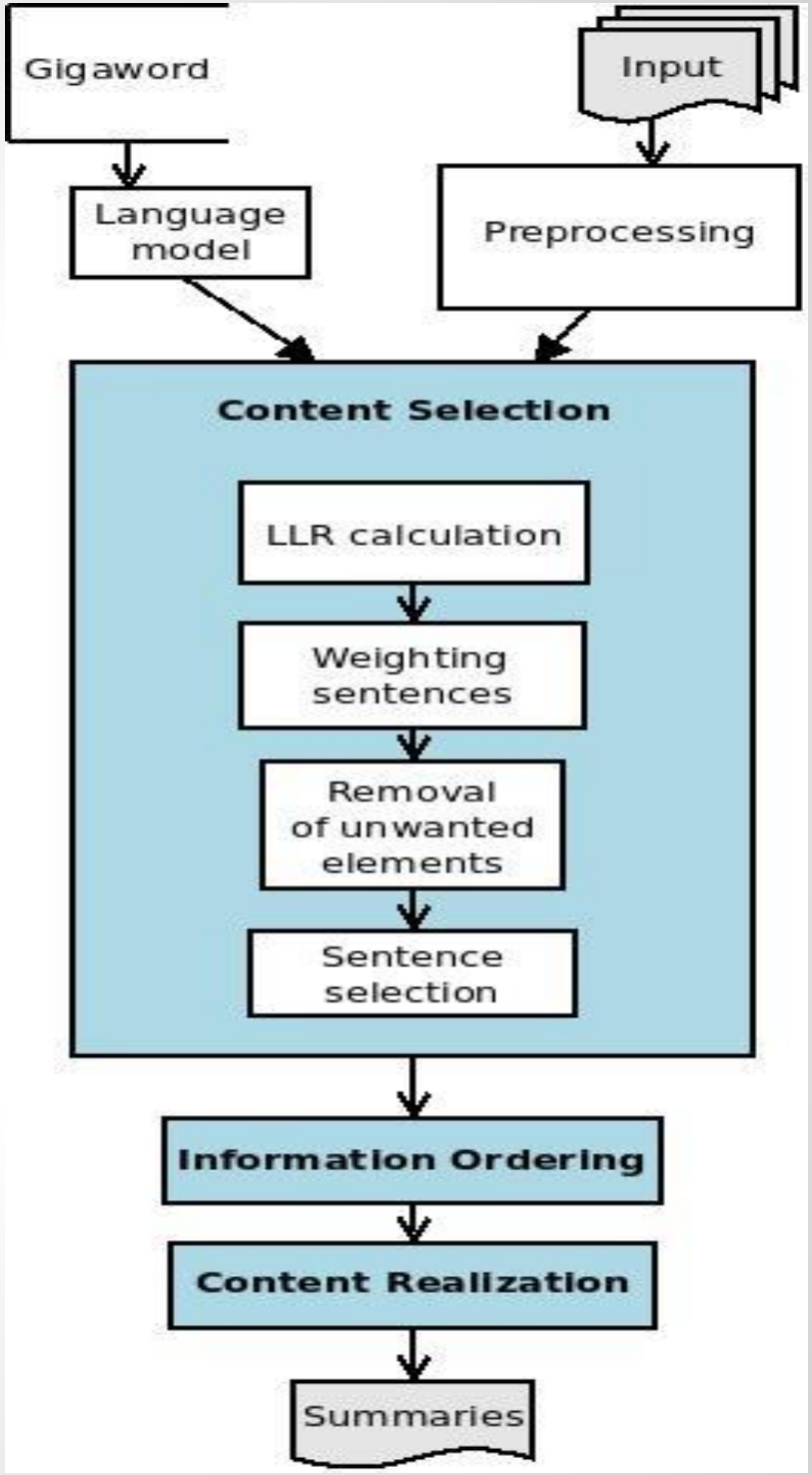
Automatic Summarization Project

Ling573 - Deliverable 2

Eric Garnick
John T. McCranie
Olga Whelan

System Architecture

- Extract document text + meta-data, store in Python data structures, save externally in pickles
- Weight and process sentences
- Select best dissimilar sentences
- Assemble summary

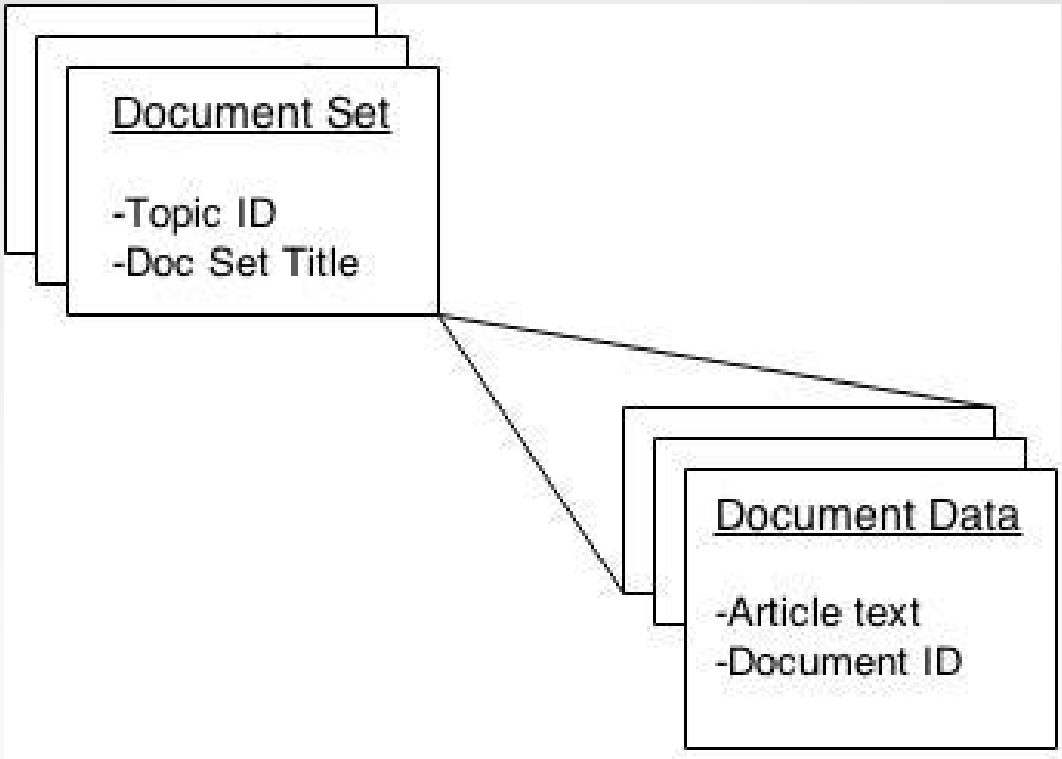


Background Corpus

- Gigaword corpus 5th Ed. ~ 26 GB text
- whitespace tokenize for alphanumeric characters
- Filter stopwords
- 6,295,429 tokens, 163,146 types
- record unigram counts

Text Extraction

- Find and save target document from file
  - regular expressions
  - string matching
- Clean xml with ElementTree
  - Save plain text
  - Save meta-data (topic-ids, titles, doc-ids)

