Text Classification

From Logistic Regression to Neural Networks

Yangfeng Ji

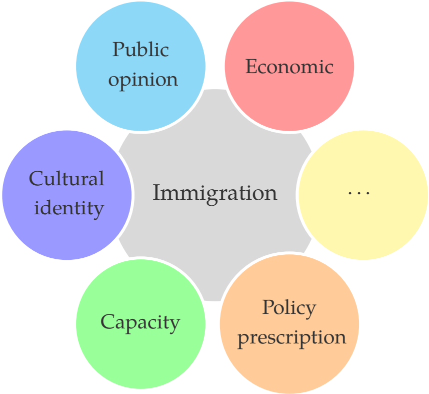
May 16, 2018

Paul G. Allen School of Computer Science & Engineering
University of Washington

Case I: Like a business?

Case II: Support a piece of legislation?

Case III: Frame a topic?



(Card *et al.*, 2015)

Analysis as Classification

- Input: a text d
- Output: $y \in \mathcal{Y}$, where $\mathcal{Y}$ is the predefined category set

$$\text{Text } d \longrightarrow \boxed{\text{Classifier}} \longrightarrow \text{Category } y$$

Overview

- Case study: Sentiment analysis
  - Logistic regression
  - Bag-of-words representation
- Neural Network Models
  - Summation over word embeddings
  - Recurrent neural networks
  - Recursive neural networks

Case Study: Sentiment Analysis

Sentiment Analysis

Task: predicting user rating based on the review

Applications of sentiment analysis [Liu, 2012]

A Simple Predictor

Example I

Super quick and really friendly staff. I like starting off my mornings at this store!!

- SentiWordnet: a publicly available word sentiment polarity dictionary.

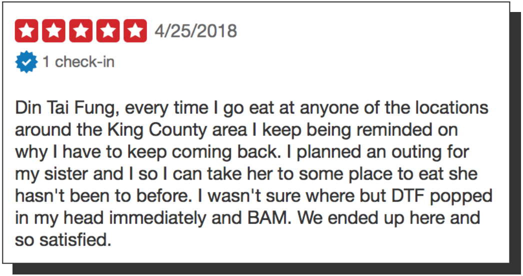
Another Example

Example II

Din Tai Fung, every time I go eat at anyone of the locations around the King County area,
I keep being reminded on why I have to keep coming back to this restaurant.
…

- No signal word

Data Driven Approach



- Discover the relationship between *words* and *sentiment polarity* (user rating) from data
- Need a collection of texts and their category labels

Basic Idea of Statistical Learning

Given a collection of data points and their labels

- Principle: Discover the *statistical* relationship between patterns and categories from training data.
- Goal: To make better decisions for *unseen* data points in a test set

Standard setup

- Training set $\mathcal{T} = \{d_n, y_n\}_{n=1}^{N}$
- Test set $\mathcal{U}$

Basic Idea of Statistical Learning

Given a collection of data points and their labels

- Principle: Discover the *statistical* relationship between patterns and categories from training data.
- Goal: To make better decisions for *unseen* data points in a test set

Standard setup

- Training set $\mathcal{T} = \{d_n, y_n\}_{n=1}^{N}$
- Development set $\mathcal{D}$
- Test set $\mathcal{U}$

Example Dataset

A subset of the Yelp Dataset

`https://www.yelp.com/dataset/challenge`

	Training	Development	Test
Documents	40K	5K	5K
Words	4.7M	0.5M	0.6M

- 5 classes (user rating from 1 to 5)
- Code available on
  `https://github.com/jiyfeng/textclassification`

A Simple Framework

Decision function

$$h_y(d) = \boldsymbol{w}_y^\top \boldsymbol{f}(d) \tag{1}$$

- $\boldsymbol{f}(d) : d \rightarrow \mathcal{R}^n$ mapping text d into a numeric vector
- $\boldsymbol{w}_y$ classification weight associated with category label y

A Simple Framework

Decision function

$$h_y(d) = \boldsymbol{w}_y^\top \boldsymbol{f}(d) \tag{1}$$

- $\boldsymbol{f}(d) : d \rightarrow \mathcal{R}^n$ mapping text d into a numeric vector
- $\boldsymbol{w}_y$ classification weight associated with category label y

Within this framework

- Logistic regression
- Support vector machine
- Neural network classifiers
- $\cdots$

Decision Rule

Decision rule

$$\hat{y} = \arg\max_{y} h_y(d) \tag{2}$$

$$= \arg\max_{y} \boldsymbol{w}_y^\top \boldsymbol{f}(d) \tag{3}$$

Two questions

- Learning: How to determine $\{\boldsymbol{w}_y\}$?
- Feature representation: How to construct the mapping $\boldsymbol{f}(d)$?

