# Bag of Words Meets Bags of Popcorn

## Sentiment Analysis via Text Mining and Natural Language Processing

John Koo

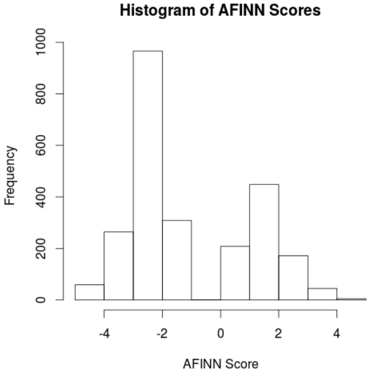
Tarleton State University

July 16, 2015

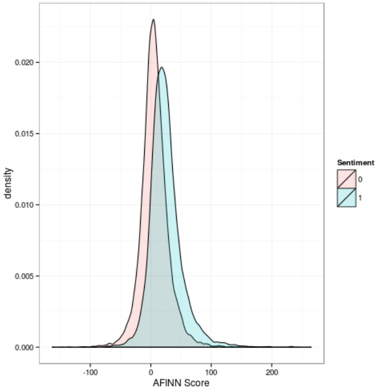
# Data Description

# AFINN List

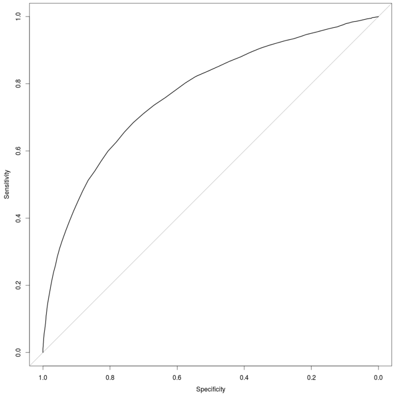
| word | score |
|---|---|
| invincible | 2 |
| mirthful | 3 |
| flops | −2 |
| hypocritical | −2 |
| upset | −2 |
| overlooked | −1 |
| hooligans | −2 |
| welcome | 2 |



# AFINN Score




Probability Densities by Sentiment




Bayes Classifier, AUC = .770

# Bag of Words

- Count up the number of times each term occurs in a review

| | no | good | war | great | bad | ... |
|---|---|---|---|---|---|---|
| Review 1 | 8 | 0 | 1 | 0 | 3 | ... |
| Review 2 | 2 | 0 | 1 | 1 | 0 | ... |
| Review 3 | 4 | 1 | 0 | 0 | 1 | ... |
| Review 4 | 4 | 1 | 0 | 1 | 0 | ... |
| Review 5 | 4 | 0 | 0 | 0 | 0 | ... |
| Review 6 | 3 | 3 | 0 | 0 | 1 | ... |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ |

# Bag of Words

- Count up the number of times each term occurs in a review
- Using AFINN list to start

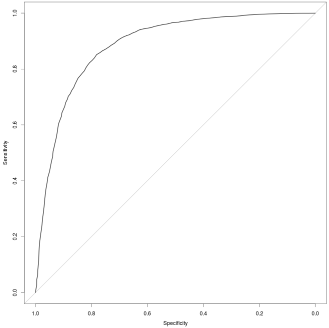
| | no | good | war | great | bad | ... |
|---|---|---|---|---|---|---|
| Review 1 | 8 | 0 | 1 | 0 | 3 | ... |
| Review 2 | 2 | 0 | 1 | 1 | 0 | ... |
| Review 3 | 4 | 1 | 0 | 0 | 1 | ... |
| Review 4 | 4 | 1 | 0 | 1 | 0 | ... |
| Review 5 | 4 | 0 | 0 | 0 | 0 | ... |
| Review 6 | 3 | 3 | 0 | 0 | 1 | ... |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ |

# Bag of Words

- Count up the number of times each term occurs in a review
- Using AFINN list to start

| | no | good | war | great | bad | ... |
|---|---|---|---|---|---|---|
| Review 1 | 8 | 0 | 1 | 0 | 3 | ... |
| Review 2 | 2 | 0 | 1 | 1 | 0 | ... |
| Review 3 | 4 | 1 | 0 | 0 | 1 | ... |
| Review 4 | 4 | 1 | 0 | 1 | 0 | ... |
| Review 5 | 4 | 0 | 0 | 0 | 0 | ... |
| Review 6 | 3 | 3 | 0 | 0 | 1 | ... |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ |

