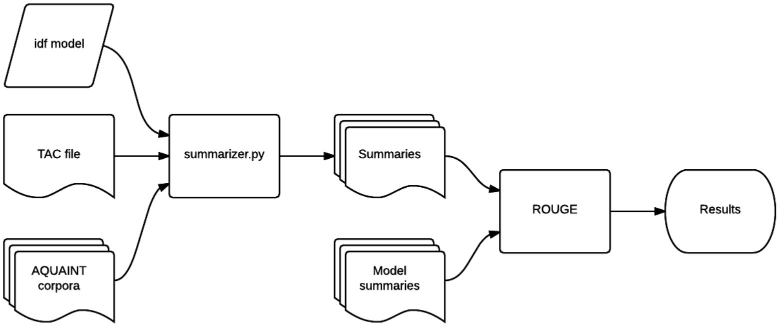
Automatic Summarization Project

Anca Burducea
Joe Mulvey
Nate Perkins

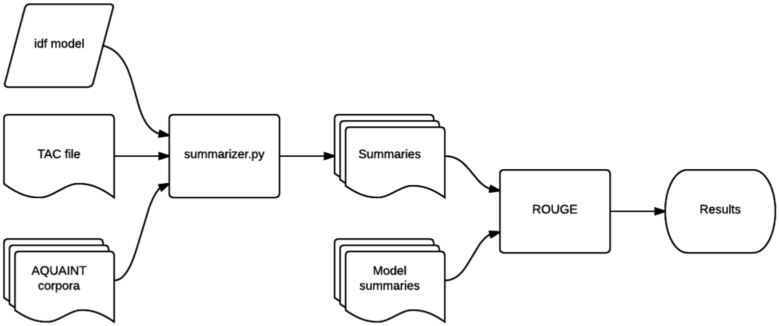
April 28, 2015

Outline

System overview

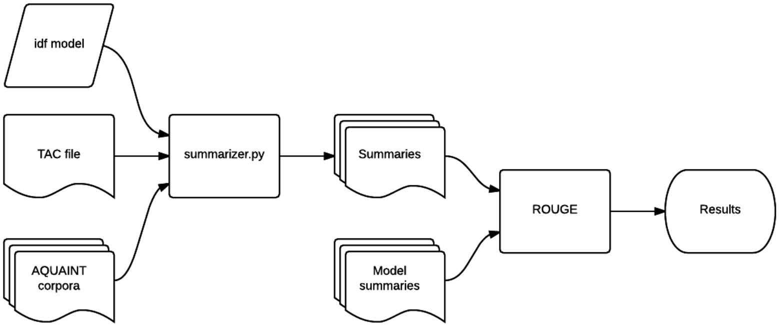


System overview



- Pyhton 3.4

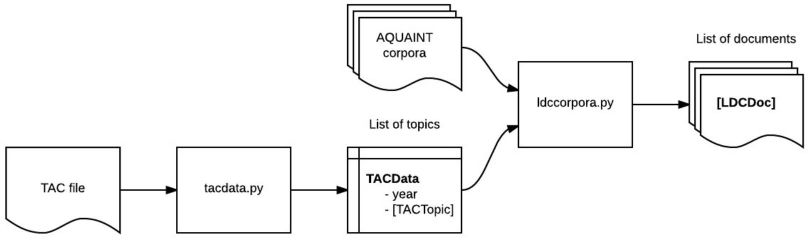
System overview



- Pyhton 3.4
- TF-IDF sentence scoring

Outline

Data cleanup

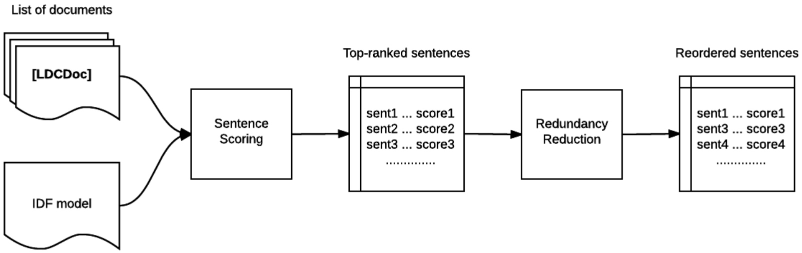


For each news story `N` in topic `T`:

- find the file `F` containing `N`
  - check files that have LDC document structure (`<DOC>`)
  - check file names (regex)
- clean/parse `F`
  - XML parse on `<DOC>...<\DOC>` structures
- find N inside F
- return N as an `LDCDoc` (timestamp, title, text ...)

