bamboo was also found blooming .
2	The nature preserve , occupying about 220,000 hectares , has 102 giant pandas living wild .
3	Giant pandas , said to have been around during the time of dinosaurs , are cited as a `` national gem '' of China .
4	The Baishuijiang State Nature Reserve is home to more than 100 giant pandas .
5	If the situation worsens all the pandas in the reserve will have to be transferred elsewhere .