# Bag of Words




Random Forest Classifier, AUC = .886

# Text Frequency–Inverse Document Frequency

- Compare a term's relevance in a document to the inverse of its relevance in a collection of documents

# Text Frequency–Inverse Document Frequency

- Compare a term's relevance in a document to the inverse of its relevance in a collection of documents
- The more frequently a term occurs in a document, the more relevant it is to that document

# Text Frequency–Inverse Document Frequency

- Compare a term's relevance in a document to the inverse of its relevance in a collection of documents
- The more frequently a term occurs in a document, the more relevant it is to that document
- The more frequently a term occurs in a collection of documents, the less relevant it is to each document in the collection

# Text Frequency–Inverse Document Frequency

$$tf(t, d) = n(t|d)$$

# Text Frequency–Inverse Document Frequency

$$tf(t, d) = n(t|d)$$

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

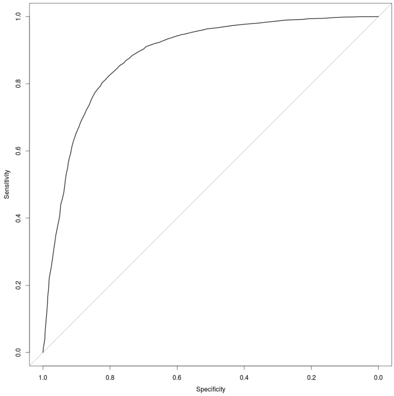
# Text Frequency–Inverse Document Frequency

$$tf(t,d) = n(t|d)$$

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

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

# Text Frequency–Inverse Document Frequency




AUC = .883

# Feature Extraction

- *A priori* feature extraction tends to perform poorly for simple analyses

# Feature Extraction

- *A priori* feature extraction tends to perform poorly for simple analyses
- It's typically better to learn features from the data themselves

# Term Frequency

| Word | Frequency |
|---:|---:|
| movie | 125,307 |
| film | 113,054 |
| one | 77,447 |
| like | 59,147 |
| just | 53,132 |
| good | 43,279 |
| ⋮ | ⋮ |

# Difference in Term Frequencies

| Word | Freq (Pos) | Freq (Neg) | Difference |
| ---: | ---: | ---: | ---: |
| movie | 18,139 | 23,668 | 5,529 |
| bad | 1,830 | 7,089 | 5,259 |
| great | 6,294 | 2,601 | 3,693 |
| just | 7,098 | 10,535 | 3,437 |
| even | 4,899 | 7,604 | 2,705 |
| worst | 246 | 2,436 | 2,190 |
| ⋮ | ⋮ | ⋮ | ⋮ |

# Normalized Difference *Sentiment* Index

$$NDSI := \frac{n(t|1) - n(t|0)}{n(t|1) + n(t|0)}$$

# Normalized Difference *Sentiment* Index

$$NDSI := \frac{(n(t|1) + \alpha) - (n(t|0) + \alpha)}{(n(t|1) + \alpha) + (n(t|0) + \alpha)}$$

# Normalized Difference *Sentiment* Index

$$NDSI := \frac{n(t|1) - n(t|0)}{n(t|1) + n(t|0) + 2\alpha}$$

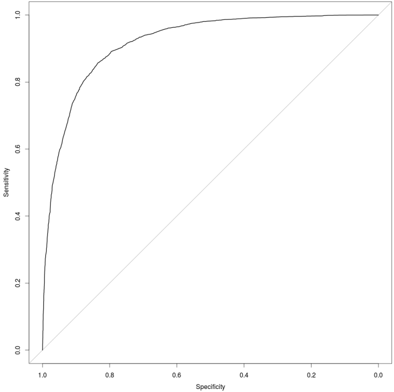
# Normalized Difference *Sentiment* Index

$$NDSI := \frac{|n(t|1) - n(t|0)|}{n(t|1) + n(t|0) + 2\alpha}$$

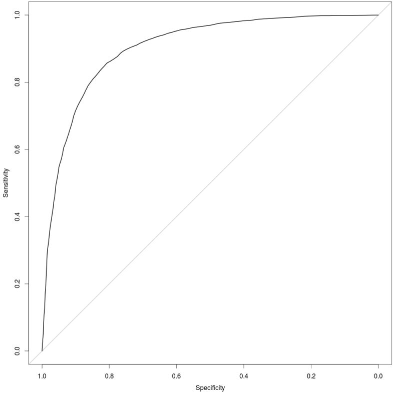
# Normalized Difference *Sentiment* Index