Supervised Learning Pipeline

Given training set $\{d_n, y_n\}$

- Predict the label of d_n as $\hat{y}_n$ using the decision function, $\forall n$
- Define the loss function as

$$\sum_n L_w(y_n, \hat{y}_n)$$

- Learn $\{\boldsymbol{w}_y\}$ by optimizing the following

$$\min_{\{\boldsymbol{w}_y\}} L_w(y_n, \hat{y}_n)$$

Logistic Regression

$$\log P(y|d) \propto \boldsymbol{w}_y^\top \boldsymbol{f}(d) \tag{4}$$

Sklearn function

`sklearn.linear_model.LogisticRegression`

Overview

- ✓ Case study: Sentiment analysis
  - ✓ Logistic regression
  - Bag-of-words representation
- Neural Network Models
  - Summation over word embeddings
  - Recurrent neural networks
  - Recursive neural networks

Bag-of-Words Representation

From texts to bag of words:

Super quick and really friendly staff. I like starting off my mornings at this store!! $\Rightarrow$ SUPER,QUICK,AND, REALLY, FRIENDLY, $\cdots$, STORE

NLTK function

`nltk.tokenize.wordpunct_tokenize`

Bag-of-Words Representation (Cont.)

Given the texts from training set, build a vocab first

$$\left\{\begin{array}{c} \text{SUPER} \\ \cdots \\ \text{QUICK} \\ \text{FOOD} \\ \text{FRIENDLY} \\ \text{EAT} \\ \cdots \\ \text{STAFF} \end{array}\right\} \tag{5}$$

Bag-of-Words Representation (Cont.)

Based on the vocab, convert each text into a numeric vector

Example I

Super quick and really friendly staff. I like starting off my mornings at this store!!

$$\left\{\begin{matrix} \text{SUPER} \\ \cdots \\ \text{QUICK} \\ \text{FOOD} \\ \text{FRIENDLY} \\ \text{EAT} \\ \cdots \\ \text{DELICIOUS} \end{matrix}\right\} \qquad \begin{bmatrix} 1 \\ \cdots \\ 1 \\ 0 \\ 1 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

Building BoW Representations

Sklearn function

`sklearn.feature_extraction.text.CountVectorizer`

- Given a collection of texts, it will build a vocab and also convert all texts into numeric vectors
- With Logistic Regression, the classification performance on dev data is 61.4%

Interpretability

Weights learned from training data

$$
\text{Vocab} \quad \left\{ \begin{array}{c} \textsc{super} \\ \cdots \\ \textsc{quick} \\ \textsc{food} \\ \textsc{friendly} \\ \textsc{eat} \\ \cdots \\ \textsc{delicious} \end{array} \right\}
\qquad
w_{\text{rating}=1} \quad \begin{bmatrix} 0.33 \\ \cdots \\ -1.26 \\ 0.08 \\ -2.57 \\ -0.47 \\ \cdots \\ -3.60 \end{bmatrix}
\qquad
w_{\text{rating}=1} \quad \begin{bmatrix} -0.09 \\ \cdots \\ -0.01 \\ -0.09 \\ 0.16 \\ 0.00 \\ \cdots \\ 0.64 \end{bmatrix}
$$

Interpretability (II)

Top features

rating = 5	rating = 1
exceptional	worst
incredible	joke
phenomenal	disgusted
body	unprofessional
regret	garbage
worried	disgusting
skeptical	*luck*
hesitate	pathetic
happier	apologies
mike	horrible

- wouldn't *regret*?
- bad *luck*?

How Far We Can Go with BoW (I)

Uni-gram vs. bi-gram

{SUPER, QUICK}

VS.

