# OutbreakSum

Automatic Summarization of Texts Relating to Disease Outbreaks

CS 4984: Computational Linguistics
12/10/2014

Richard Gruss, Daniel Morgado,
Nate Craun, Colin Shea-Blymyer

## Collection Processing

- Your Small
  - Encephalitis
  - 365 files
- Your Big
  - Ebola
  - 15,000 files
- News Articles

## Collection Processing

- Problems
  - Non-English Languages
  - Blank Files
  - Irrelevant Files
  - Duplicate Files
- Solutions
  - Hashing
  - Classification

## Collection Processing

- Classification
  - Features
    - tfidf
    - Document Length
    - Target Words
- Future Improvements
  - “Web” Stop Words
  - HTML Parsing

## Hadoop: Performance

- Small file problem
  - Excessive calls to Map function
  - Reading 64mb blocks containing few kb of data.
  - Built *corpus_squeeze.py* utility
    - Classifying 7k documents: 13m27s → 1m18s
- Passing data representations
  - JSON
  - pickle: ASCII serialized Python objects

## Hadoop: Workflow

- Streaming interface does not facilitate code reuse.
  - Framework for reusable mapper and reducer
  - Shared utilities module
  - All jobs define same 3 functions:
    - Compose → Combine → Compute
- Debugging
  - MapReduce simulation script

## Language Processing

1. Find words with highest tf-idf
2. Create “your words” list of intuitively identifying words
3. Combination of these lists used for classification

| | | |
|---|---|---|
| disease | vaccine | children |
| outbreak | cure | environment |
| epidemic | antibiotic | study |
| infect | hospital | kill |
| quarantine | health | medicine |
| sick | contract | CDC |
| airborne | transmit | WHO |
| waterborne | organism | diagnose |
| pathogen | toxic | |
| virus | sanitation | |
| bacteria | cholera | |
| parasite | case | |
| symptoms | die | |
| treatment | patients | |

## Language Processing

Named Entity Recognition → Best Results

Stanford NER - Very effective only when passed a single tokenized sentence

NLTK NE Chunk - Far less accurate than SNER

Open NLP - Faster and more effective than SNER

Clustering & Topics → Bad Results

## Summarization: Template

[DISEASE] has struck [LOCATION].  As of [DATE], there have been [NUMBER] number of cases either killed or hospitalized.  Authorities are [AUTHORITIES MEASURES]. [DISEASE] spread [HOW IT SPREADS]. [HEALTH ORGANIZATION] is  [WHAT HEALTH ORG IS DOING].  [DISEASE] include [SYMPTOMS].  Treatments include. [TREATMENTS].  The total cost of the outbreak is estimated at  [MONEY AMOUNT].  The disease may spread to [OTHER LOCATIONS].

## Summarization: NLG

**Report using NLG**

*Database*

**{'date': '3069-12-02', 'source': '10208-3', 'cases': 0, 'location': 'Liberia', 'deaths': '1552'}**

**{'date': '2014-12-04', 'source': '10219-4', 'cases': 0, 'location': 'West Africa', 'deaths': '700'}**

**{'date': '2014-12-04', 'source': '10219-4', 'cases': 0, 'location': 'Liberia', 'deaths': '700'}**

**{'date': '2014-12-02', 'source': '1023-5', 'cases': '201', 'location': 'West Africa', 'deaths': '672'}**

## Summarization: NLG

News in the United States of 4000 cases recently reported in West Africa has caused concern.

```
(NP
  (NP (NNP News))
  (PP (IN in)
    (NP
      (NP (DT the) (NNP United) (NNPS States))
      (PP (IN of)
        (NP
          (NP (CD 4000) (NNS cases))
          (PP (IN in)
            (NP (NNP West) (NNP Africa))))))))
(VP (VBZ has)
  (VP (VBN caused)
    (NP (NN concern)))
```

## Summarization: NLG

**summary** ::= intro-sentence detail-paragraph history-paragraph closing-sentence
**intro-sentence** ::= "There has been an outbreak of Ebola in the following locations: " location-list
**detail-paragraph** ::= detail-sentence detail-sentence-with-transition*
**detail-sentence-with-transition** ::= transition detail-sentence
**transition** ::= ("Also" | "In addition" | "Likewise" | "Additionally" | "Furthermore")
**detail-sentence** "In [month year], there were between [num] and [num] cases of Ebola in location, with between [num] and [num] deaths."
**history-paragraph** ::= "There have been earlier cases of Ebola." history-sentence history-sentence-with-transition*
**history-sentence-with-transition** ::= transition + history-sentence
**history-sentence** ::= "Ebola was found in [location] in [year]."
**closing_sentence** ::= "[Ebola boilerplate]"

## Summarization: NLG

There has been an outbreak of Ebola reported in the following locations: Liberia, West Africa, Nigeria, Guinea, and Sierra Leone.