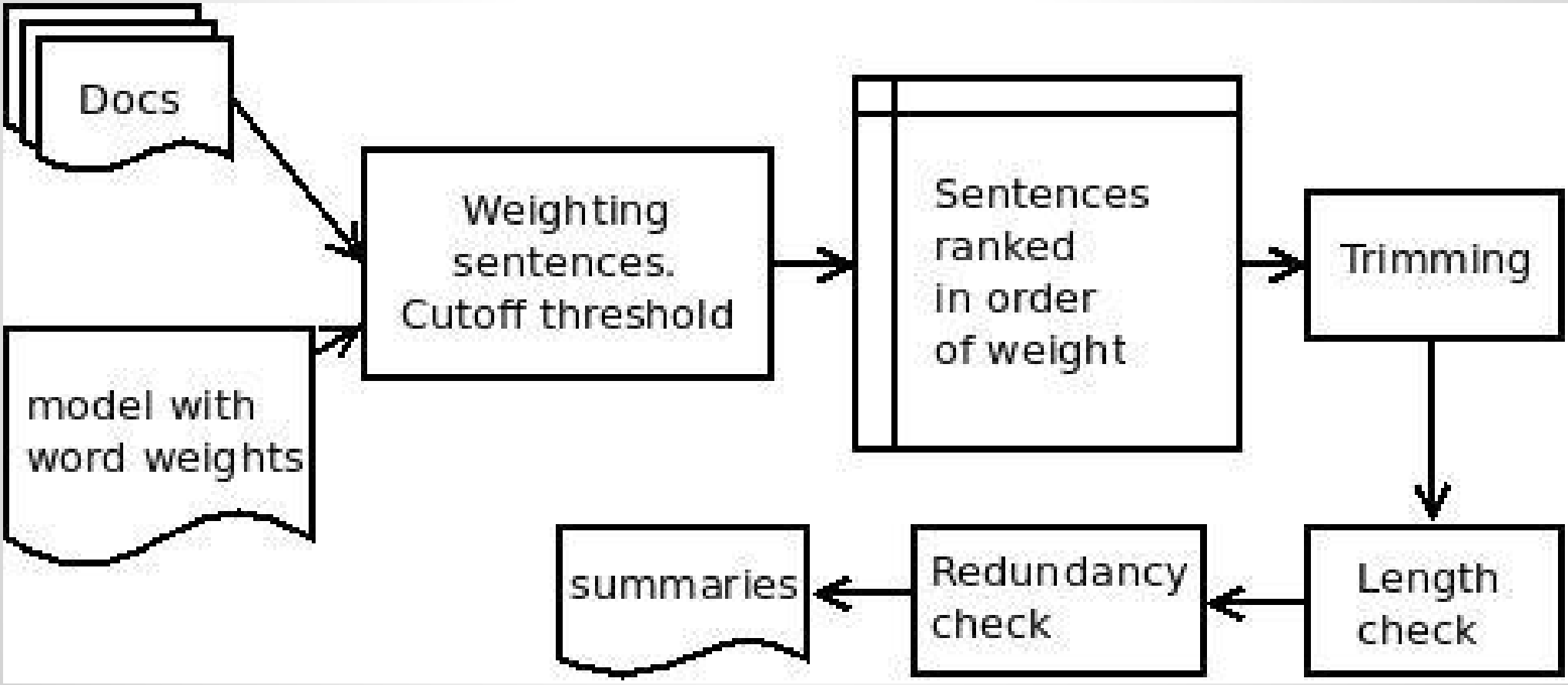


Input Pre-Processing

- Sentence-split with NLTK sentence tokenizer

Content Selection

1. LLR weighting
2. Remove extraneous tokens
3. Check length
4. Check sentence overlap with existing summary



LLR Calculation

$$\lambda(w_i) = \frac{\text{word occurs equally in target text and in the wild}}{\text{word occurrence is unequal in both environments}}$$

1. Compare counts for word in target text and background corpus
2. w_i = -2 log λ (w_i) – score for word w_i
3. Sentence weight is count of words in sentence with LLR score > 10 normalized by sentence length.

Sentence Filtering

- Remove extraneous tokens
  - Common forms of contact information
  - Uninformative “phrases”
  - Common non-alphanumeric “tokens”
- Keep relatively long sentences (> 8 words)
- Check word overlap with existing summary sentences
  - Simple cosine similarity score
  - Omit if similarity > 0.5

Info Ordering / Content Realization

- arrangement follows document order by doc ID (time stamp)
- intra-document order disregarded

- sentences realized as they appear in the document or in whatever form they take after shortening

Results

Lead:

System	Precision	Recall	F
ROUGE-1	0.21170	0.19181	0.20061
ROUGE-2	0.04912	0.04469	0.04661
ROUGE-3	0.01411	0.01270	0.01331
ROUGE-4	0.00373	0.00341	0.00355

LLR + processing:

System	Precision	Recall	F
ROUGE-1	0.25619	0.26501	0.25988
ROUGE-2	0.07232	0.07484	0.07338
ROUGE-3	0.02379	0.02491	0.02428
ROUGE-4	0.00962	0.01029	0.00993

Analysis and Issues

We have given priority to the afforestation in the habitats.

Shaanxi has so far established 13 giant pandas protection zones and nature reserves focused on pandas' habitats.

The Qinling panda has been identified as a sub-species of the giant panda that mainly resides in southwestern Sichuan province.

Nature preserve workers in northwest China's Gansu Province have formulated a rescue plan to save giant pandas from food shortage caused by arrow bamboo flowering.

Currently more than 1,500 giant pandas live wild in China, according to a survey by the State Forestry Administration.

- Ordering of sentences affects the impression
- Non-coreferred pronouns are confusing
- Irrelevant information takes up summary space
- Word removal approach relies too much on punctuation

Resources

- basic design, LLR calculation:
  - Jurafsky & Martin, 2008
- filtering sentences by length, checking sentence similarity:
  - Hong & Nenkova, 2014
- computing LLR with Gigaworld:
  - Parker & al., 2011

Future Work

Content Selection

- coreference resolution - CLASSY (Conroy et al., 2004)
- sentence position

Information Ordering

- clustering sentences based on similarity (word overlap and other semantic similarity measures)

