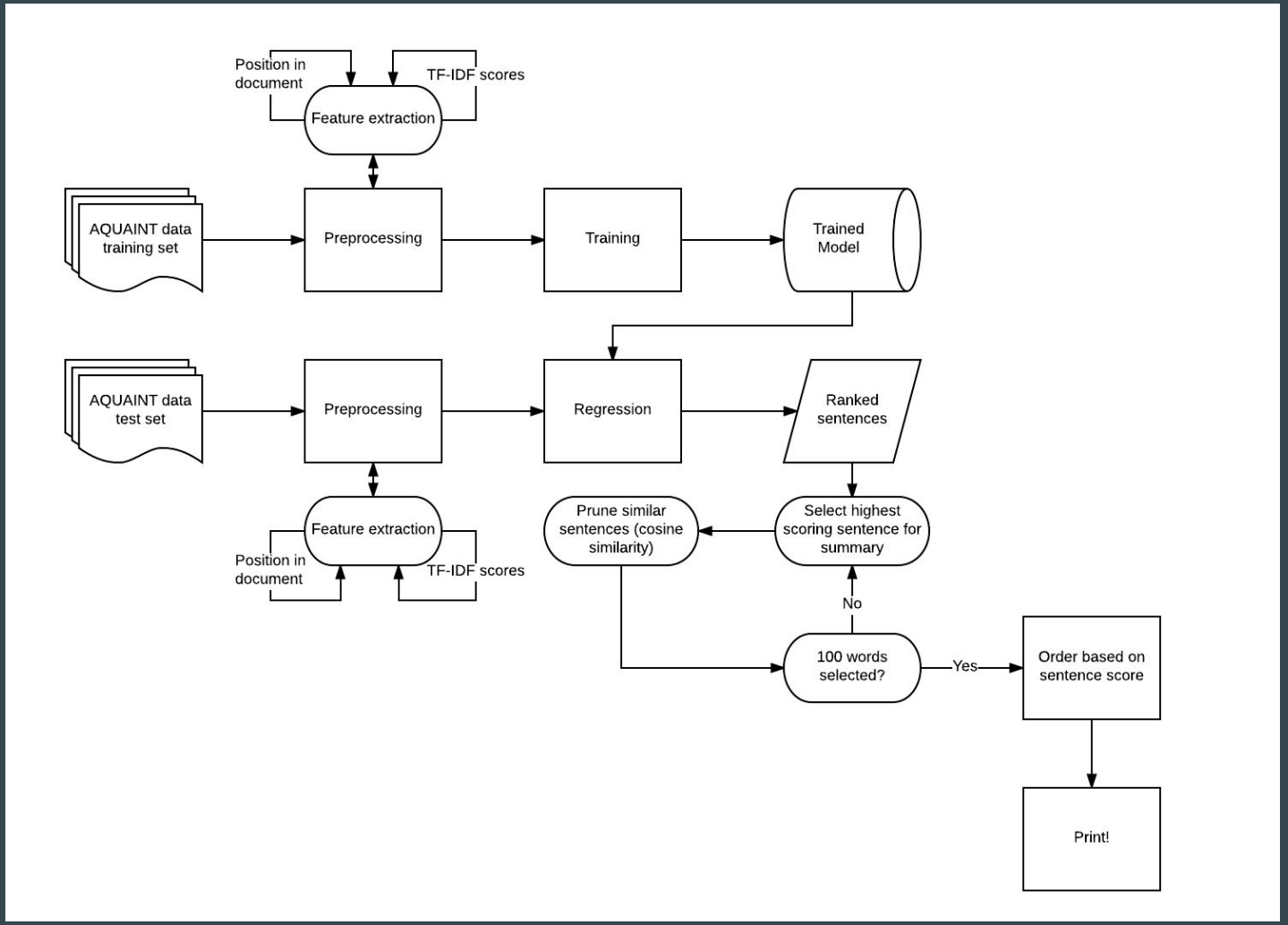
Deliverable #2

Alex Spivey, Eli Miller, Mike Haeger, and Melina Koukoutchos
April 25, 2017

System Architecture



Content Selection

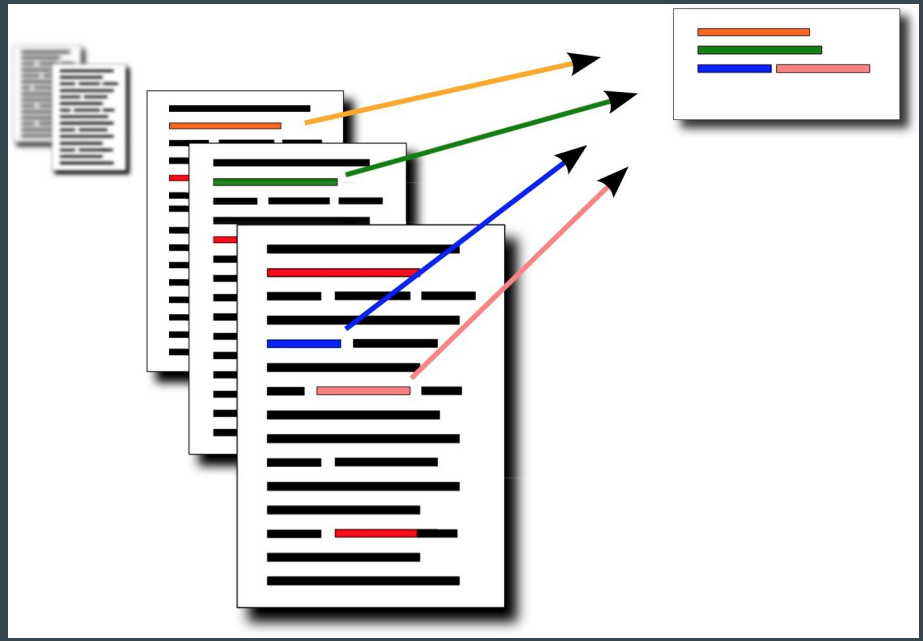
Preprocessing

Feature Extraction

The Model

Logistic Regression

Sentence Selection



Feature Extraction/TF-IDF

TF-IDF scores are collected for each word in the sentence, with the document frequency taken from each document in the dataset

The average TF-IDF score for each sentence is computed and used as a feature in the logistic regression model

The Model

Logistic regression model

Current features: position, TF-IDF

Labels: 1=in summary, 0=not in summary

We don’t do straight classification

We use probabilities calculated by model as scores

Sentence Selection

Select highest scoring sentence

Calculate cosine similarity

Prune sentences

Repeat until there is enough summary content

Information Ordering & Content Realization

Still to come!

For now:

Sentences are ordered by scores

Content is printed until adding another sentence would exceed 100 words

Results

ROUGE Recall

ROUGE-1	0.18765
ROUGE-2	0.0434
ROUGE-3	0.01280
ROUGE-4	0.00416

A Sample Summary

NEW YORK _ With the indictments barely unsealed against fourpolice officers in the Amadou Diallo shooting, a battle is alreadytaking shape over physical evidence in the case, as lawyers andexperts seek to buttress their own versions of what happened basedon entrance wounds, bullet trajectories and other forensic details.