| Word | Freq (Pos) | Freq (Neg) | Difference | NDSI |
|---|---|---|---|---|
| worst | 246 | 2,436 | 2,190 | .745 |
| waste | 94 | 1,351 | 1,257 | .739 |
| poorly | 0 | 620 | 620 | .708 |
| lame | 0 | 618 | 618 | .691 |
| awful | 159 | 1,441 | 1,282 | .691 |
| mess | 0 | 498 | 498 | .660 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

# Normalized Difference *Sentiment* Index




Bag of Words, AUC = .919




tf-idf, AUC = .904

# Word Vectors

- Vector representation of words
- word $\sim \overrightarrow{v}_i = [v_{i1}, v_{i2}, \ldots, v_{iN}] \in V \subseteq \mathbb{R}^N$
- Relative word meanings reflected in vector representations

# Word Vectors

- Distributional hypothesis: Two words appear in similar contexts iff they share similar meaning

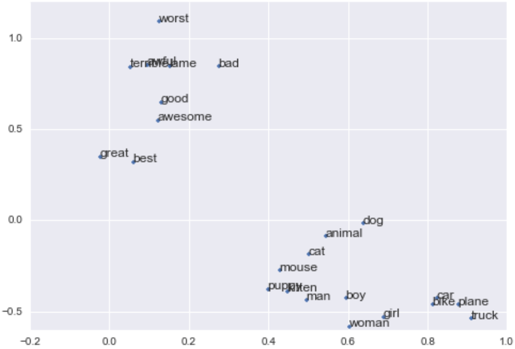
# Word Vectors

- Distributional hypothesis: Two words appear in similar contexts iff they share similar meaning
- Context similarity: If two words appear in similar contexts, then their vector representations are similar, i.e. $P(\vec{v}_i|c) \approx P(\vec{v}_j|c) \implies \vec{v}_i \approx \vec{v}_j$

# Word Vectors

- Distributional hypothesis: Two words appear in similar contexts iff they share similar meaning
- Context similarity: If two words appear in similar contexts, then their vector representations are similar, i.e. $P(\vec{v}_i|c) \approx P(\vec{v}_j|c) \implies \vec{v}_i \approx \vec{v}_j$
- Distributional hypothesis + context similarity $\implies$ If two words share similar meaning, then their vector representations are similar

# Word Vectors



# One Word Contexts (Bigrams)

- Multinomial Logistic (Softmax) Regression
- $P(\overrightarrow{v_j} | \overrightarrow{v_i}) = \frac{e^{\overrightarrow{\beta_j} \cdot \overrightarrow{v_i}}}{\sum_k^{|V|} e^{\overrightarrow{\beta_k} \cdot \overrightarrow{v_i}}}$
- Generalizable into multi-word contexts

# Word Similarity

- What do we mean by “similar”?

# Word Similarity

- What do we mean by “similar”?
- Cosine Similarity

# Word Similarity

- What do we mean by “similar”?
- Cosine Similarity
- $\text{sim}(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|}$

# Word Similarity

- What do we mean by “similar”?
- Cosine Similarity
- $\text{sim}(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|}$
- 
```
In [16]: model.most_similar('physics')
Out[16]: [(u'quantum', 0.5752027034759521),
          (u'laws', 0.4510610401630401 6),
          (u'scientific', 0.4351451992988586 4),
          (u'engineering', 0.4271385669708252),
          (u'gravity', 0.4245604276657104 5),
          (u'theory', 0.4180764555931091 3),
          (u'mechanics', 0.3903239369392395)]
```

# Analogies

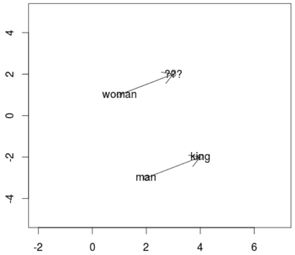
- Relative meanings and word relationships preserved in vector representations

# Analogies

- Relative meanings and word relationships preserved in vector representations
- MAN : KING :: WOMAN : ???