Document Summarization

LING 573, Spring 2015

Jeff Heath
Michael Lockwood
Amy Marsh

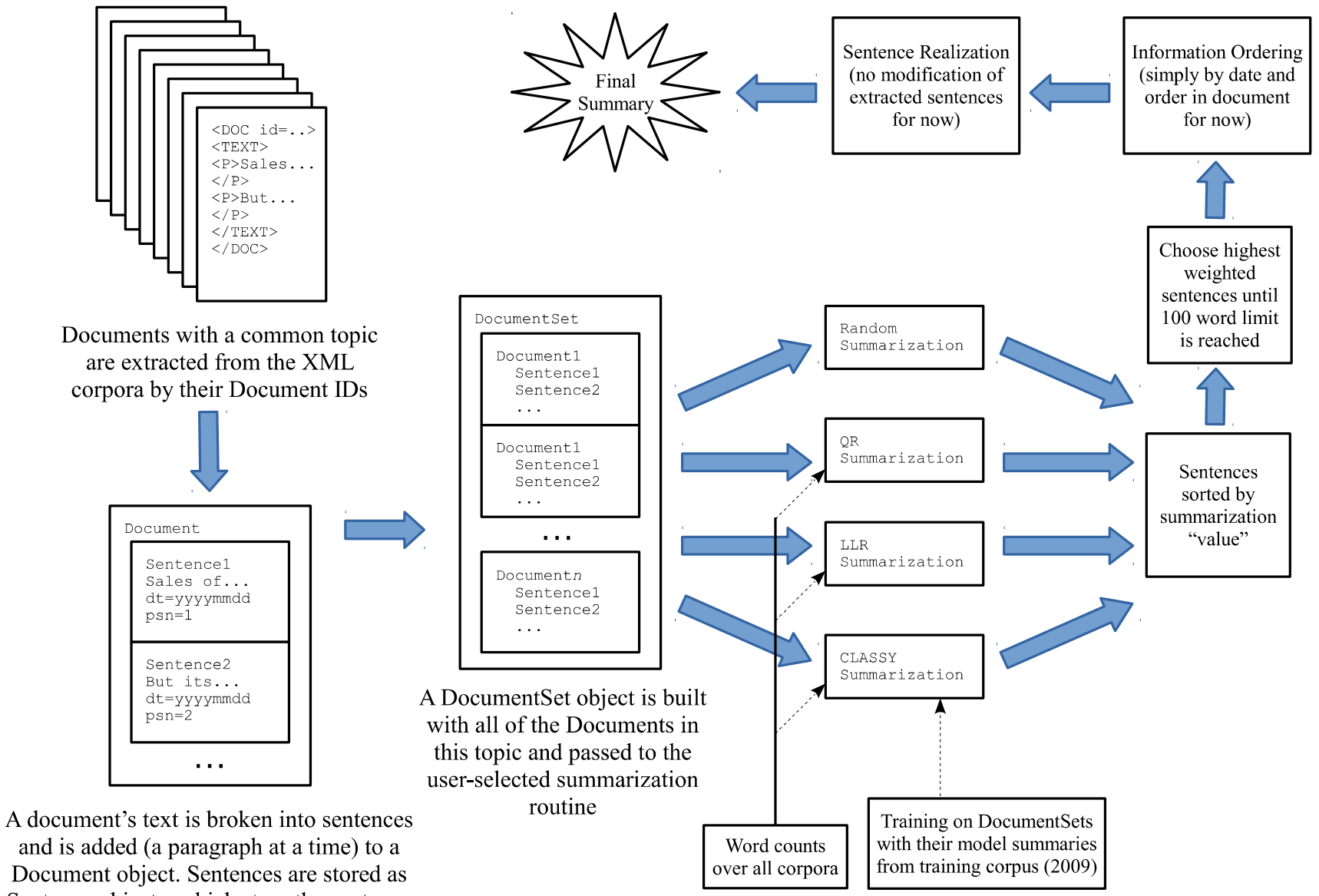
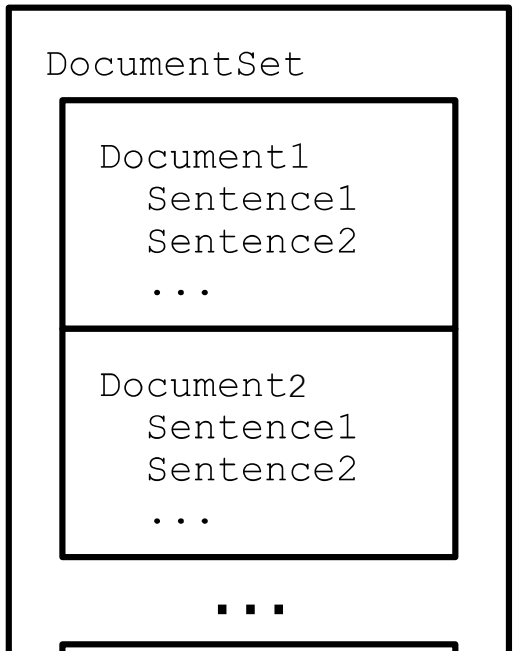
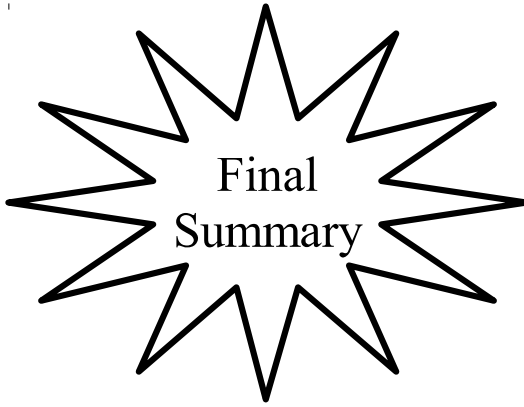
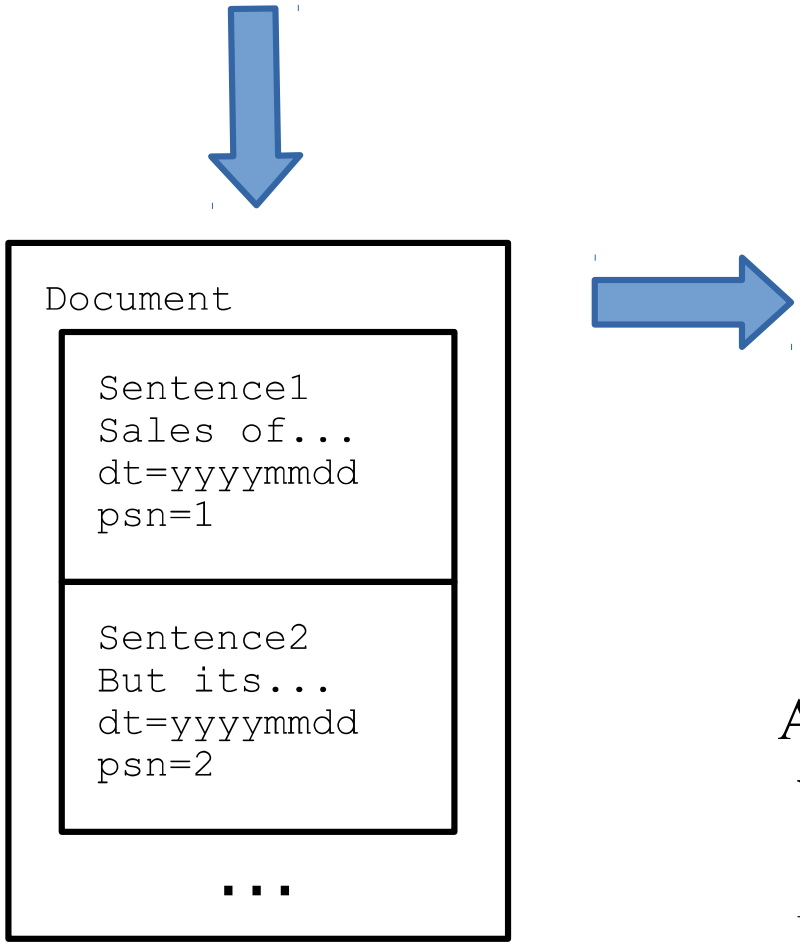


Figure 1: Summarization System Architecture

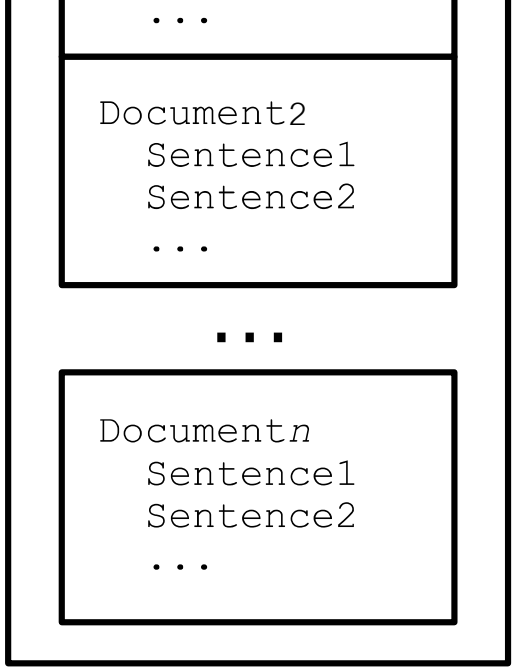
Documents with a common topic are extracted from the XML corpora by their Document IDs







A document's text is broken into sentences and is added (a paragraph at a time) to a Document object. Sentences are stored as Sentence objects, which store the sentence string, a tokenized list of words in the sentence, the document date, the position in the paragraph – everything that might help us in our summarization



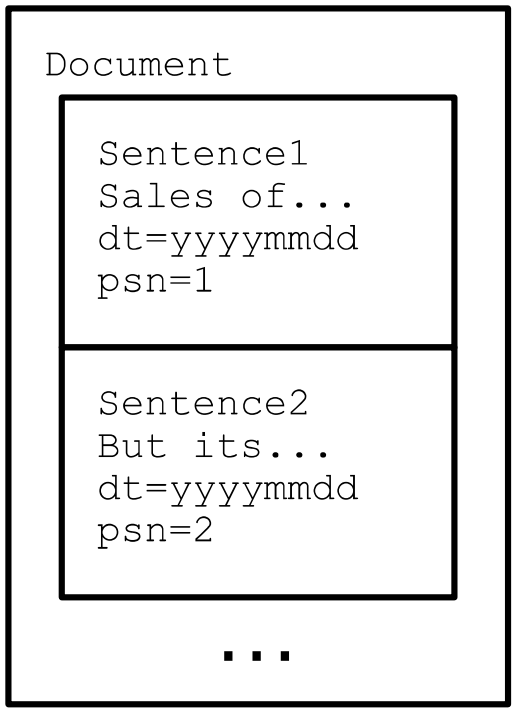
A DocumentSet object is built with all of the Documents in this topic and passed to the user-selected summarization routine