Issues & Successes

- Preprocessing
  - Confusing data directories
  - Some difficult to work with file formats
- Gold standard data
  - Gold standard summaries are generative, not extractive
  - Treated gold standard summaries as another document for this milestone
  - Might also try cosine similarity to gold standard, or another option, when there is a more complete system to tune.
- It does actually run to completion

Resources

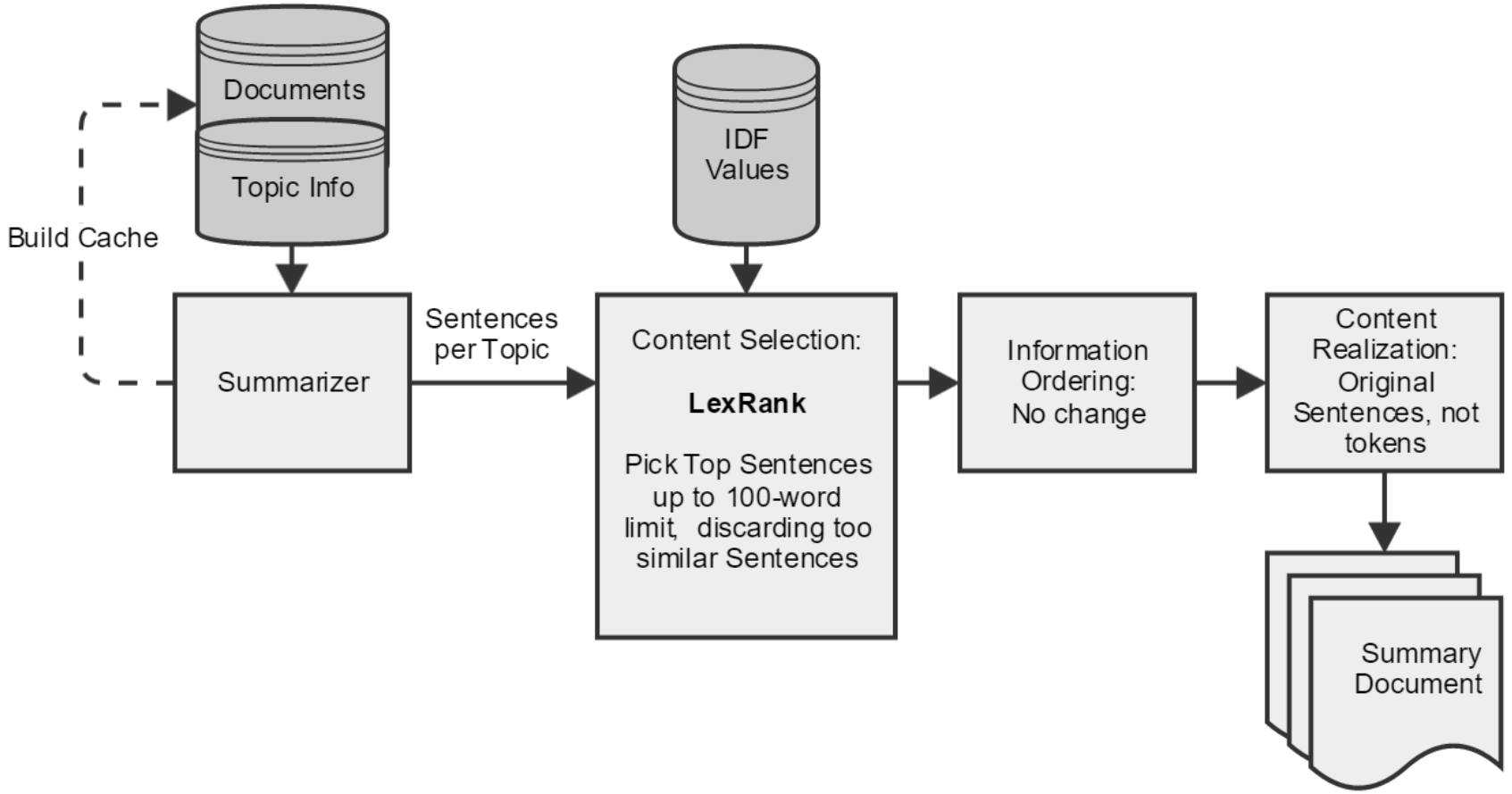
M. Wang, X. Wang, C. Li and Z. Zhang. 2008. *Multi-document Summarization Based on Word Feature Mining*. 2008 International Conference on Computer Science and Software Engineering, 1: 743-746.

You Ouyang, Wenjie Lia, Sujian Lib, and Qin Lu. 2011. *Applying regression models to query-focused multi-document summarization*. Information Processing Management, 47(2): 227-237.