{SUPER QUICK}

Sklearn function

`sklearn.feature_extraction.text.CountVectorizer` with `ngram_range=(1,2)`

- Even larger vocab size: from 60K to 1M
- Performance change: from 61.4% to 62.4% on dev data

How Far We Can Go with BoW (II)

Tricks to reduce vocab size

- remove punctuation (default)
- lowercase (↗)
- remove low-frequency words (↗)
- remove high-frequency words (↘)
- replace numbers with a special token

Attention

- Not always helpful (these are empirical tricks)
- Not always the case (it depends on the data/domain)

Feature Selection

With supervision information,

- select informative features ($\nearrow$)

Sklearn function

`sklearn.feature_selection.mutual_info_classif`

- train with additional objectives ($\nearrow$)

$$\sum_n \mathscr{L}(y_n, \hat{y}_n) + \lambda \|\boldsymbol{w}\|_1 \tag{6}$$

Sklearn function

`LogisticRegression` with `penalty=l1` and $C = \frac{1}{\lambda} < 1.0$

Summary of BoW Representations

- Simple and powerful
- Extend it with bi-gram features
- Preprocessing and feature selection

Fundamental Limitation of BoW

$$\text{Vocab} \qquad\qquad\qquad\qquad w_{\text{rating}=5}$$

$$\left\{\begin{array}{c} \text{SUPER} \\ \dots \\ \text{QUICK} \\ \text{YUMMY} \\ \text{FRIENDLY} \\ \text{EAT} \\ \dots \\ \text{DELICIOUS} \end{array}\right\} \qquad\qquad \begin{bmatrix} \\ \dots \\ \\ 0.06 \\ \\ \\ \dots \\ 0.64 \end{bmatrix}$$

Problem

- Anything a model learned about DELICIOUS cannot be transferred to similar words, like YUMMY

Overview

- ✓ Case study: Sentiment analysis
  - ✓ Logistic regression
  - ✓ Bag-of-words representation
- ▸ Neural Network Models
  - ▸ Summation over word embeddings
  - ▸ Recurrent neural networks
  - ▸ Recursive neural networks

Deep Learning for Text Classification

How to Represent a Word?

$$v_{\text{delicious}} = \begin{bmatrix} 0 \\ \cdots \\ 0 \\ 0 \\ 0 \\ 0 \\ \cdots \\ 1 \end{bmatrix} \in \mathscr{R}^{\sim 30K}$$

$$\left\{ \begin{array}{c} \text{SUPER} \\ \cdots \\ \text{QUICK} \\ \color{red}{\text{YUMMY}} \\ \text{FRIENDLY} \\ \text{EAT} \\ \cdots \\ \color{blue}{\text{DELICIOUS}} \end{array} \right\}$$

Semantic Similarity

- One-hot representation

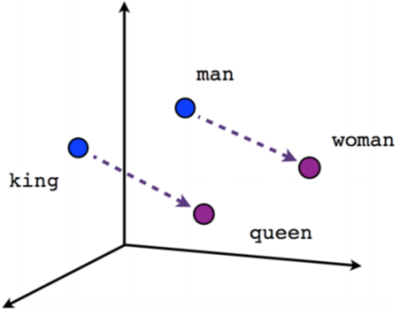
$$\|v_{\text{delicious}} - v_{\text{yummy}}\|_2 = \|v_{\text{delicious}} - v_{\text{super}}\|_2 \tag{7}$$

- What we need

$$\|v_{\text{delicious}} - v_{\text{yummy}}\|_2 > \|v_{\text{delicious}} - v_{\text{super}}\|_2 \tag{8}$$

Similar conclusion with cosine similarity

Dense Representation of Words



$$\text{king} - \text{man} + \text{woman} \approx \text{queen} \tag{9}$$

[Mikolov *et al.*, 2013]

How to Get Dense Representations?

Three off-the-shelf tools

- Word2vec
  `https://github.com/dav/word2vec/`
- GloVe
  `https://nlp.stanford.edu/projects/glove/`
- ELMo (from **AllenNLP**)
  `http://allennlp.org/elmo`

In general, these are called *word embeddings*

Word Similarity Examples

Top five similar words:

yummy	horrible
delicious	terrible
tasty	poor
delish	awful
yum	customer
incredible	exceptional

Dense Representation of Texts

How to represent texts with word embeddings?

- Addition operation
- Recurrent neural networks
- Recursive neural networks

Classification Framework: Review

- Decision function

$$h_y(d) = \boldsymbol{w}_y^\top \boldsymbol{f}(d) \tag{10}$$

- Decision rule

$$\begin{aligned} \hat{y} &= \arg\max_y h_y(d) && (11) \\ &= \arg\max_y \boldsymbol{w}_y^\top \boldsymbol{f}(d) && (12) \end{aligned}$$

Central Question

How to learn $\boldsymbol{f}(d)$ automatically and efficiently?