Outline

Content Selection



Sentence scoring

Sentence S: [– + + * + – – + * * –]

- – meaningless word → punctuation, numbers, stopwords
- + meaningful word → the rest
- * topic signature word → top 100 words scored with TF*IDF

Sentence scoring

Sentence S: [– + + * + – – + * * –]

– meaningless word → punctuation, numbers, stopwords

+ meaningful word → the rest

* topic signature word → top 100 words scored with TF*IDF

$$\text{Score}(S) = \frac{\sum_{w \in TS} \textit{tf-idf}(w)}{|\ \textit{meaningful words}\ |}$$

Redundancy reduction

Rescore sentence list according to similarity with already selected sentences:

Redundancy reduction

Rescore sentence list according to similarity with already selected sentences:

$$NewScore(S_i) = Score(S_i) \times (1 - Sim(S_i, LS))$$

Topic signature example

```
nausherwani
rebel
sporadic
rape
tribal
pakistan
people
rocket
cheema
left
gas
tribesman
```

Summary example

```
Lasi said Sunday that about 5,000 Bugti tribesmen have
taken up positions in mountains near Dera Bugti.
Dera Bugti lies about 50 kilometers (30 miles) from
Pakistan's main gas field at Sui.
Baluchistan was rocked by a tribal insurgency in the 1970s
and violence has surged again this year.
The tribesmen have reportedly set up road blocks and dug
trenches along roads into Dera Bugti.
Thousands of troops moved into Baluchistan after a rocket
barrage on the gas plant at Sui left eight people dead
in January.
"We have every right to defend ourselves," Bugti told AP
by satellite telephone from the town.
```

Outline

ROUGE scores

	R	P	F
ROUGE-1	0.25909	0.30675	0.27987
ROUGE-2	0.06453	0.07577	0.06942
ROUGE-3	0.01881	0.02138	0.01992
ROUGE-4	0.00724	0.00774	0.00745

Further improvements

- try new sentence scoring methods
  - LLR
  - sentence position
  - deep methods

Further improvements

- try new sentence scoring methods
  - LLR
  - sentence position
  - deep methods
- use a classification approach for sentence selection