Multi-Document Summarization

DELIVERABLE 2: BASE END-TO-END SUMMARIZATION SYSTEM

TARA CLARK, KATHLEEN PREDDY, AND KRISTA WATKINS

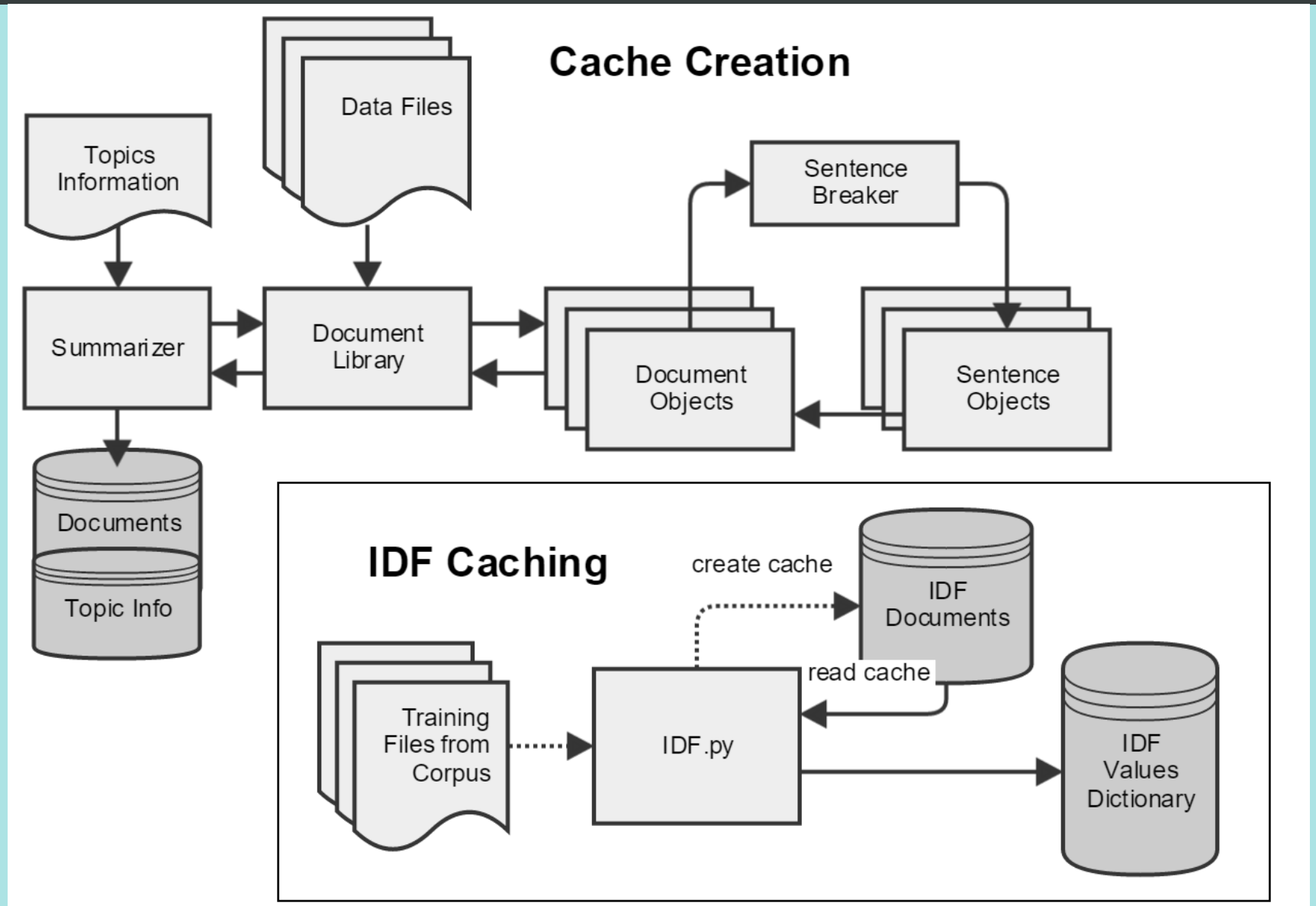


System Architecture

Our system is a collection of independent Python modules, linked together by the *Summarizer* module.

Text Processing

- Read in the Topics file as a tree
- Use the Topics document id’s to read in Document objects
- Store documents in a DocumentLibrary
- Sentence breaking: breakSent Perl script
- Tokenization:
  - NLTK
  - NLTK, stemmed and downcased



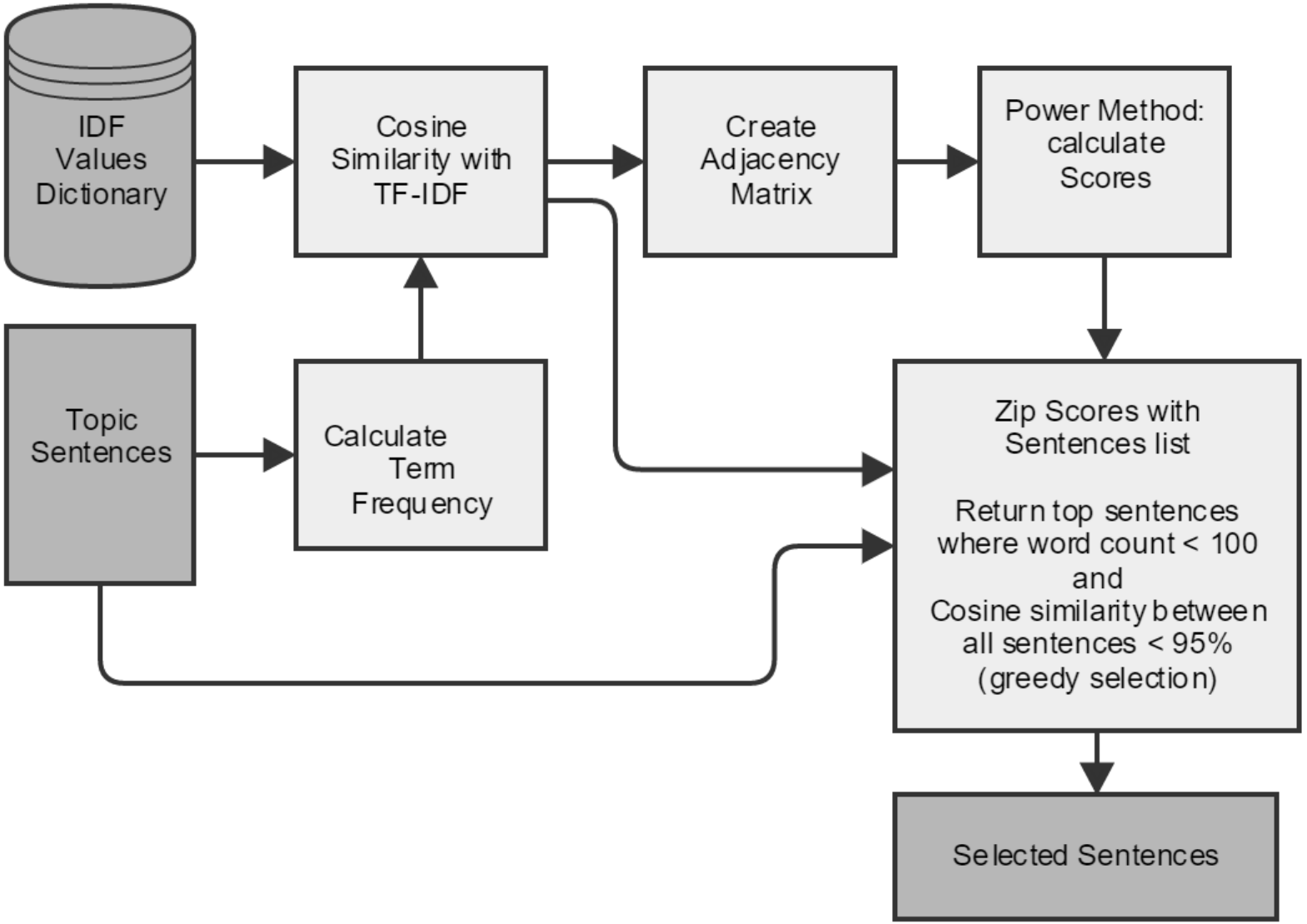
Caching

The *Summarizer* module runs caching for training or devtest documents if no cache is found.

IDF Document caching is manual. IDF Values caching can be run separately, following document caching

Caching

- Use Pickle for caching:
  - Topic Library for training/devtest data sets
  - Document Library for training/devtest data sets
  - IDF Document Library corpus
  - IDF score dictionary
- Caches each item as it is processed



Content Selection

We use the LexRank algorithm, followed by filters for summary length and sentence similarity.