Three Examples

- fastText
- Hierarchical recurrent neural networks
- Recursive neural networks with document structure

Approach I

Summation of dense representations

Let $\boldsymbol{x}$ be the bag-of-words representation of text d

$$\boldsymbol{v}_d = \sum_i x_i \cdot \boldsymbol{w}_i \tag{13}$$

Example

Example I

Super quick and really friendly staff. I like starting off my mornings at this store!!

$$v_{\text{text}} = v_{\text{super}} + \cdots + v_{\text{store}} \tag{14}$$

Classification

Decision function

$$h(d) = \mathbf{w}_y^\top \boldsymbol{v}_{\text{text}} \tag{15}$$

Alternative way of learning word representations

- joint training with the final task — fastText

fastText

fastText

Library for efficient text classification and representation learning

GET STARTED | DOWNLOAD MODELS

What is fastText?

FastText is an open-source, free, lightweight library that allows users to learn text representations and text classifiers. It works on standard, generic hardware. Models can later be reduced in size to even fit on mobile devices.

- Dev performance: %65.1 (best dev performance from LR + BoW: $\sim$ %61)
- No interpretability

Extension

Example I

Super quick and really friendly staff. I like starting off my mornings at this store!!

$$v_{\text{text}} = v_{\text{super}} + \cdots + v_{\text{store}} + v_{\text{super fast}} + \cdots + v_{\text{this store}}$$

Extension

Example I

Super quick and really friendly staff. I like starting off my mornings at this store!!

$$v_{\text{text}} = v_{\text{super}} + \cdots + v_{\text{store}} + v_{\text{super fast}} + \cdots + v_{\text{this store}}$$

About v_{text} constructed in this way

- give really competitive baselines on text classification
- a low-dimensional dense representation
- limited contextual information

Overview

- ✓ Case study: Sentiment analysis
  - ✓ Logistic regression
  - ✓ Bag-of-words representation
- ▸ Neural Network Models
  - ✓ Summation over word embeddings
  - ▸ Recurrent neural networks
  - ▸ Recursive neural networks

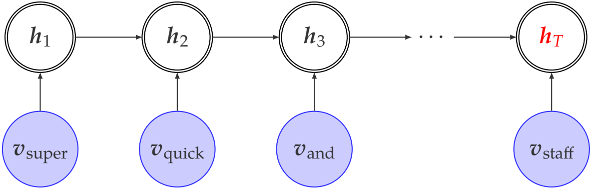
Approach II

Recurrent neural networks

- To capture large contextual information

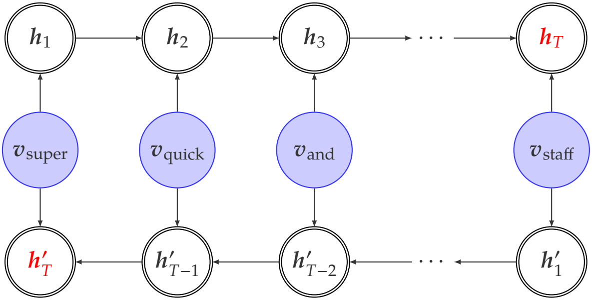
Example I

Super quick and really friendly staff. I like starting off my mornings at this store!!



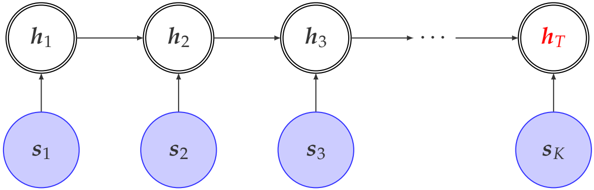
- $\boldsymbol{h}_T$: sentence representation

Bi-directional Recurrent Neural Networks



- $[h_T, h'_T]$: sentence representation

Text Representation with Hierarchical RNNs



Each s_k is from a RNN

Online Resources

- Paper: **Hierarchical Attention Networks for Document Classification**

  `http://aclweb.org/anthology/N/N16/N16-1174.pdf`
- Code

  `https://github.com/richliao/textClassifier`

Overview

- ✓ Case study: Sentiment analysis
  - ✓ Logistic regression
  - ✓ Bag-of-words representation
- ▸ Neural Network Models
  - ✓ Summation over word embeddings
  - ✓ Recurrent neural networks
  - ▸ Recursive neural networks

Approach III

Recursive neural networks

- Prioritize parts of texts

Case I: Another Review

A restaurant review

Although the food was amazing and I was in love with the spicy pork burrito, the service was really awful.
We watched our waiter serve himself many drinks.
He kept running into the bathroom instead of grabbing our bill.