Summarization Task

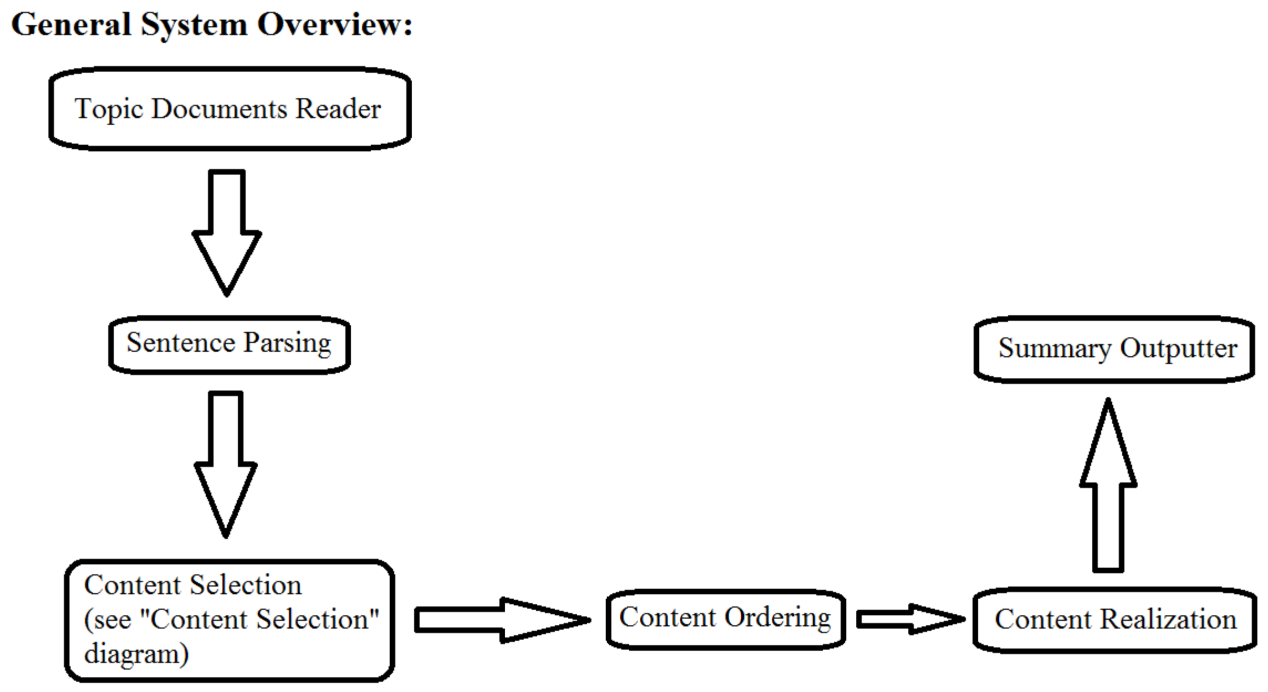
LING 573

Team Members

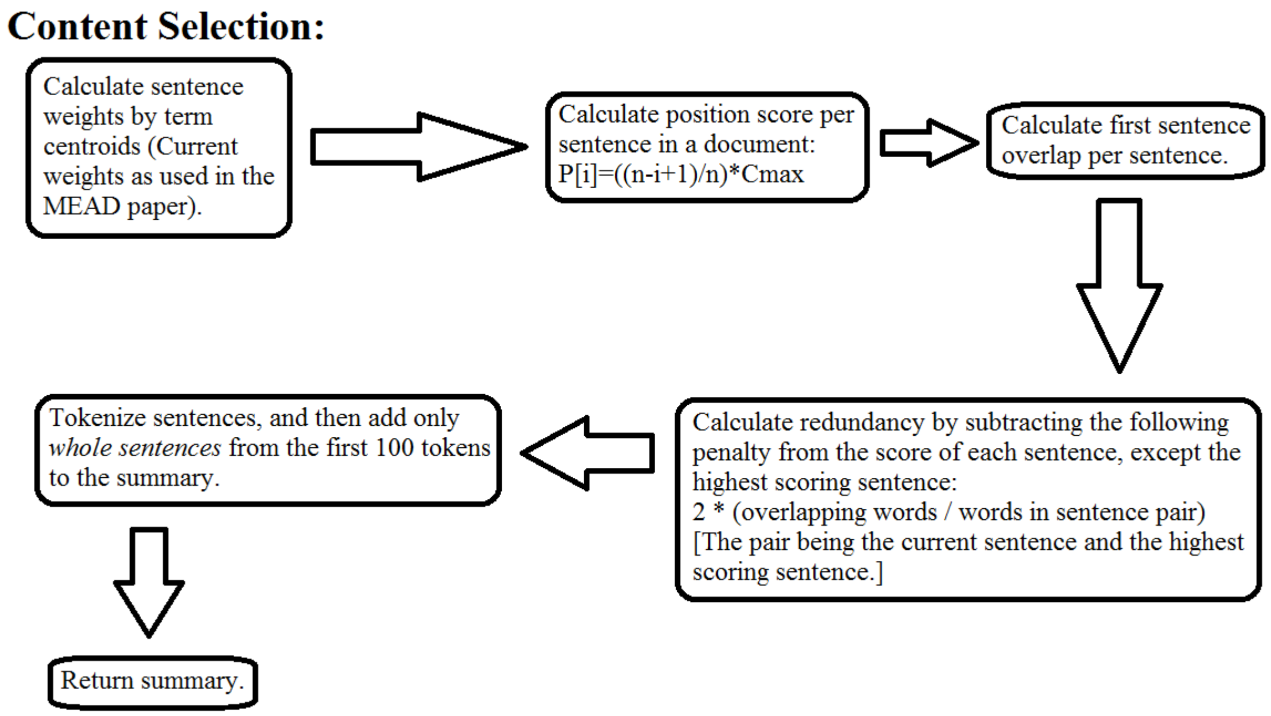
- John Ho
- Nick Chen
- Oscar Castaneda

Contents

System Architecture



Content Selection



Content Selection:

Calculate sentence weights by term centroids (Current weights as used in the MEAD paper).

Calculate position score per sentence in a document: P[i]=((n-i+1)/n)*Cmax

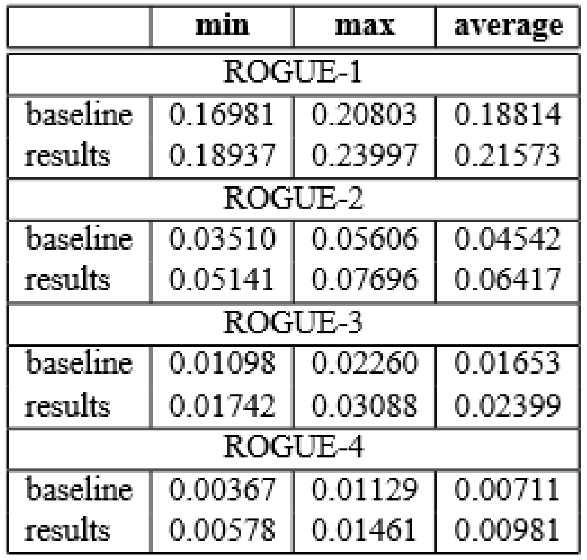
Calculate first sentence overlap per sentence.

Calculate redundancy by subtracting the following penalty from the score of each sentence, except the highest scoring sentence:
2 * (overlapping words / words in sentence pair)
[The pair being the current sentence and the highest scoring sentence.]

Tokenize sentences, and then add only *whole sentences* from the first 100 tokens to the summary.

Return summary.

Current Results



	min	max	average
ROGUE-1			
baseline	0.16981	0.20803	0.18814
results	0.18937	0.23997	0.21573
ROGUE-2			
baseline	0.03510	0.05606	0.04542
results	0.05141	0.07696	0.06417
ROGUE-3			
baseline	0.01098	0.02260	0.01653
results	0.01742	0.03088	0.02399
ROGUE-4			
baseline	0.00367	0.01129	0.00711
results	0.00578	0.01461	0.00981

Table 1: ROGUE scores.

Sample output

- The sheriff's initial estimate of as many as 25 dead in the Columbine High massacre was off the mark apparently because the six SWAT teams that swept the building counted some victims more than once.
- Sheriff John Stone said Tuesday afternoon that there could be as many as 25 dead.
- The discrepancy occurred because the SWAT teams that picked their way past bombs and bodies in an effort to secure building covered overlapping areas, said sheriff's spokesman Steve Davis.
- "There were so many different SWAT teams in there, we were constantly getting different counts," Davis said.

Topic?