Content Selection

- Input: Documents in a Topic
- Algorithm: LexRank
- Output: List of best sentences, ordered by rank

LexRank: A Graphical Approach

- Nodes are sentences; edges are similarity scores
- Nodes: TF-IDF vector over each stem in the sentence

$$tf_t = \frac{number\ of\ times\ term\ t\ appears\ in\ doc}{total\ terms\ in\ doc}$$

$$idf_t = \log\left(\frac{total\ number\ of\ docs}{number\ of\ docs\ containing\ term\ t}\right)$$

Note – Unknown terms receive an IDF score of $\log(D)$

- Edges: Cosine similarity between sentences X and Y

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

Prune edges below 0.1 threshold

Power Method

- Set normalized vector p
- Update p → dot product of transposed graph and current p
- Apply until convergence
- Apply scores from p vector to the original Sentence objects
- Return the best sentences, without going over 100 words or repeating yourself (cosine similarity < 0.95)

Information Ordering

- Input: List of sentences from content selection
- Output: Copy of this list

Content Realization

- Input: List of sentences from Information Ordering
- Output: Write each sentence on a new line to the output file

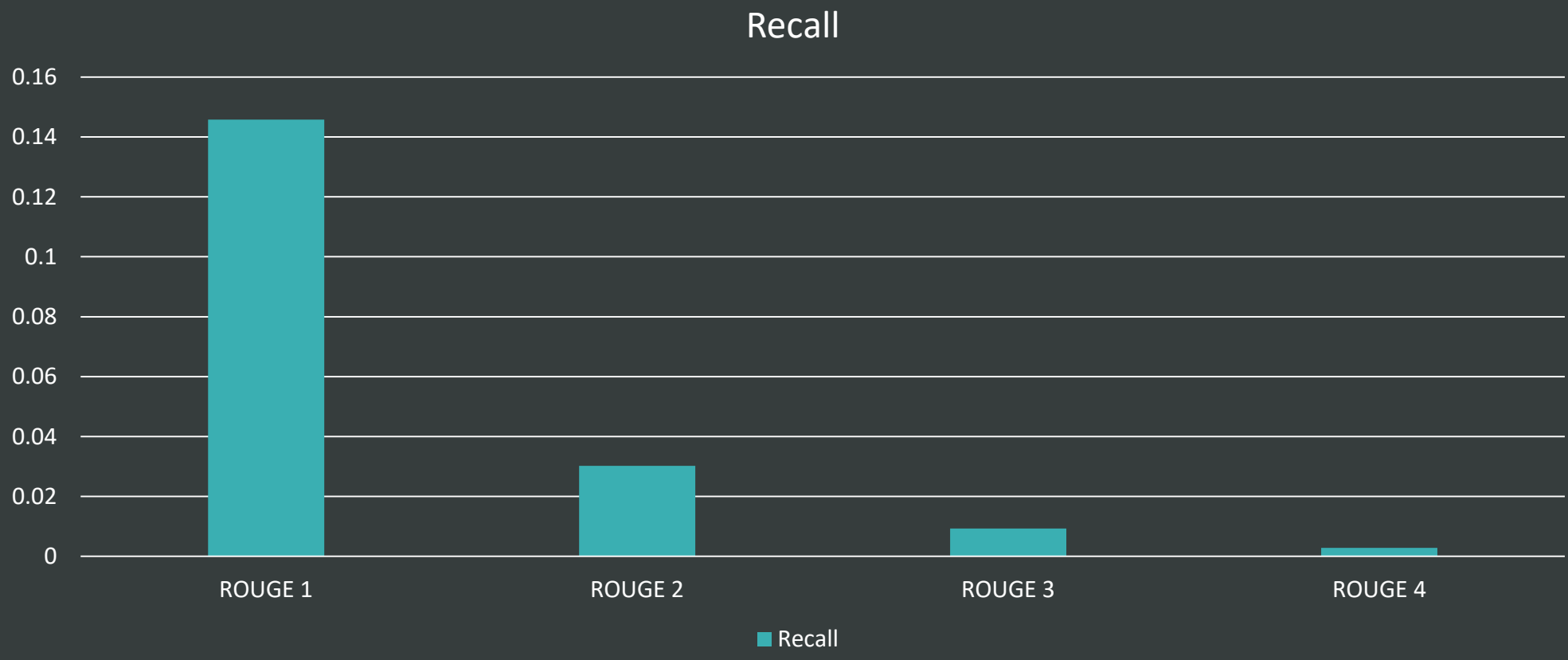
Issues and Successes

- File reading
  - Choosing the appropriate files to improve performance
  - Switching from xml.etree.ElementTree.parse() to BeautifulSoup
- Sentence breaking
  - Performance of breakSent with wrapper
  - Adding abbreviations to the breakSent abbreviation dictionary
  - Need to handle decimal breaks
  - Need to make improvements in breaking sentences with quotations

Issues and Successes

- Content Selection
  - Long sentences
    - Average summary length of 2.087 sentences
    - Next steps: Check the content selection algorithm to ascertain that it is not favoring long sentences.
  - Similarity threshold value
    - Still too many similar sentences
    - Next steps: Lower the similarity threshold value
  - Punctuation
    - Single punctuation is stripped, but double punctuation like '' and `` are treated as tokens
    - Possibly overweighting the value of quotations for some topics
    - Next steps: Remove double punctuation
  - Short summaries (1 sentence long)
    - 26% of summaries
    - Next steps:
      - Check that the adjacency matrix is keeping good paths.
      - Check the sentence lengths in the documents.
      - Check that we're reading in all the sentences appropriately.

Results


Recall
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
ROUGE 1
ROUGE 2
ROUGE 3
ROUGE 4
Recall

Related Reading

- Günes Erkan , Dragomir R. Radev, LexRank: graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research, v.22 n.1, p.457-479, July 2004
- Ani Nenkova , Rebecca Passonneau , Kathleen McKeown, The Pyramid Method: Incorporating human content selection variation in summarization evaluation, ACM Transactions on Speech and Language Processing (TSLP), v.4 n.2, p.4-es, May 2007 [doi>10.1145/1233912.1233913]
- Karen Spärck Jones, Automatic summarising: The state of the art, Information Processing and Management: an International Journal, v.43 n.6, p.1449-1481, November, 2007 [doi>10.1016/j.ipm.2007.03.009]

Questions?