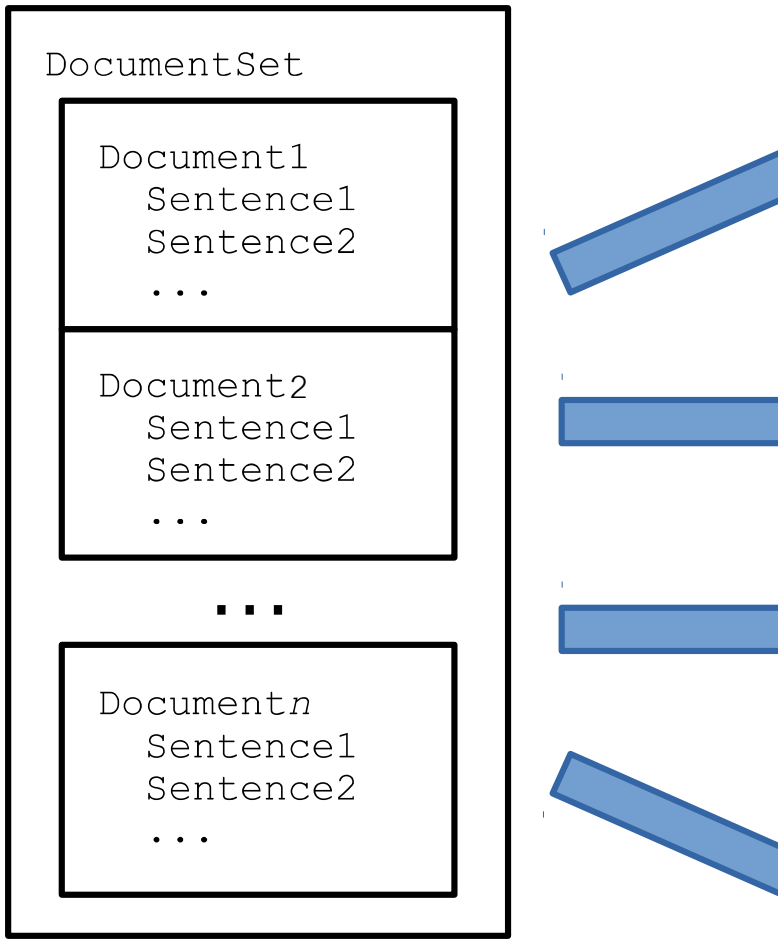
Word
over al

Figure 1: Summarization Sys

</TEXT>
</DOC>

Documents with a common topic are extracted from the XML corpora by their Document IDs