Redundant

Redundant

96 words

Successes

- The pipeline works end to end and is built with a model in which we can easily plug in new parts to it
- The content selection step selects important sentences
- The project reuses code libraries from external resources that have been proved to work
- Evaluation results are consistent with our expectations for the first stage of the project

Issues

- Processing related (Solved now):
  - Non-standard XML
  - Inconsistent naming scheme
  - Inconsistent formatting
- Summarization related (Need to be solved):
  - ROUGE scores still low
  - Need to test content selection
  - Need to tune content selection
  - Need to improve our content ordering and content realization pipeline
  - Duplicated content
  - Better topic surfacing

References and Resources

- Dragomir R. Radev, Sasha Blair-Goldensohn, and Zhu Zhang. 2004. Experiments in Single and MultiDocument Summarization Using MEAD University Of Michigan

- Scikit-learn: Machine Learning in Python, Pedregosa et al., (2011). JMLR 12, pp. 2825-2830, 2011

- Steven Bird, Edward Loper and Ewan Klein (2009). Natural Language Processing with Python.. OReilly Media Inc.

P.A.N.D.A.S.

(Progressive Automatic Natural Document Abbreviation System)

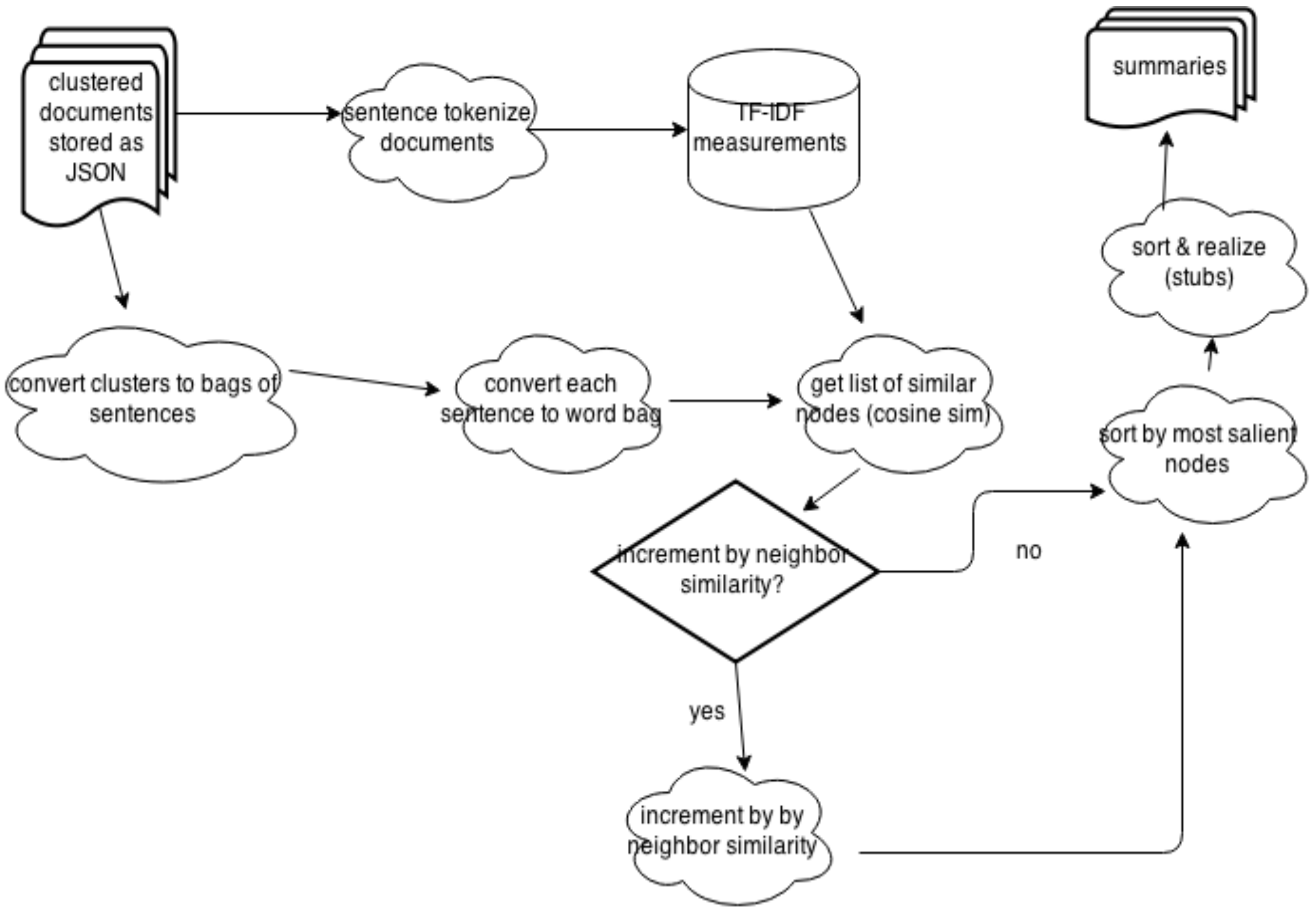
Ceara Chewning, Rebecca Myhre, Katie Vedder

Related Reading

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.