Multi-document Summarization

Ling 573 group project by Joanna Church, Anna Gale, Ryan Martin

Presented by Joanna
April 2017

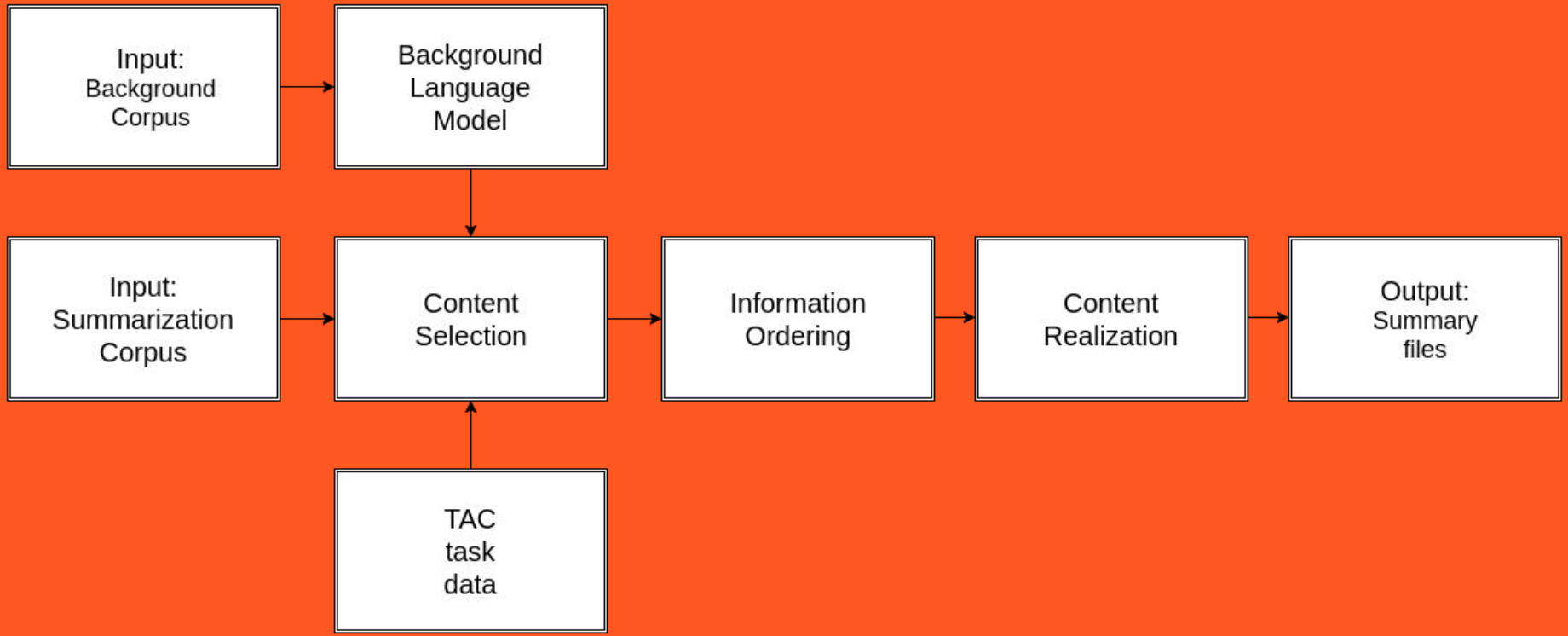
Resources

Our Inspiration

John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2006. Topic-focused multi-document summarization using approximate oracle score. In *Proceedings of the COLING-ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 152-159, Stroudsburg, PA, USA. Association for Computational Linguistics.

- Extractive content selection using oracle score
- "Our 'oracle score' will then compute the expected number of summary terms a sentence contains, where the expectation is taken from the space of all human summaries on the topic τ."
- Human variation is modeled using a unigram bag-of-words

System Architecture


Input:
Background
Corpus
Background
Language
Model
Input:
Summarization
Corpus
Content
Selection
Information
Ordering
Content
Realization
Output:
Summary
files
TAC
task
data

System Design

Preprocessing:

- Process AQUAINT input: model topics into DocumentSet, Document, Sentence objects
- Sentence object contains tokens and surface form
- Background Corpora: 25 files from New York Times Gigaword Corpus
  - ~160 million total tokens and ~650 unique tokens
- Generate unigram language model using background Corpora

Main components:

- Content Selection
- Information Ordering
- Content Realization

Content Selection

Content Selection Method

- Query terms: collect potential query terms from title and narrative strings
  - Stanford CoreNLP POS tagger
  - Only keep NN, VB, JJ and RB
- Signature terms: collect signature terms from each sentence using log likelihood ratio (LLR)
- Select sentences
  - Must be longer than 8 tokens
  - Weight is distributed evenly over query words and signature words (0.5)

```
for each DocumentSet:
  #Find query terms
  potentialQueryTerms := PosTagger(title) +
PosTagger(narrative)
  for term in potentialQueryTerms:
    if term is in [NN, VB, JJ, RB]:
      add to queryTerms

  #Find signature terms
  for each Sentence:
    for each token:
      calculate log-likelihood
      if log-likelihood > threshold
        add to signatureTerms

  #Pick sentences
  for each Sentence:
    if sentlen >= 8:
      for each token:
        if token is in queryTerms:
          score += 1/2
        if token is in signatureTerms:
          score += 1/2
      score /= numTokens
      add (sent, score) to priorityQ
  while (summaryLen < 100):
    add priorityQ.pop() to summary
```

Formulas

- Term probabilities:
  - qt is an indicator function for query terms (1 or 0)
  - k = 0.5
  - st is an indicator function for the presence of the signature term

$$P(t|\tau) = kq_t(\tau) + (1-k)s_t(\tau)$$

$$k \in [0,1]$$

- Sentence score:
  - |x| is the number of distinct terms
  - T is the set of all terms
  - x(t) is the indicator function for the term in the sentence (1 or 0)

$$w(x) = \frac{1}{|x|} \sum_{t \in T} x(t) P(t|\tau)$$

Information Ordering

Information Ordering Strategy

- Relevance order: the sentences appear according to their score, with the highest scoring sentences first

More to come....

Content Realization

Content Realization

- Line by line

More to come....

Issues and Successes

Ideas to Explore

- ROUGE results

Task	R-1	R-2	R-3	R-4
devtest	0.15755	0.02176	0.00477	0.00180

- Optimize parameters used in the formula (currently 0.5 for both query and signature term).
- Remove redundancy
- Capture that article HEADLINEs that contain high-value words.
- Improve shallow parsing
    - Ex. “avalanche” and “avalanches”

Thanks for listening!