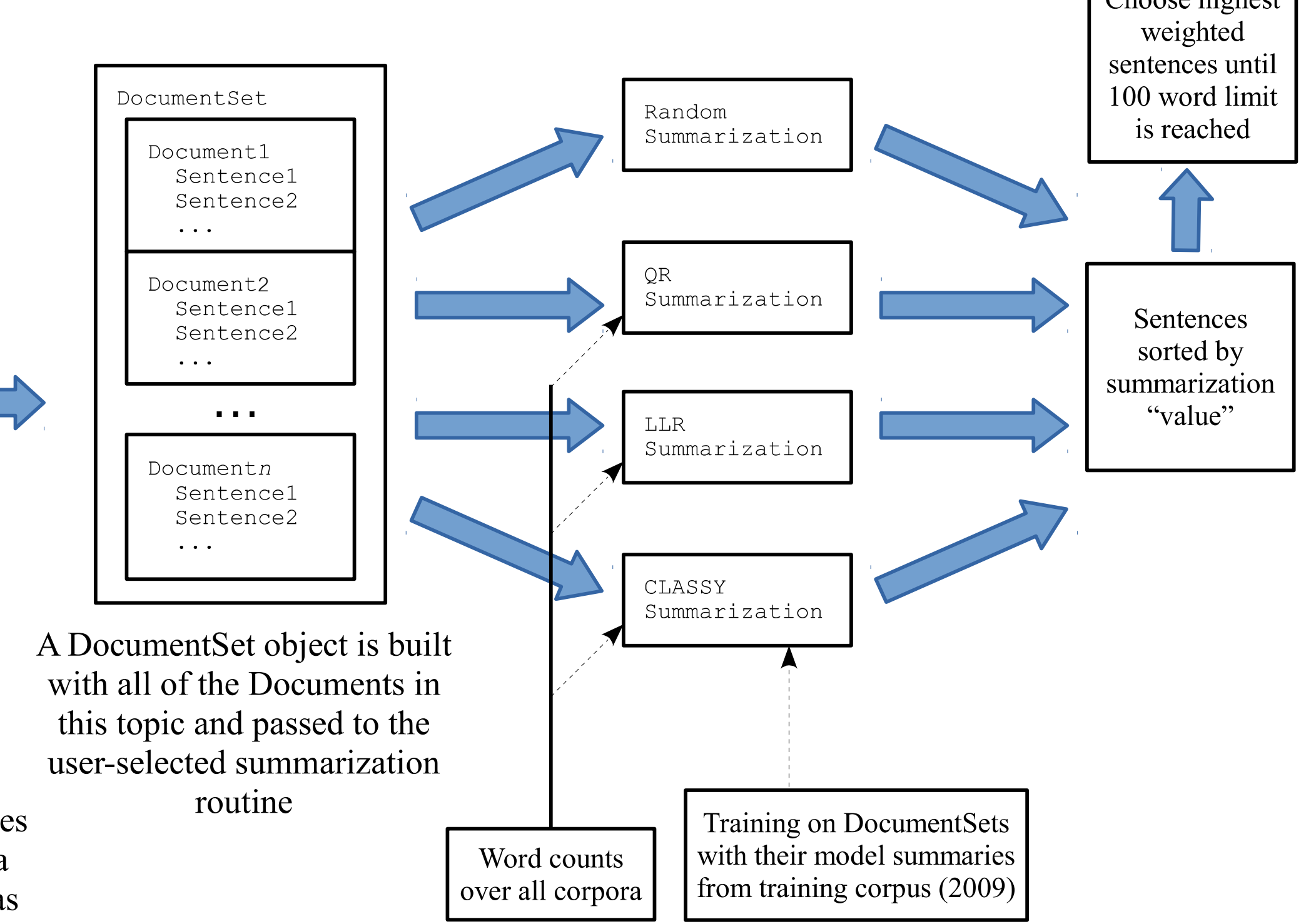


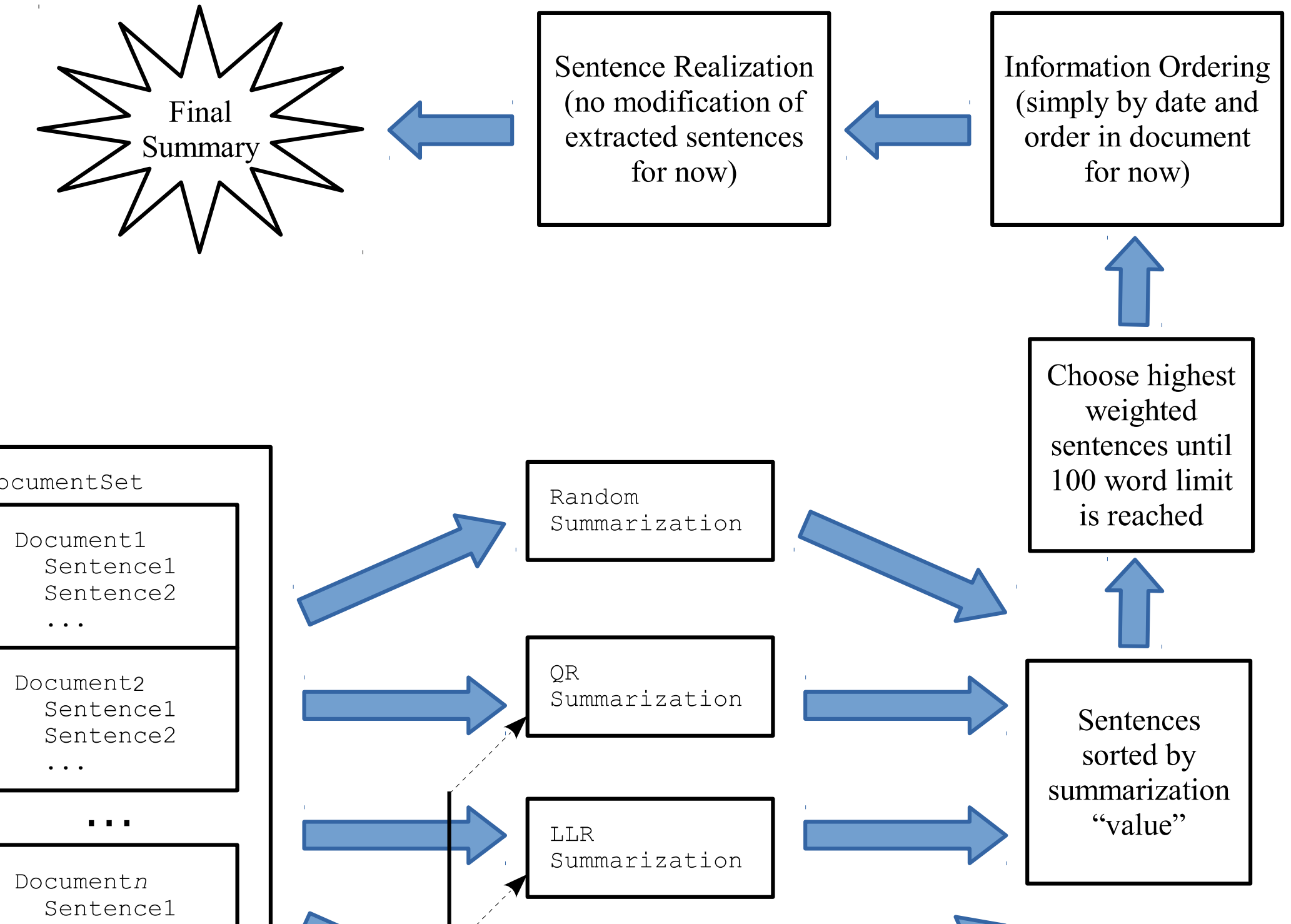


A DocumentSet object is built with all of the Documents in this topic and passed to the user-selected summarization routine

A document's text is broken into sentences and is added (a paragraph at a time) to a

Word


DocumentSet
Document1
Sentence1
Sentence2
...
Document2
Sentence1
Sentence2
...
...
Documentn
Sentence1
Sentence2
...
Random Summarization
QR Summarization
LLR Summarization
CLASSY Summarization
Choose highest weighted sentences until 100 word limit is reached
Sentences sorted by summarization "value"
A DocumentSet object is built with all of the Documents in this topic and passed to the user-selected summarization routine
Word counts over all corpora
Training on DocumentSets with their model summaries from training corpus (2009)


Final Summary
Sentence Realization (no modification of extracted sentences for now)
Information Ordering (simply by date and order in document for now)
Choose highest weighted sentences until 100 word limit is reached
Sentences sorted by summarization "value"
ocumentSet
Document1
Sentence1
Sentence2
...
Document2
Sentence1
Sentence2
...
...
Documentn
Sentence1
Random Summarization
QR Summarization
LLR Summarization

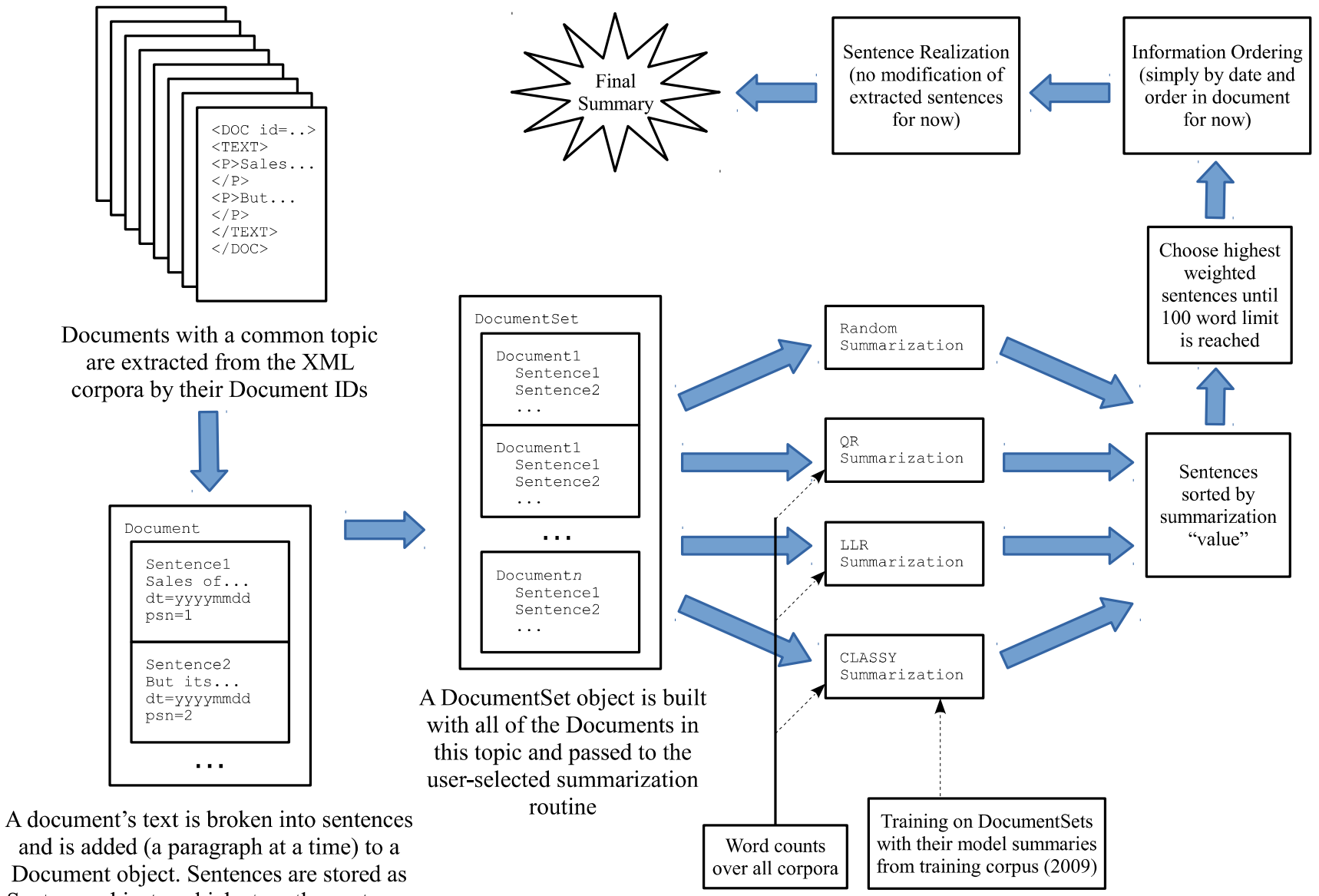


Figure 1: Summarization System Architecture

Random Baseline

ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
0.15323	0.02842	0.00654	0.00256

CLASSY Overview

- Hidden Markov Model trained on features of summary sentences of training data
- Used to compute weights for each sentence in test data
- Select sentences with highest weights
- QR Matrix Decomposition used to avoid redundancy in selected sentences

Log Likelihood Ratio

- Find words that are significantly more likely to appear in this document cluster compared to background corpus
- If LLR > 10, word counts as topic signature word
- Sentence score is # of topic signature words/length of sentence
- Cosine similarity to avoid redundancy

Selection Based on LLR

ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
0.28021	0.07925	0.02656	0.01071

QR Matrix Decomposition

- Represent each sentence as a vector
- Conroy and O'Leary (2001): dimensions of vector are open-class words
- We use log likelihood ratio to determine dimensions of vector
- Terms weighted by sentence's position in document:

$$g * \mathrm{e}^{\frac{-8*j}{n}} + t$$

where j = sentence number, n = # of sentences in document, g = 10, t = 3

QR Matrix Decomposition

- Choose sentence (vector) with highest magnitude
- Keep components of remaining sentence vectors that are orthogonal to the vector chosen
- Repeat until you reach 100 word summary