+

System Architecture


clustered documents stored as JSON
sentence tokenize documents
TF-IDF measurements
summaries
convert clusters to bags of sentences
convert each sentence to word bag
get list of similar nodes (cosine sim)
sort & realize (stubs)
sort by most salient nodes
increment by neighbor similarity?
no
yes
increment by by neighbor similarity

+
Results

	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Top N	0.21963	0.05173	0.01450	0.00461
Random	0.16282	0.02784	0.00812	0.00334
MEAD	0.22641	0.05966	0.01797	**0.00744**
PANDAS	**0.24886**	**0.06636**	**0.02031**	0.00606

+

Content Selection

- Graph-based, lexical approach
- IDF-modified cosine similarity equation (Erkan and Radev, 2004):

$$sim_{x,y} = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}}$$

- Sentences scored by degree of vertex
- Redundancy accounted for with a second threshold

+
Information Ordering

- Nothing fancy
- Sentences ordered by decreasing saliency

+
Content Realization

- Nothing fancy
- Sentences realized as they appeared in the original document

+

Issues:

- More sophisticated node scoring method was unsuccessful
  - "Social networking" approach (increasing score of a node based on degree of neighboring nodes) significantly impacted ROUGE scores
  - Scored nodes by degree instead

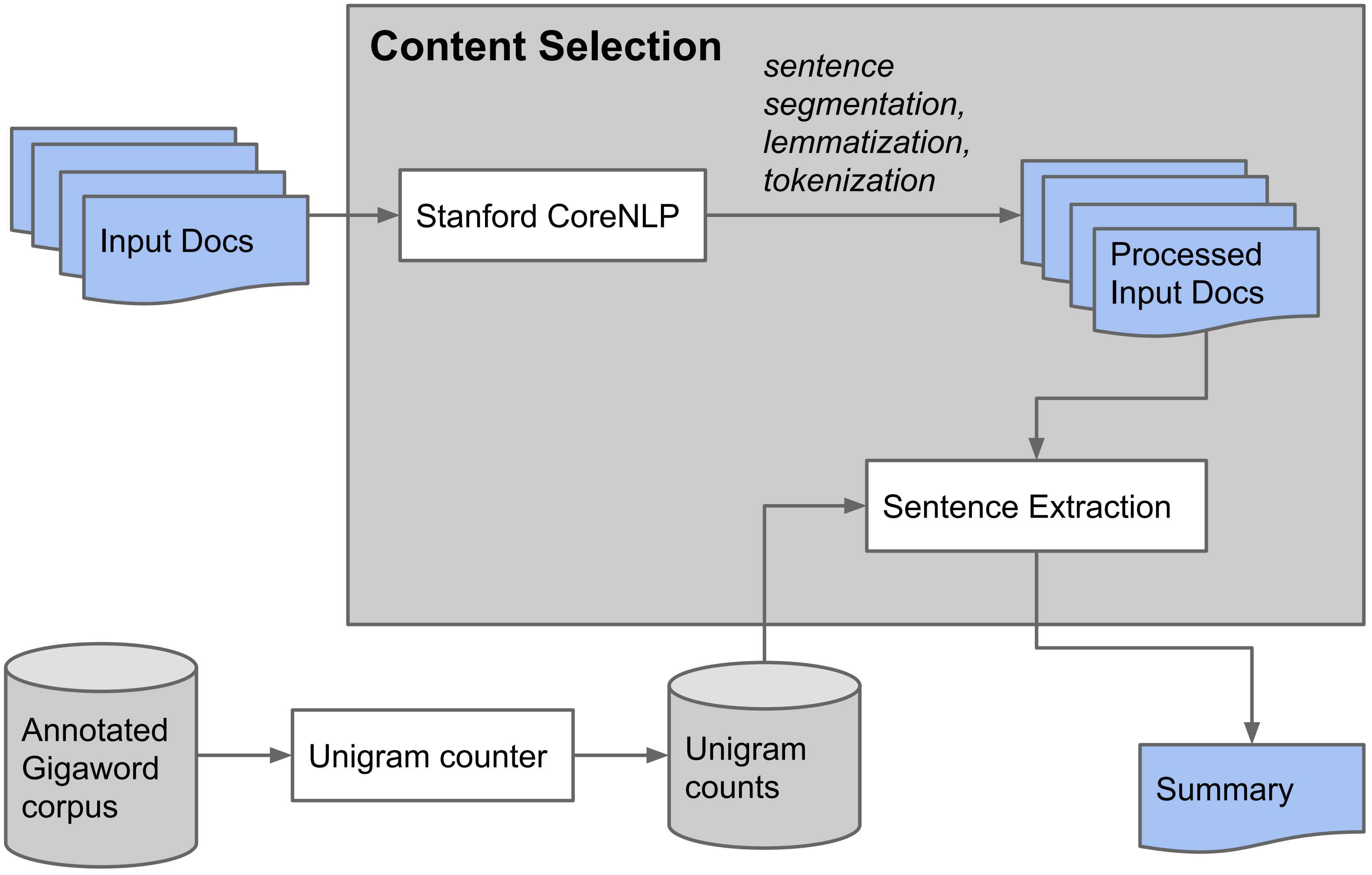
Successes

- Redundancy threshold worked well, based on manual evaluation
  - Depressed ROUGE-3 and ROUGE-4 scores