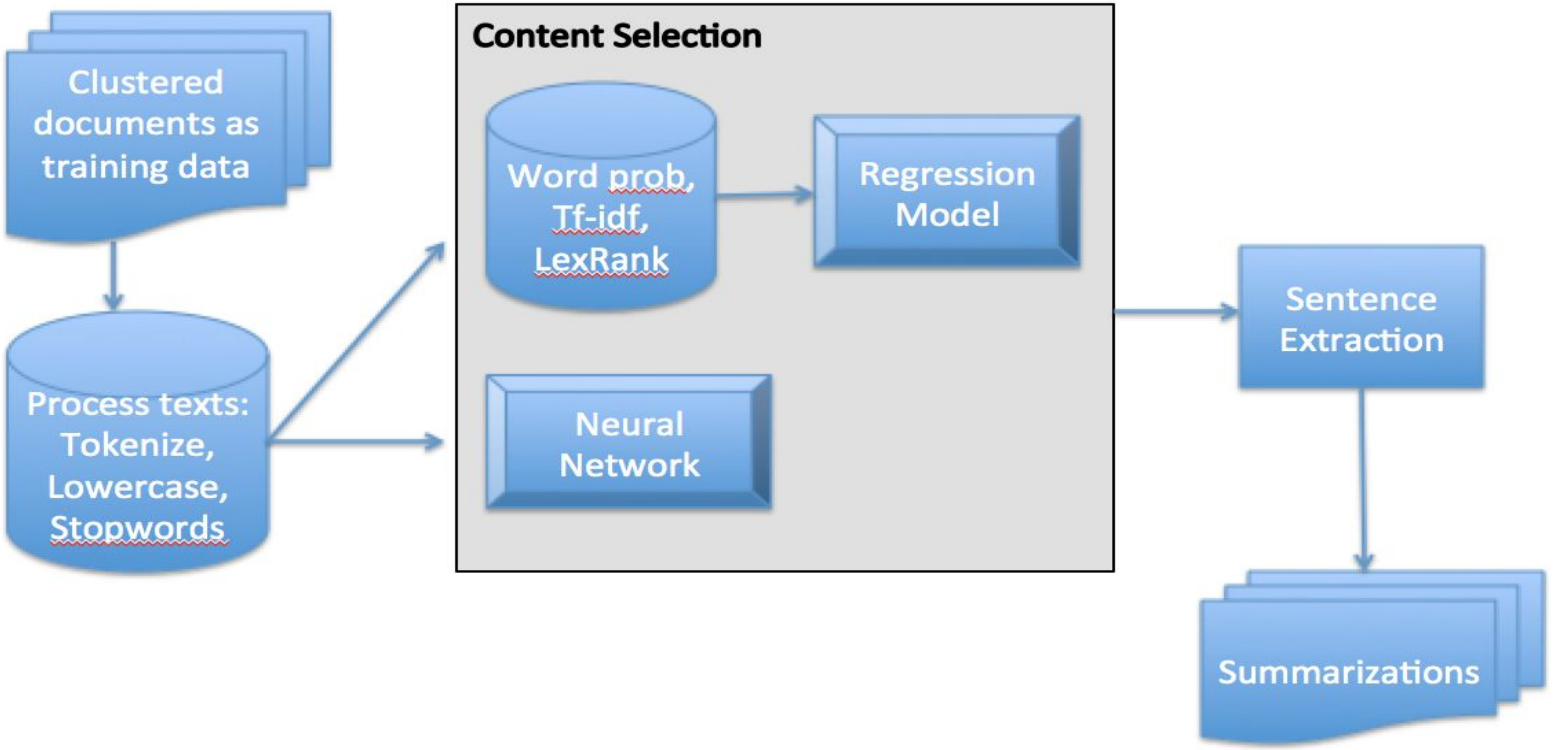
Ling 573 Project
Automatic Summarization System

Wenxi Lu, Yi Zhu, Meijing Tian

Outline

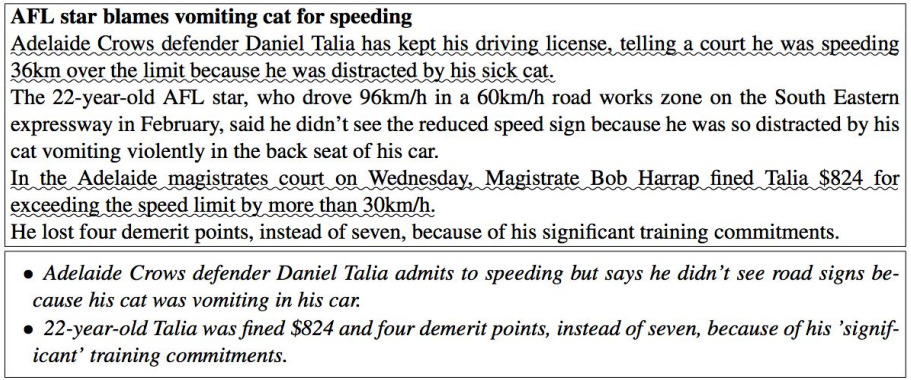
- System Architecture
- Data
- Preprocessing
- Content Selection
- Neural Network
- Issues and Discussion
- References

System Architecture



Data

- Training:
  - CNN/DailyMail articles for sente
    single-document summarizations
    - CNN: 83,568 articles
    - DailyMail: 193,981 articles
- Test:
  - 2010 TAC shared task dataset (46
    topic-oriented document)
  - 500 samples from DailyMail testset (10346)



AFL star blames vomiting cat for speeding

Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat.

The 22-year-old AFL star, who drove 96km/h in a 60km/h road works zone on the South Eastern expressway in February, said he didn't see the reduced speed sign because he was so distracted by his cat vomiting violently in the back seat of his car.

In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h.

He lost four demerit points, instead of seven, because of his significant training commitments.

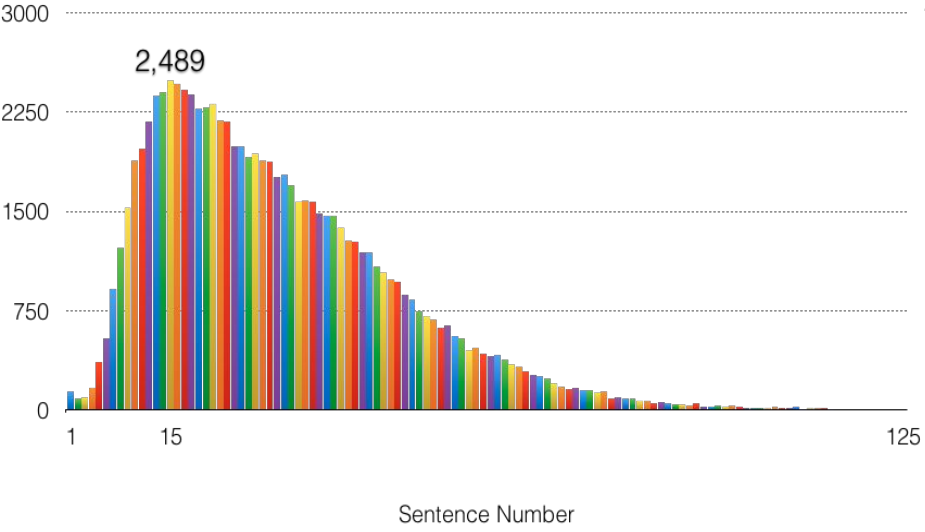
- *Adelaide Crows defender Daniel Talia admits to speeding but says he didn't see road signs because his cat was vomiting in his car.*
- *22-year-old Talia was fined $824 and four demerit points, instead of seven, because of his 'significant' training commitments.*