Selection Based on QR Decomposition

ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
0.23280	0.05685	0.01540	0.00380

HMM Training

- Build transition, start, and emission counts
- Turn emissions into covariance matrix/precision matrix
- Record column averages
- Store pickle outputs

HMM Decoding

- Decode class to manage data structures with document set objects
- Process forward and backward recursions
- Observation sequence:
  - Build $(O_t - mu_i)^T \Sigma^{-1} (O_t - mu_i) \rightarrow$ 1 x 1 matrix
  - Apply the χ^2-distribution
  - Subtract from identity

HMM Decoding

- Create ω value from forward recursion
- Calculate γ weight for each sentence
- Final weights from sum of the even states

Selection Based on HMM and QR Decomposition

ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
0.17871	0.04425	0.01729	0.00714

All Results

	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Random	0.15323	0.02842	0.00654	0.00256
LLR	0.28021	0.07925	0.02656	0.01071
QR	0.23280	0.05685	0.01540	0.00380
HMM+QR	0.17871	0.04425	0.01729	0.00714

Future Work

- Need to apply the linguistic elements of CLASSY
- Revise decoding so that forward and backward relatively balance
- Consider updating the features to more contemporary methods
- Further parameter tuning

D2 Summary

Sentence Selection Solution

Brandon Gahler
Mike Roylance
Thomas Marsh

Architecture: Technologies

Python 2.7.9 for all coding tasks

NLTK for tokenization, chunking and sentence segmentation.

pyrouge for evaluation

Architecture: Implementation

Reader:

- Topic parser reads topics and generates filenames
- Document parser reads documents and makes document descriptors

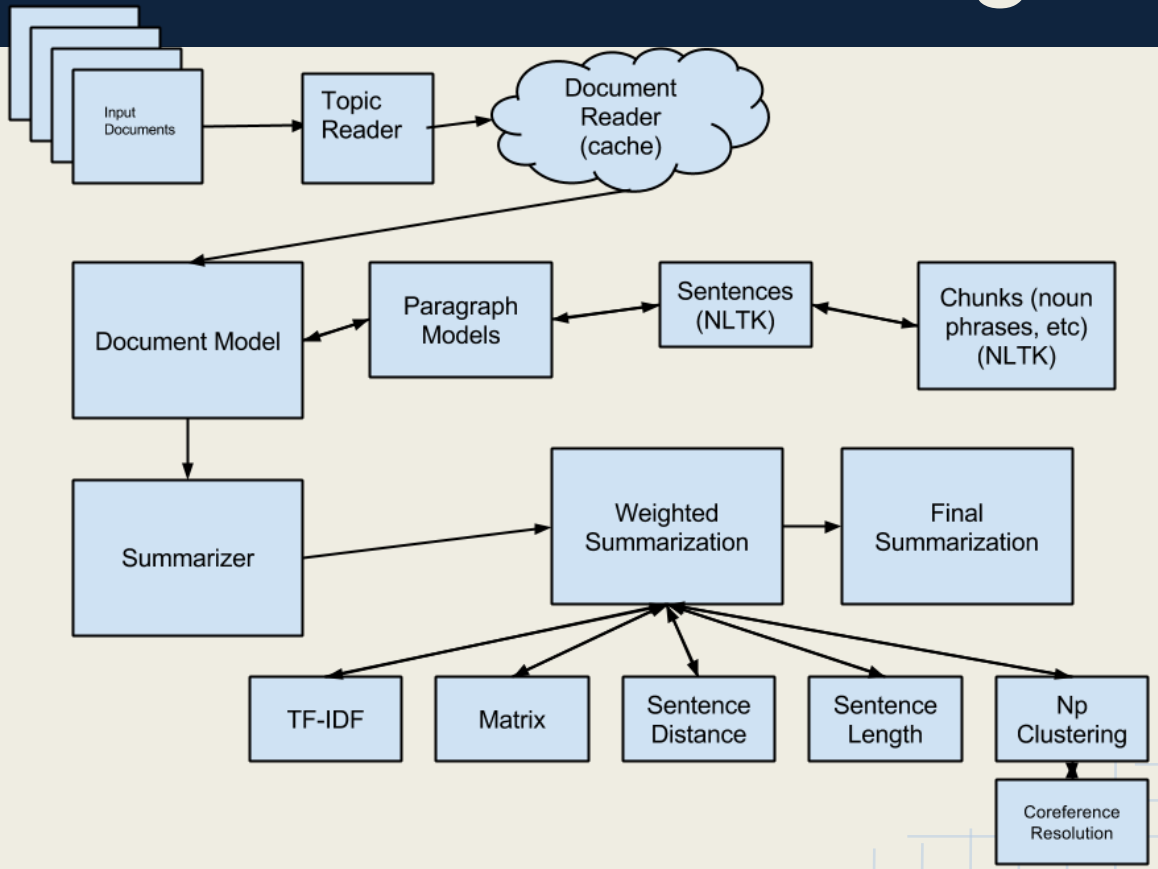
Document Model:

- Sentence Segmentation and “cleaning”
- Tokenization
- NP Chunker

Summarizer - creates summaries

Evaluator - uses pyrouge to call ROUGE-1.5.5.pl

Architecture: Block Diagram



Summarizer

Employed Several Techniques:

Each Technique:

- Computes rank for all sentences normalized from 0 to 1
- Is given a weight from 0 to 1

Weighted sentence rank scores are added together
Overall best sentences are selected from the summary sum

Summary Techniques

- Simple Graph Similarity Measure
- NP Clustering
- Sentence Location
- Sentence Length
- tf*idf

Trivial Techniques

- Sentence Position Ranking - Highest sentences get highest rank
- Sentence Length Ranking - Longest sentences get best rank
- tf*idf - All non-stop words get tf*idf computed and the total is divided by sentence length. Sentences with the highest sum of tf*idf get best rank.
  - We use the Reuters-21578, Distribution 1.0 Corpus of news articles as a background corpus.
  - Scores are scaled so the best score is 1.0

Simple Graph Technique

Iterate:

- Build a fully connected graph of the cosine similarity (non-stopword raw counts) of the sentences
- Compute the most connected sentence
- Give that sentence the highest score
- Change the weights of its edges to negative to discourage redundancy
- recompute

NP-Clustering Technique

Compute the most connected sentences:

- Use coreference resolution:
  - Find all the pronouns, and replace them with their antecedent
- Compare just the noun phrases of each sentence with every other sentence.
  - Use edit distance for minor forgiveness
  - Normalize casing
- Similarity metric is the count of shared noun phrases
- Rank every sentence with between 0-1, with the highest being 1

Technique Weighting

It is difficult to tell how important each technique is in contributing to the overall score. Because of this, we established a **weight generator** which did the following:

for each technique:

- compute unweighted sentence ranks.

- Iterate weights of each technique from 0 to 1 at intervals of 0.1
  - for each weight set:
    - rank sentences based on new weights
    - generate rouge scores

At the end, the best set of weights is the one with the optimal score!

Optimal Weights at Time of Submission

AAANNND... the optimal set of weights turns out to be:

Disappointing!

It looked like none of our fancy techniques were able to even slightly improve the performance of **tf*idf** by itself.

Results?

Average ROUGE scores for our tf*idf-only solution:

ROUGE Technique	Recall	Precision	F-Score
ROUGE1	0.55024	0.52418	0.53571
ROUGE2	0.44809	0.42604	0.43580
ROUGE3	0.38723	0.36788	0.37643
ROUGE4	0.33438	0.31742	0.32490

Results?

Obviously, we had done something wrong. It's pretty unlikely that we got three times better than the best summarizers! We figured out pretty quickly that it was our method of calling rouge, and reran our weight generator.

Optimal Weights Revisited

Hurray! Upon running again, discovered that our hard work had paid off after all! The NP-Clustering technique proved to be the best, followed closely by “equal weight” for every technique.