In January 2014, there were between 425 and 3052 cases of Ebola in Liberia, with between 2296 and 2917 deaths. Additionally, In January 2014, there were between 425 and 4500 cases of Ebola in West Africa, with between 2296 and 2917 deaths. Also...

There were previous Ebola outbreaks in the past. Ebola was found in 1989 in Liberia. As well, Ebola was found in 1989 in West Africa. ...

Ebola virus disease (EVD; also Ebola hemorrhagic fever, or EHF), or simply Ebola, is a disease of humans and other primates caused by ebolaviruses...

## Questions?

Special thanks to:

- Edward Fox
- Xuan Zhang
- Tarek Kanan
- Mohammad Halimi

Sponsors:

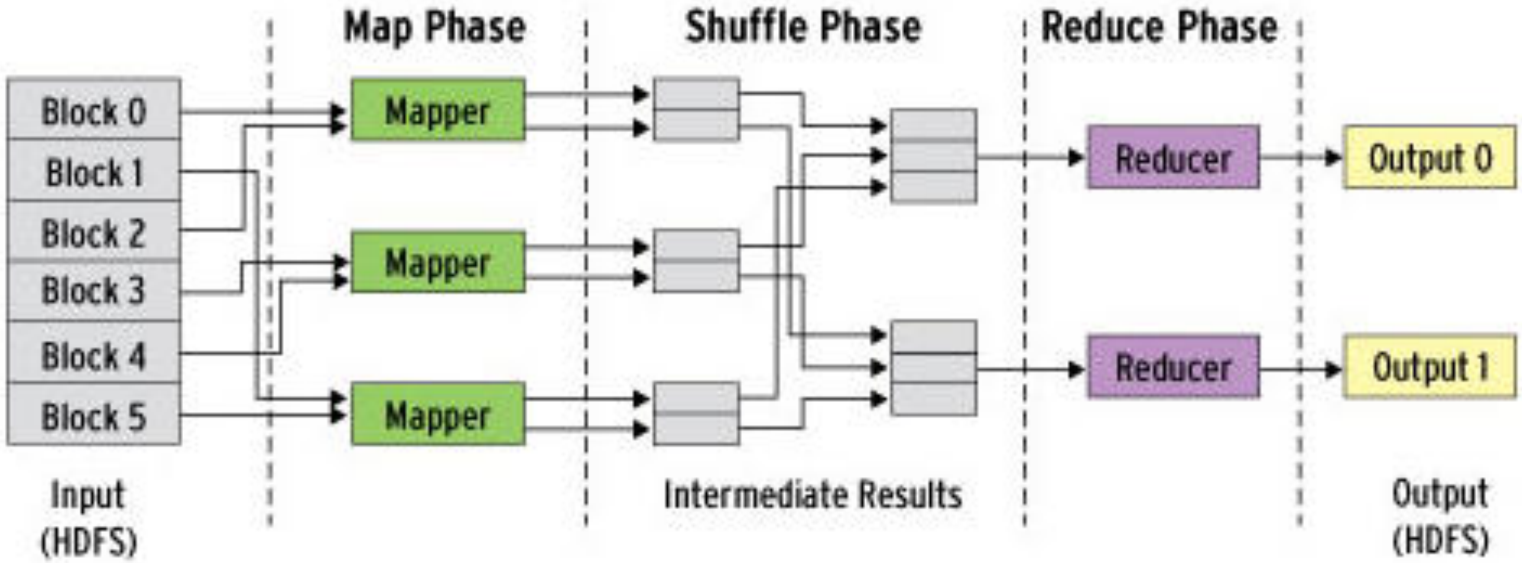
- NSF DUE-1141209
- IIS-1319578

## Additional Slides

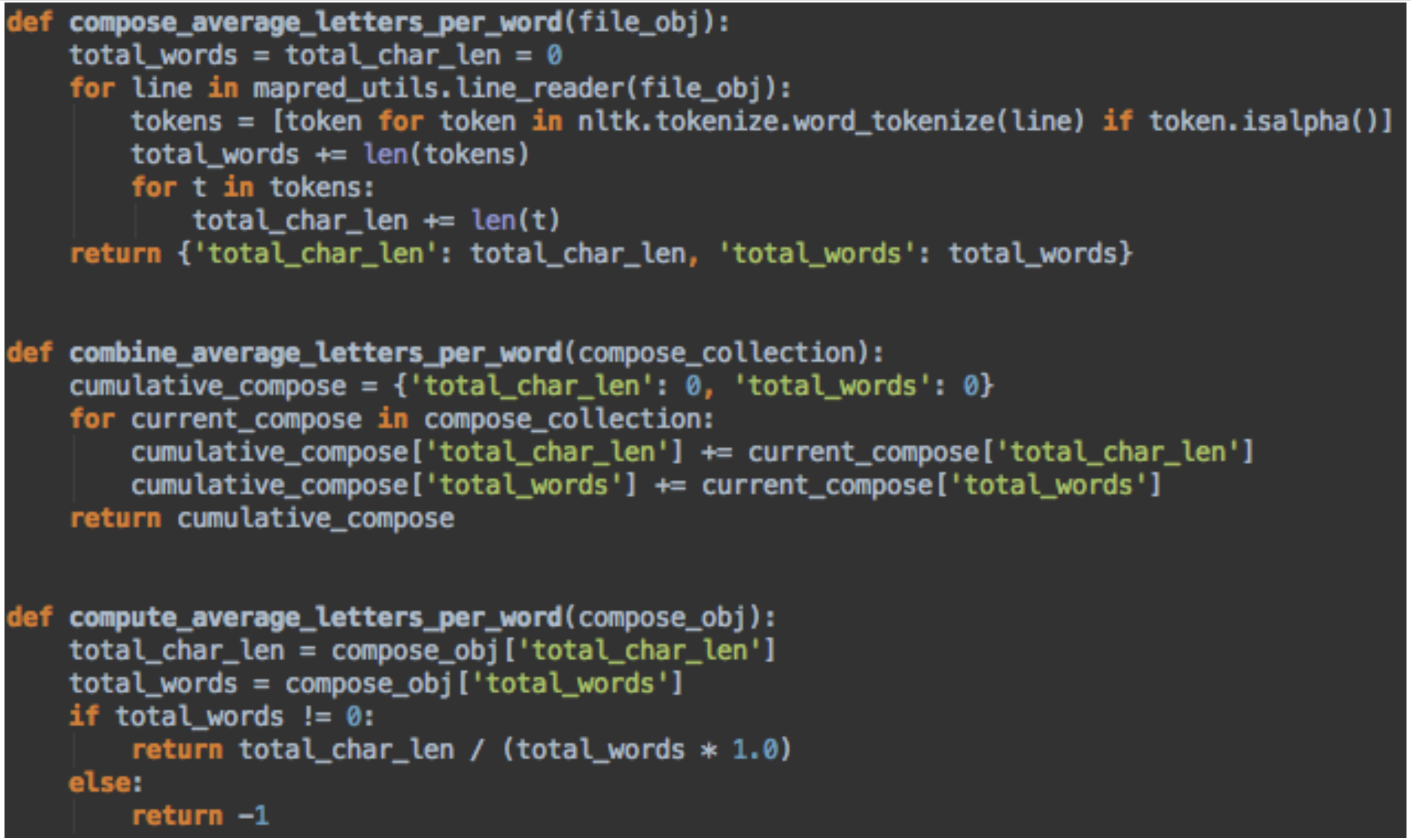
## Hadoop → Local Transition

| **Task** | **Method** | **Local Time** | **Cluster Time** |
|---|---|---|---|
| Named Entity Recognition | Stanford NER | > 4hours | 61m18.391s |
| Named Entity Recognition | OpenNLP | 8m26.338s | |

## MapReduce


Map Phase
Shuffle Phase
Reduce Phase
Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Mapper
Mapper
Mapper
Reducer
Reducer
Output 0
Output 1
Input
(HDFS)
Intermediate Results
Output
(HDFS)

## Compose → Combine → Compute



```python
def compose_average_letters_per_word(file_obj):
    total_words = total_char_len = 0
    for line in mapred_utils.line_reader(file_obj):
        tokens = [token for token in nltk.tokenize.word_tokenize(line) if token.isalpha()]
        total_words += len(tokens)
        for t in tokens:
            total_char_len += len(t)
    return {'total_char_len': total_char_len, 'total_words': total_words}


def combine_average_letters_per_word(compose_collection):
    cumulative_compose = {'total_char_len': 0, 'total_words': 0}
    for current_compose in compose_collection:
        cumulative_compose['total_char_len'] += current_compose['total_char_len']
        cumulative_compose['total_words'] += current_compose['total_words']
    return cumulative_compose


def compute_average_letters_per_word(compose_obj):
    total_char_len = compose_obj['total_char_len']
    total_words = compose_obj['total_words']
    if total_words != 0:
        return total_char_len / (total_words * 1.0)
    else:
        return -1
```