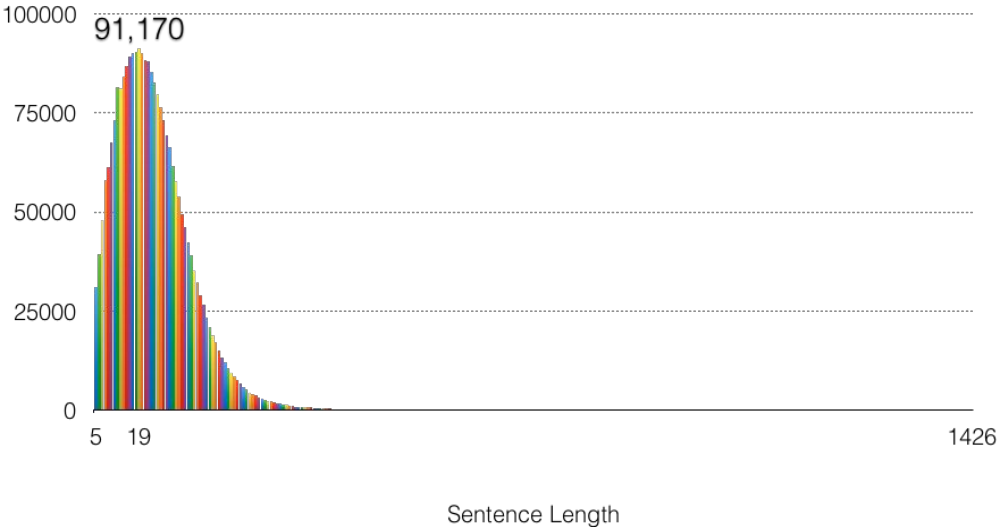
Preprocessing

- Tokenization
- Lowercase
- Vocabulary List
  - The union of the CNN, DailyMail words (~200k)
  - Sort according to word probability
  - Set a threshold, extracting top ~20k words

Preprocessing

- DailyMail Statistics





Content Selection

- Inspired by Hong & Nenkova (2014)
- Goal: select most salient sentences from articles in given document for summarization
- Sentence level extraction instead of word level extraction

Content Selection

Word Probability

$$p(w) = \frac{c(w)}{N}$$

c(w): number of times word w occurs in the given document

N: total number of word tokens in the document

Content Selection

Tf-idf

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|d \in D : t \in d|}$$

tf(t,d): raw count of word t appears in article d
N: total number of articles in the document
$|\{d \in D : t \in d\}|$:number of articles in the document where the term t appears

Content Selection

LexRank

$$idf - modified - cosine(x, y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}}$$

x, y : sentences in article

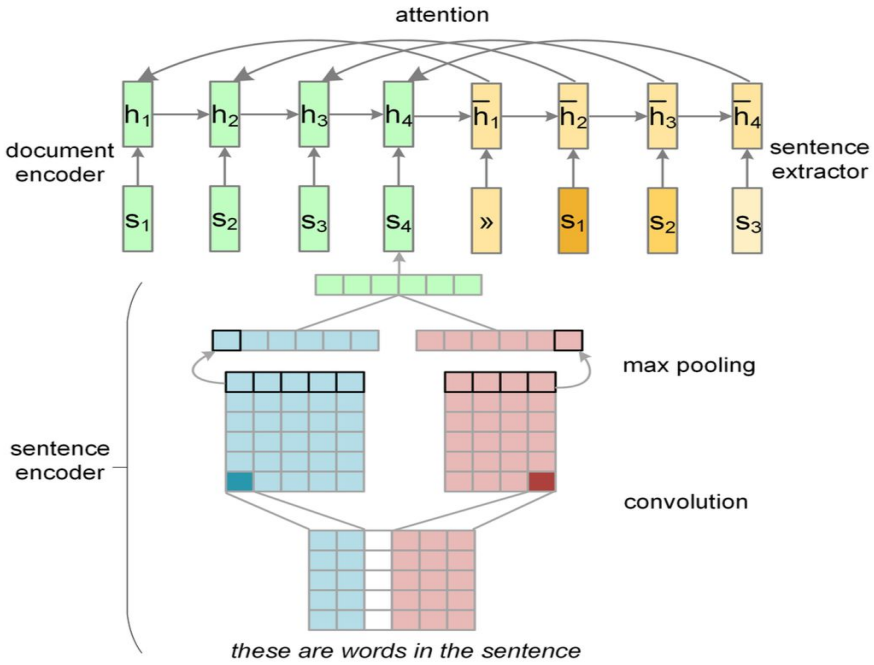
w: word in sentence

Content Selection

- Input:
  - A single document with sentences (from several articles)
- Output:
  - Labels of each sentence in the document
  - Label 1 or 0
  - Add sentences with label 1 to summary

Neural Network

- Input
  - A single document with sentences
- Output
  - Labels of each sentence in the document
- Loss Function
  - Cross entropy (sigmoidal loss)
- Summary
  - Single-document summarization
    - all sentences with label 1
  - Multi-document summarization
    - the most plausible sentence for each doc
    - merge to a single summary in **document chronological order**



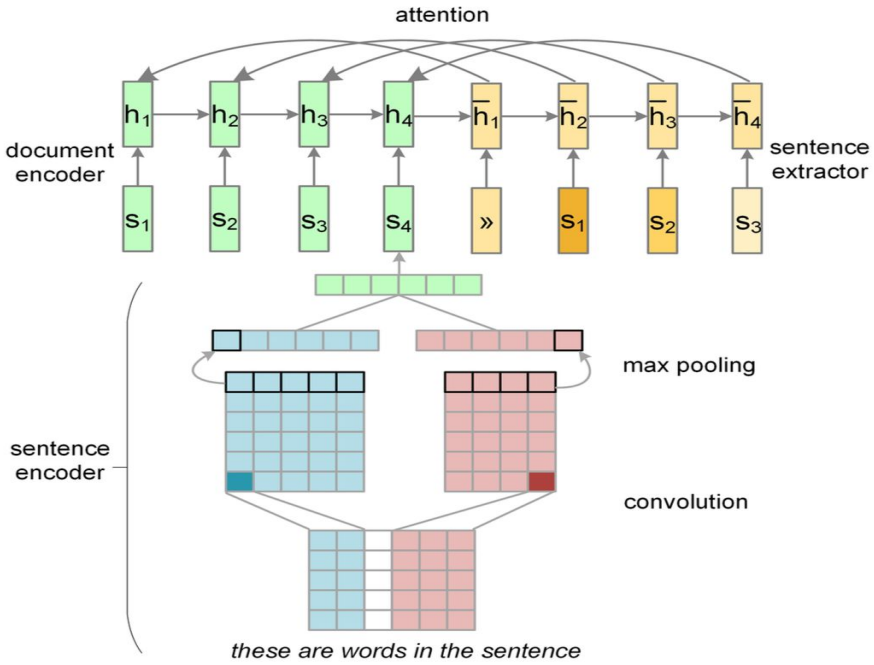
Neural Summarization by Extracting Sentences and Words [Cheng et al; 2016]