Optimal Weights

Optimal Technique Weights:

Technique	Weight
tf*idf	0.0
Simple Graph	0.0
NP-Clustering	1.0
Sentence Position	0.0
Sentence Length	0.0

NP-Clustering Results

Average ROUGE scores for the NP-Clustering-only solution:

ROUGE Technique	Recall	Precision	F-Score
ROUGE1	0.23391	0.28553	0.25522
ROUGE2	0.05736	0.07053	0.06272
ROUGE3	0.01612	0.01969	0.01758
ROUGE4	0.00533	0.00657	0.00584

Equal Weight Results

Average ROUGE scores for our “equal weight” solution:

ROUGE Technique	Recall	Precision	F-Score
ROUGE1	0.23336	0.28628	0.25516
ROUGE2	0.05708	0.07044	0.06251
ROUGE3	0.01612	0.01969	0.01758
ROUGE4	0.00533	0.00657	0.00584

Simple Graph Results

Average ROUGE scores for the Simple Graph-only solution:

ROUGE Technique	Recall	Precision	F-Score
ROUGE1	0.19379	0.25550	0.21845
ROUGE2	0.04473	0.05859	0.05033
ROUGE3	0.01170	0.01505	0.01305
ROUGE4	0.00362	0.00453	0.00400

tf*idf Only Results

Average ROUGE scores for our (tf*idf-only) solution:

ROUGE Technique	Recall	Precision	F-Score
ROUGE1	0.15341	0.20846	0.17522
ROUGE2	0.03014	0.04037	0.03426
ROUGE3	0.00746	0.01038	0.00863
ROUGE4	0.00242	0.00329	0.00278

Room for Improvement

- Our individual content selection techniques are simple, and much tuning and improvement remains to be done
  - Implement LLR and compare with tf*idf
  - Test other vector weighting schemes for cosine similarity in Simple Graph technique
  - Merge the Simple Graph style of redundancy reduction into NP Clustering technique
- Move coreference into document model so all content selection techniques and future ordering/realization techniques can take advantage of it

References

Heinzerling, B and Johannsen, A (2014). pyrouge (Version 0.1.2) [Software]. Available from https://github.com/noutenki/pyrouge

Lin, C (2004). ROUGE (Version 1.5.5) [Software]. Available from http://www.berouge.com/Pages/default.aspx

Summarization LING573

RUTH MORRISON
FLORIAN BRAUN
ANDREW BAER

Contents

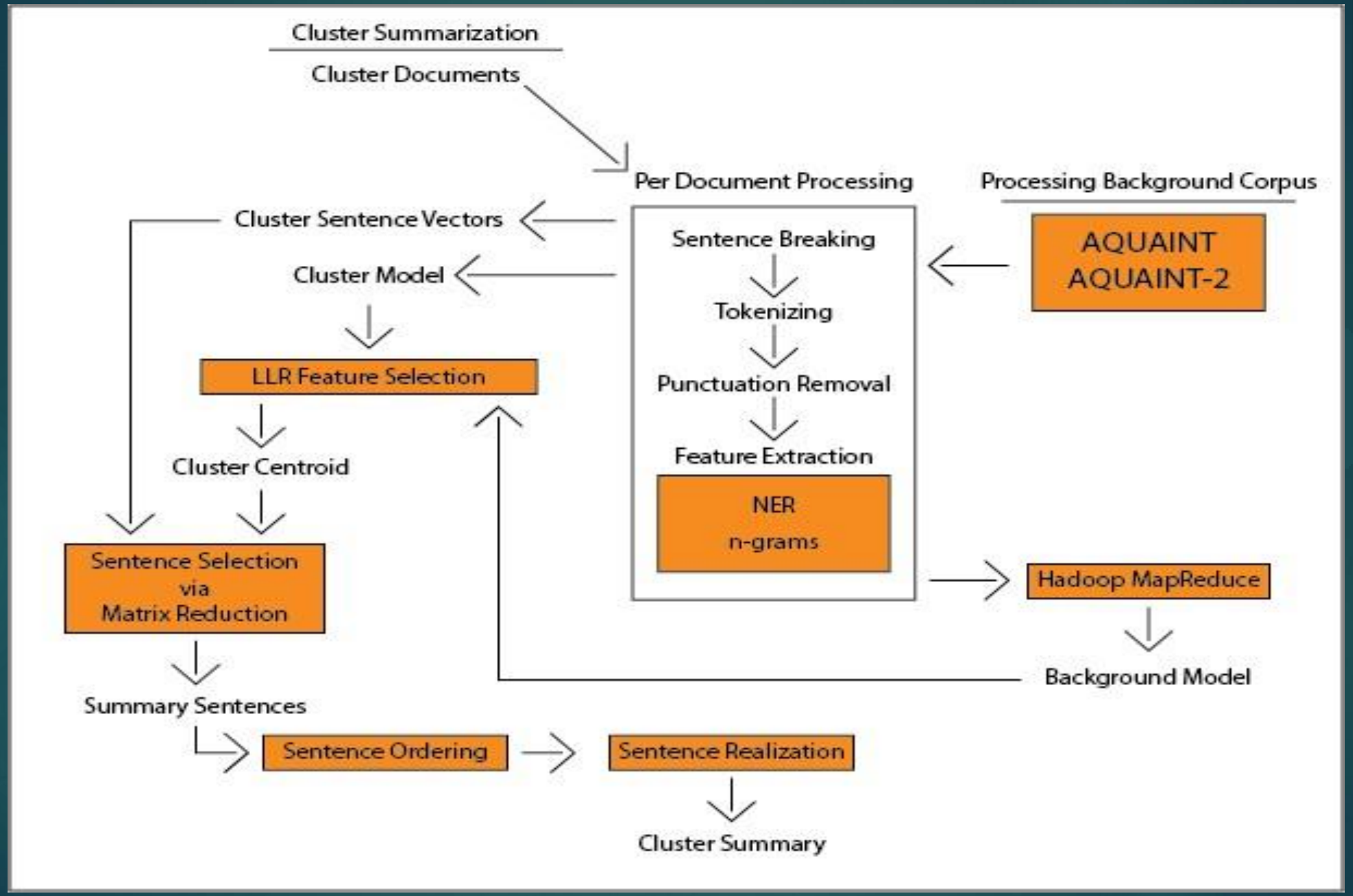
- System Overview
- Approach
  - Preprocessing
  - Centroid Creation
  - Sentence Extraction
  - Sentence Ordering
  - Realization
- Current Results

Overview: Influences

- MEAD (Radev et. al., 2000)
  - Centroid based model
  - Some scoring measures for use in extracted summaries
- CLASSY (Conroy et. al., 2004)
  - Log Likelihood Ratio to detect features in the cluster when compared with the background corpus.
  - Matrix Reduction

Overview: Corpus

- Model:
  - AQUAINT and AQUAINT2
- Document Clusters:
  - AQUAINT and AQUAINT2
- The clusters of documents to be classified are generally 6-10 articles, while the two corpus' are around 2 million articles.
- Because of this, we believe that pulling our model and articles to summarize from the same corpus's will not negatively affect the results.


Cluster Summarization
Cluster Documents
Per Document Processing
Processing Background Corpus
Cluster Sentence Vectors
Cluster Model
Sentence Breaking
Tokenizing
Punctuation Removal
Feature Extraction
NER
n-grams
AQUAINT
AQUAINT-2
LLR Feature Selection
Cluster Centroid
Sentence Selection
via
Matrix Reduction
Hadoop MapReduce
Background Model
Summary Sentences
Sentence Ordering
Sentence Realization
Cluster Summary

Approach: Model Creation

- Background processing for LLR calculation
- Sentence breaking
- Feature vectors
  - Unigrams, trigrams, and named entities.
  - Punctutation removal, stopword removal, and lowercasing were done for the creation of n-gram features.
- NLTK was used for sentence breaking, tokenization, NER and stopword removal.
- The NLTK NE Chunker does a poor job of categorizing the types of names, so we kept it in binary mode.
- Feature types are kept separate to maintain the probability space.
  - Each are kept as their own model, enabling us to load any combination of features we want into the summarizer.

Approach: Centroid Creation

- Similar preprocessing: sentence breaking and vectorization
- Feature counts are stored to compute LLR and then binarized.
- Calculate LLR of all features of a given type.
  - Any feature above a threshold (10.0 for us) is weighted as 1, and any feature below is weighted as 0.
  - Allows retention of features on a per type basis.
  - More favorable approach than simply Top N features from all type by LLR value.
  - A variable number of active features could capture differences in in topic signature that may not be captured when every cluster centroid is kept to an arbitrary number of non-zero weighted features.