LING 573 Deliverable #2

George Cooper, Wei Dai, Kazuki Shintani

System Overview



Content Selection

Algorithm Overview

- Modeled after KLSum algorithm
- Goal: Minimize KL Divergence/maximize cosine similarity between summary and original documents
- Testing every possible summary is $O(2^n)$, so we used a greedy algorithm

Algorithm Details

- Start with an empty summary M
- Select the sentence S that has not yet been selected that maximizes the similarity between M + S and the whole document collection
- Repeat until no more sentences can be added without violating the length limit

Vector Weighting Strategies

Creating vectors: Raw Counts

Each element of the vector corresponds to the unigram count of the document/sentence as lemmatized by Stanford CoreNLP.

Creating vectors: TF-IDF

Weight raw counts using a variant of TF-IDF:

$$(n_v/N_v)\log(N_c/n_c)$$

- n_v: raw count of the unigram in the vector
- N_v: total count of all unigrams in the vector
- n_c: raw count of the unigram in the background corpus (Annotated Gigaword)
- N_c: total count of all unigrams in the background corpus

Creating vectors: Log-likelihood ratio

- Weight raw counts using log-likelihood ratio
- We used Annotated Gigaword corpus as the background corpus

Creating vectors: Normalized log-likelihood ratio

- Weight the vector for the whole document collection using log-likelihood
- Weight each item in individual sentences as $w_b(w_s/n_s)$
  - w_b: weight of the item in the background corpus
  - w_s: raw unigram count in sentence vector
  - n_s: total of all unigram counts in the sentence vector
- Intended to correct preference for shorter sentences

Filtering stop words

- 85 lemmas
- Manually compiled from the most common lemmas in the Gigaword corpus
- Stop words ignored when creating all vectors

Results

Results: Stop words filtered out

Comparison	Weighting	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
KL divergence	raw counts	0.28206	0.07495	0.02338	0.00777
KL divergence	TF-IDF	0.28401	0.07636	0.02440	0.00798
KL divergence	LL	0.29039	0.08304	0.02889	0.00984
KL divergence	LL (normalized)	0.27824	0.07306	0.02268	0.00746
cosine similarity	raw counts	0.28232	0.07336	0.02114	0.00686
cosine similarity	TF-IDF	0.28602	0.07571	0.02305	0.00758
cosine similarity	LL	0.26698	0.06646	0.01976	0.00632
cosine similarity	LL (normalized)	0.27016	0.06603	0.01946	0.00604

Results: Stop words not filtered out

Comparison	Weighting	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
KL divergence	raw counts	0.24185	0.06338	0.02266	0.00778
KL divergence	TF-IDF	0.25736	0.06790	0.02301	0.00814
KL divergence	LL	0.28682	0.08110	0.02716	0.00875
KL divergence	LL (normalized)	0.27813	0.07283	0.02248	0.00718
cosine similarity	raw counts	0.18632	0.04202	0.01345	0.00450
cosine similarity	TF-IDF	0.24422	0.05887	0.01918	0.00612
cosine similarity	LL	0.26686	0.06694	0.02031	0.00655
cosine similarity	LL (normalized)	0.26842	0.06525	0.01929	0.00604

Discussion

Issues

- We need to remove the dateline (e.g. SEATTLE (AP)) as a preprocessing step
- There is too much redundancy in some of the summaries (no explicit method to handle redundancy in our approach yet)
- The last sentence is often very short and not useful

Potential Improvements

- Expand the search space a little
- Replace pronouns with their referents as a preprocessing step
- Take advantage of similarities, particularly synonyms, between different words using WordNet or word embeddings for better comparison of vectors