Neural Network

- Hyperparameters
  - Pretrained embeddings: Glove
  - Word embedding dim: 100
  - CNN filter size: 1 ~ 7
  - Filter nums (sentence dim): 300
  - RNN hidden dim: 750
  - Max sentence length: 40
  - Max sentence number: 150
  - Batch size: 100
  - Epoch: 10 (never reached :))
- Optimizer
  - Adam (default for tensorflow)

$$\bar{\mathbf{h}}_t = \mathrm{LSTM}(p_{t-1}\mathbf{s}_{t-1}, \bar{\mathbf{h}}_{t-1})$$

$$p(y_L(t) = 1|D) = \sigma(\mathrm{MLP}(\bar{\mathbf{h}}_t : \mathbf{h}_t))$$



Neural Summarization by Extracting Sentences and Words [Cheng et al; 2016]

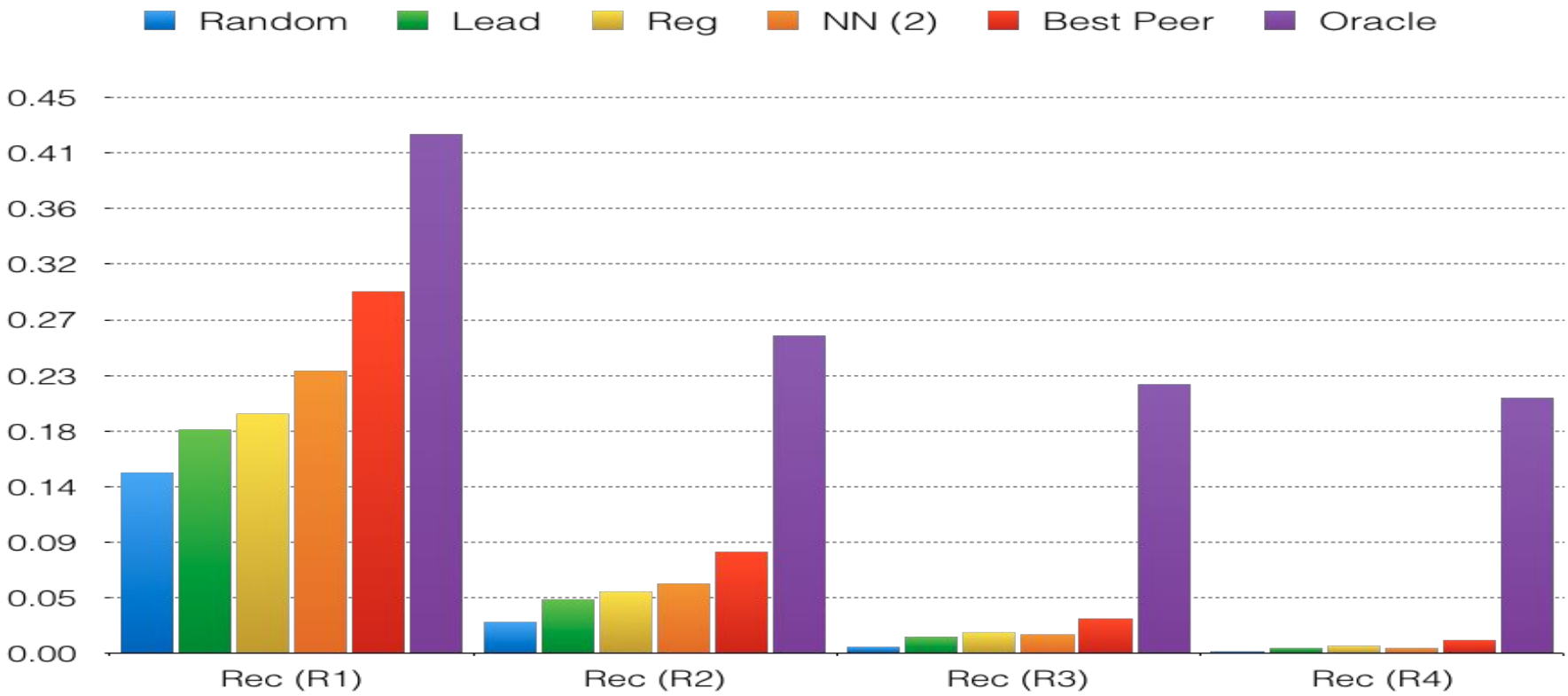
Experiments and Results

- Regression Model
  - Train models on CNN training set (3000 files)
  - Test on TAC 10 test set
- Neural Network Model
  - Train models on CNN and DailyMail training set
  - Test both models on TAC10 test set
  - Test DailyMail model on the DailyMail test set

Experiments and Results (TAC10 test data)

	Rec (R1)	Rec (R2)	Rec (R3)	Rec (R4)
Random	0.14647	0.0256	0.00554	0.00142
Lead	0.18094	0.0438	0.01302	0.00395
Reg	0.19351	0.0501	0.0167	0.0057
NN (2)	0.22868	0.05655	0.0154	0.00394
Best Peer	0.29261	0.08206	0.0278	0.01069
Oracle	0.42004	0.25752	0.21786	0.20666

Experiments and Results (TAC10 test data)



Experiments and Results (DailyMail)

Model	Rouge-1	Rouge-2
Cheng et al.	0.212	0.083
NN (acc = ~80%)	**0.531**	**0.22**

Issues & Discussion

- Regression Model
  - Training accuracy: Not enough features (3 so far)
  - Need more training
- Neural Network Model
  - Just wait :)
    - If seems working, just still wait …
    - If not working, kill it and check the code
  - Discrepancy between loss function and the final target
  - Domain adaptation problem (CNN & DailyMail to TAC)
  - Difference between single-document summarization and multi-document summarization
    - Strategies to merge summaries
  - Preprocessing could be crucial
    - Sentence splitting
  - TAC training data available but not used

Issues & Discussion

- Use more data as training data (add label for each sentence)
- Improve data preprocessing
- Use word-level extraction instead of sentence level extraction
- Information ordering and content realization
- Alternative document merge strategies
  - Fixed length of summary for each document
- Change network architecture for multi-document summarizations
  - Might be appropriate to model ROUGE directly or via reinforcement learning
  - Shifting to **Abstractive** summarization (pure seq2seq)
    - Sample the rest of the words and use rouge as rewards

References

- Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi- document summarization. In EACL. pages 712–721.
- Gu¨nes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research 22:457–479.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems. pages 3104–3112.
- Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Berlin, Germany, pages 484–494. http://www.aclweb.org/anthology/P16-1046.

Reference Scores

- TAC 2010 guided summarization task:
  - ROUGE-2:
    - LEAD baseline: 0.05376
      - First 100 words of latest articles
    - MEAD baseline: 0.05927
      - Default MEAD settings
    - Best official: 0.09574