Approach: Sentence Extraction

- Three main goals:
  - Cosine similarity between sentence and centroid
  - The position of the sentence within the document it occurs in
    - Organized in decreasing value from 1 to 0, with the first sentence having one and the last having 0.
  - Overlap between the score of the first sentence and the current sentence
    - Dot product of the sentence vectors
    - If there's a headline it is treated as the first sentence
- Each subscore above is weighted and added together for the total score.
- Makes the first sentence the highest scoring sentence

Approach: Sentence Extraction

- Matrix reduction model similar to CLASSY
- For new sentences, features that have already been present in previous sentences are not factored into the score.
  - Avoids redundancy
  - The score recalculated and the new top scoring sentence is added to the summary.

Approach: Sentence Ordering and Realization

- The current sentence ordering is nothing more than the order of appearance.
  - Document IDs are sorted by date when they first read.
- Realization simply prints the sentences as they were retrieved.

Results

- Trigrams yielded the best results on 2010 data across all ROUGE measures.
  - NER was second, followed by unigrams.
  - Combining feature sets did not improve results.
- Unigrams did better on 2009 data
  - May hint at the difference between the two being negligible, depending more on the content being summarized than anything else.

Results: ROUGE, 2010 data

	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Trigrams	**.23115**	**.06297**	**.02200**	**.00900**
Unigrams	.21506	.05213	.01481	.00481
Named Entities	.22417	.05498	.01585	.00453
Trigrams+Unigrams	.21547	.05354	.01578	.00543
Trigrams+Named Entities	.22972	.06087	.02064	.00727
Unigrams+Named Entities	.21655	.05244	.01508	.00491
All	.21725	.05270	.01607	.00527

Results: ROUGE, 2009 data

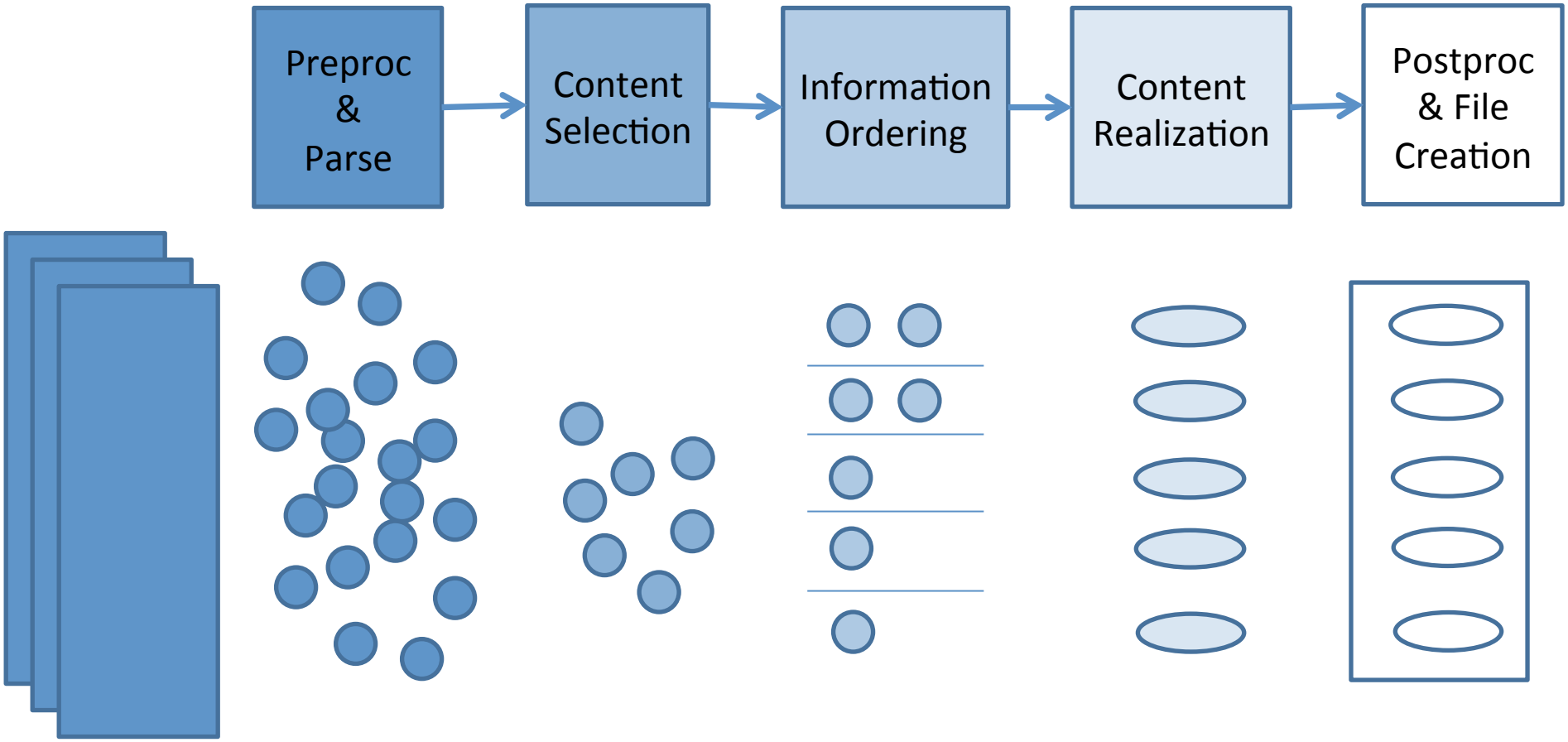
	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Trigrams	.25916	.07706	.02943	.01308
Unigrams	**.28889**	**.08884**	**.03354**	**.01512**

Discussion

- Major source of error could be sparseness of the feature vectors being compared to the LLR generated centroid.
  - Caused by the large number of features and matrix reduction
  - The latter could remove most or all the features from a particular vector, resulting in sentences with no similarity to the centroid.
  - The system will choose many lead sentences, replicating the LEAD baseline algorithm.
  - Could avoid this by using a more MMR based system to avoid redundancy
  - Could add more features to the content selection, downweighting sentence position relative to other factors.
  - Change the weights entirely.
  - Change the overlap score to represent the overlap with given topic name, rather than the headline.

Text Summarization

Syed Sameer Arshad
Tristan Chong



Preprocessing and Parsing

- XML formatting was an issue
- Both corpora had different arrangements for the data.
- It was challenging to scrub the data in order to parse it.
- Needed to make two parsers in Python.

Content Selection

- We used MEAD
- We set count-IDF threshold for entry into the centroid to be 5.
  - This was an arbitrary choice.
- We used the Radev et al. 2000 paper to set up other constants, such as the minimum number of words in a summarized sentence.
- Final Score = c * centroid-score + p * position-score + f * first-sentence-similarity-score
- Centroid score is the sum of count-IDF for each term found in a sentence.
- First-sentence-similarity –score is the dot-product between a sentence and the first-sentence designated for its article.
  - The first sentence of an article is its headline. If the headline is smaller than 9 words, it is the first sentence in the article that is at least 15 words long.
- Position-score is set to 1 for the first sentence in an article and then drops fractionally towards 0 for every following sentence.
- All three scores were normalized with min-max normalization.
- C = 3, p = 2, f = 1

Information Ordering

- After we sort all sentences related to a topic in descending order of final-score, we pick the minimum number of top highest scoring sentences that help us reach the word limit.
- Then divide them into five cohorts.
- Each cohort represents sentences found in particular quintiles of their source documents.
- We follow this principle:
  - If a selected sentence was located in the n'th 20% of a document, it should end up in the n'th 20% of the summary, where n ranges from 1 to 5.

Content Realization

- We just copied the source sentence into the summary as is.

Post-processing and File Creation

- We pasted every sentence on a new line, as requested.

Results

	ROUGE1	ROUGE2	ROUGE3	ROUGE4
Precision	0.222	0.04705	0.01453	0.00352
Recall	0.19886	0.04261	0.01317	0.00316
F-Score	0.20922	0.04461	0.01378	0.00332