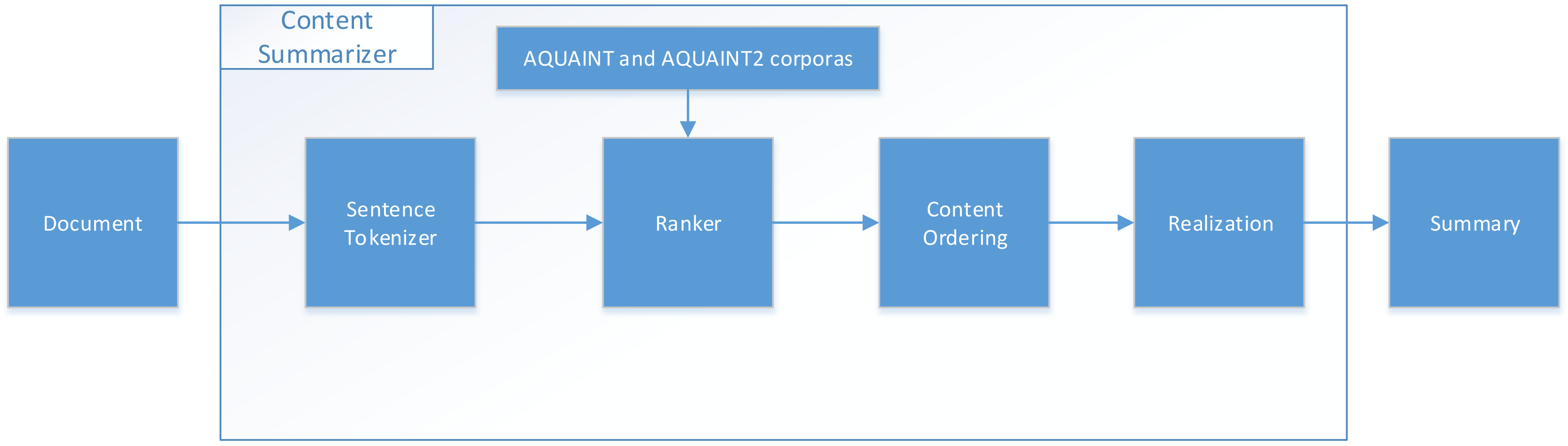
Baseline summarization system

Veljko Miljanic –Abdelrahman Baligh - Ahmed Aly
4/28/2015

Introduction

- End to end document summarization system
- We have approached extractive summarization as sentence ranking problem
- We want to build ML sentence ranker that can combine variety of features
- Baseline rankers
    - lead
    - log likelihood
- Content ordering (placeholder): output sentences in order of their rank
- Text realization (placeholder): concatenate top sentences

System architecture



System architecture (cont.)

Sentence tokenizer:

We are using NLTK sentence tokenizer to split documents into sentences.

Ranker:

- We plan on building ML ranker to be able to combine variety of features:

    log likelihood ratio, position of sentence in document, …

- Pointwise ranker: regression target will be sentence ROUGE score

- Pairwise ranker: classifier target generated based on sentence order by their ROUGE scor

System architecture (cont.)

Log Likelihood Ratio Baseline:

-We’ve used log likelihood scores to serve as our baseline ranking scheme

- Sentences are ranked by LLR weights and we pick up top N that fit into the summary size

- Background corpora is union of the entire AQUAINT and AQUAINT2 corpora

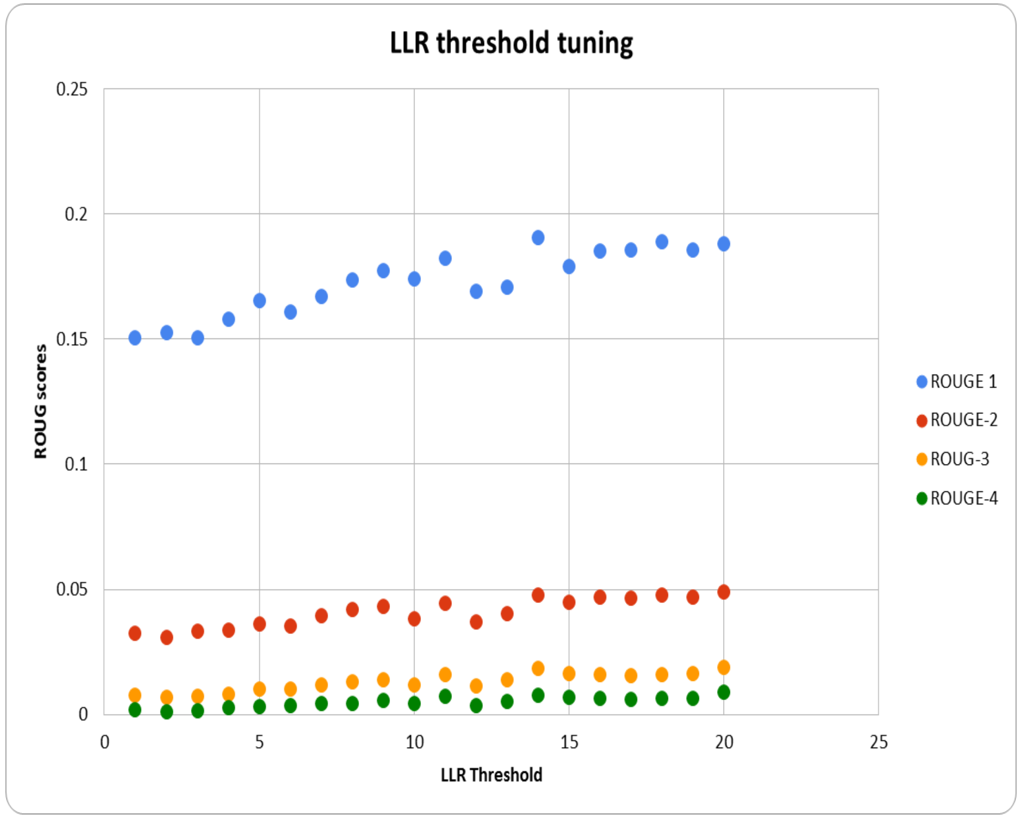
- All words converted to lowercase prior to computing LLR

- LLR threshold is tuned on devtest set (14)

System architecture (cont.)