# Analogies

- Relative meanings and word relationships preserved in vector representations
- MAN : KING :: WOMAN : ???
- $\overrightarrow{v}_{king} - \overrightarrow{v}_{man} \approx \overrightarrow{x} - \overrightarrow{v}_{woman}$



# MAN : KING :: WOMAN : ???

- ```
  In [17]: model.most_similar(positive =
                                 ['king', 'woman'],
                                 negative = ['man'])
  ```
  ```

# MAN : KING :: WOMAN : ???

- ```
  In [17]: model.most_similar(positive =
                                   ['king', 'woman'],
                                   negative = ['man'])
  ```
- ```
  Out[17]: [(u'queen', 0.3589944541454315),
             (u'princess', 0.33725661039352417),
             (u'arthur', 0.29451817274093628),
             (u'mistress', 0.29320359230041504),
             (u'france', 0.29167920351028442),
             (u'lion', 0.29003939032554626),
             (u'throne', 0.28948855400085449),
             (u'kong', 0.27626261115074158),
             (u'kingdom', 0.26161640882492065),
             (u'prince', 0.26111793518066406)]
  ```

# Document Vectors

- Combine word vectors into one document vector

# Document Vectors

- Combine word vectors into one document vector
- $f(\{\overrightarrow{v}_1, \overrightarrow{v}_2, \ldots \overrightarrow{v}_k\}) = \overrightarrow{d}$

# Document Vectors

- Combine word vectors into one document vector
- $f(\{\vec{v}_1, \vec{v}_2, \ldots \vec{v}_k\}) = \vec{d}$
- Document vectors live in the same space as word vectors

# Document Vectors

- Combine word vectors into one document vector
- $f(\{\overrightarrow{v}_1, \overrightarrow{v}_2, \ldots \overrightarrow{v}_k\}) = \overrightarrow{d}$
- Document vectors live in the same space as word vectors
- bad $\approx$ not good $\implies \overrightarrow{v}_{\text{bad}} \approx \overrightarrow{v}_{\text{not good}}$

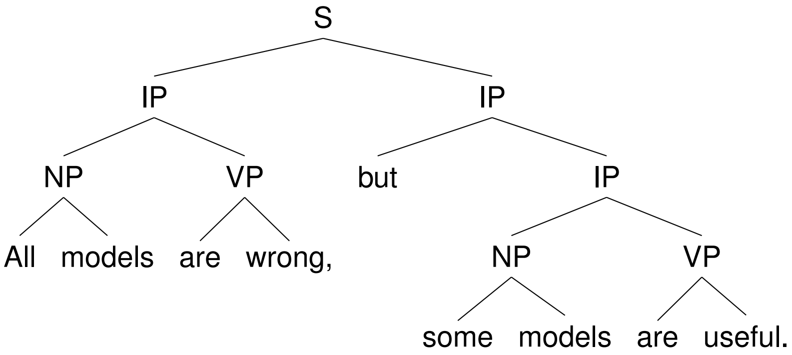
# Document Vectors

- Combine word vectors into one document vector
- $f(\{\overrightarrow{v}_1, \overrightarrow{v}_2, \ldots \overrightarrow{v}_k\}) = \overrightarrow{d}$
- Document vectors live in the same space as word vectors
- bad $\approx$ not good $\implies \overrightarrow{v}_{\text{bad}} \approx \overrightarrow{v}_{\text{not good}}$

# Document Vectors

- Combine word vectors into one document vector
- $f(\{\overrightarrow{v}_1, \overrightarrow{v}_2, \ldots \overrightarrow{v}_k\}) = \overrightarrow{d}$
- Document vectors live in the same space as word vectors
- bad $\approx$ not good $\implies \overrightarrow{v}_{\text{bad}} \approx \overrightarrow{v}_{\text{not good}}$
- Syntax trees

# Syntax Trees



# References


- "Bag of Words Meets Bags of Popcorn." Kaggle. https://www.kaggle.com/c/word2vec-nlp-tutorial
- "AFINN list." F Nielson. The Technical University of Denmark. http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
- "Tf-idf Weighting." The Stanford Natural Language Processing Group. C Manning, P Raghavan. H Schütze, http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html
- "Deep Learning for Natural Language Processing (Without Magic)." R Socher, Y Bengio, C Manning. The Stanford Natural Language Processing Group. http://nlp.stanford.edu/courses/NAACL2013/
- "word2vec Parameter Learning Explained." X Rong. arXiv:1411.2738v1.


Thank you!