Ranker(LLR): Threshold Tuning

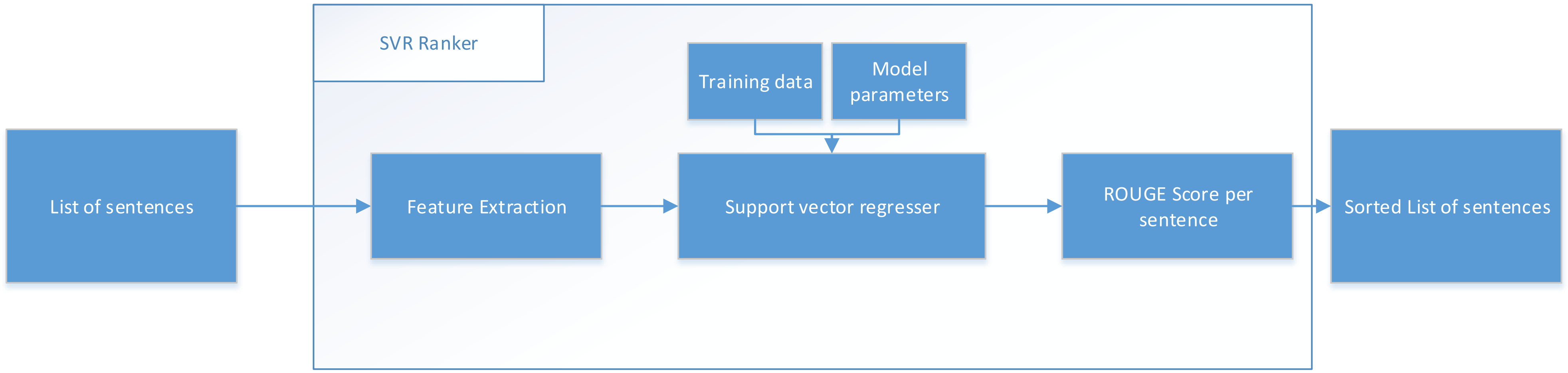
THR	ROUGE 1	ROUGE-2	ROUGE-3	ROUGE-4
1	0.15062	0.03245	0.00777	0.0021
2	0.15266	0.03078	0.00683	0.00118
3	0.15039	0.03333	0.0073	0.00137
4	0.15801	0.0337	0.00803	0.00264
5	0.1656	0.03601	0.01001	0.00309
6	0.16077	0.03532	0.01034	0.00349
7	0.16723	0.03935	0.01202	0.00438
8	0.17387	0.04192	0.01286	0.00439
9	0.17721	0.04323	0.01394	0.00557
10	0.17407	0.03833	0.01188	0.0043
11	0.18244	0.04457	0.01581	0.00732
12	0.16909	0.03716	0.01135	0.0037
13	0.1706	0.04029	0.01408	0.00531
14	0.19069	0.0478	0.01825	0.0078
15	0.17903	0.04466	0.01626	0.00674
16	0.18541	0.04679	0.01579	0.00628
17	0.18557	0.04663	0.01559	0.00609
18	0.18899	0.04771	0.01611	0.00643
19	0.18584	0.04675	0.01624	0.00654
20	0.18794	0.04887	0.01871	0.00905



System architecture (cont.)

Ranker (SVR):

- We started our experiments by trying to train a support vector machine regresser to estimate the ROUGE scores of sentences and sort them accordingly.
- We are using scikit-learn as our ML toolkit.



System architecture (cont.)

Ranker (SVR):
- We are still working on this ranker as we are having some issues with the convergence of the regression algorithm.

- Another approach that we are still working on is to train a supervised classifier to pairwise compare sentences and produce a sorted list of sentences according to their importance.

- For the next deliverable we will be working on extending our features and try different regression algorithms

System architecture (cont.)

Information ordering:

-For now, sentences are ordered in a descending order according to their ranker scores

Content Realization:

-We just join the top sentences with a new line separator in between them.

Results

Lead sentences baseline information (just taking the first n sentences from the first document in the docset)

```
1 ROUGE-1 Average_R: 0.18369 (95%-conf.int. 0.15940 - 0.20823)
1 ROUGE-2 Average_R: 0.05075 (95%-conf.int. 0.04034 - 0.06183)
1 ROUGE-3 Average_R: 0.01859 (95%-conf.int. 0.01317 - 0.02523)
1 ROUGE-4 Average_R: 0.00666 (95%-conf.int. 0.00371 - 0.01036)
```

LLR ranker results

```
1 ROUGE-1 Average_R: 0.19069 (95%-conf.int. 0.16378 - 0.21615)
1 ROUGE-2 Average_R: 0.04780 (95%-conf.int. 0.03693 - 0.05908)
1 ROUGE-3 Average_R: 0.01825 (95%-conf.int. 0.01261 - 0.02485)
1 ROUGE-4 Average_R: 0.00780 (95%-conf.int. 0.00356 - 0.01324)
```

Issues

-Most of the work went on reading AQUAINT and AQUAINT-2 corpora because data is inconsistent and also format between corpora is different. AQUAINT can't be read with XML parser while AQUAINT-2 could

-The SVR regresser didn’t converge, that is mainly because we haven’t yet extracted enough features. (We will be working on this one for the next deliverable)

-We haven’t yet implemented filtering for text that usually isn’t part of summary (e.g. citations)

-For most of the summaries we are seeing duplicate sentences. We are still working on a module that would prevent similar sentences to show up in the summary