- User rating: 2

Case II: Support a piece of legislation?

Congressional floor speech: short example

Mr. Chairman, I demand a recorded vote.

Case II: Support a piece of legislation?

Congressional floor speech: long example

Mr. Speaker, I yield myself such time as I may consume.

Mr. Speaker, I thank the gentleman from texas $\cdots$.

Mr. Speaker, the house is an institution built upon its rules.

$\cdots$ *(3,208 words ommitted)*

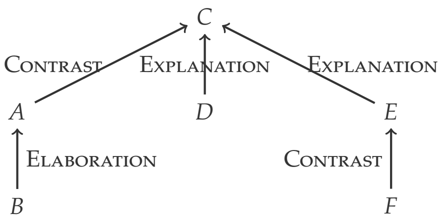
I urge support of this package of rules.

$\ldots$

Tree Structure

A restaurant review

[Although the food was amazing]^A [and I was in love with the spicy pork burrito,]^B [the service was really awful.]^C [We watched our waiter serve himself many drinks.]^D [He kept running into the bathroom]^E [instead of grabbing our bill.]^F

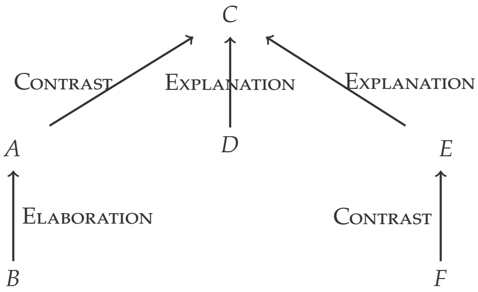


Discourse parsing: get structural information automatically (Ji and Eisenstein, 2014)

Document Representation through Composition

Composition function

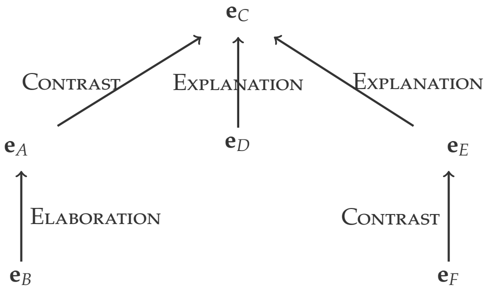
$$f(\text{parent node}, \sum \text{children nodes})$$



Document Representation through Composition

Composition function

$$f(\text{parent node}, \sum \text{children nodes})$$



Document Representation through Composition

Composition function

$$f(\text{parent node}, \sum \text{children nodes})$$

$$\begin{tikzcd}
 & & \mathbf{e}_C & & \\
\mathbf{e}_A \arrow[urr, "\textsc{Contrast}"] & & \mathbf{v}_D = f(\mathbf{e}_D) \arrow[u, "\textsc{Explanation}"] & & \mathbf{e}_E \arrow[ull, "\textsc{Explanation}"'] \\
\mathbf{v}_B = f(\mathbf{e}_B) \arrow[u, "\textsc{Elaboration}"'] & & & & \mathbf{v}_F = f(\mathbf{e}_F) \arrow[u, "\textsc{Contrast}"]
\end{tikzcd}$$

Document Representation through Composition

Composition function

$$f(\text{parent node}, \sum \text{children nodes})$$

$$\begin{tikzcd}
 & \mathbf{e}_C & \\
\textcolor{blue}{\mathbf{v}_A} = f(\mathbf{e}_A, \alpha_{A,B}\mathbf{W}_{r_{(A,B)}}\textcolor{blue}{\mathbf{v}_B}) \arrow[ur, "\textsc{Contrast}"] & \textcolor{blue}{\mathbf{v}_D} = f(\mathbf{e}_D) \arrow[u, "\textsc{Explanation}"] & \textcolor{blue}{\mathbf{v}_E} = f(\mathbf{e}_E, \alpha_{F,E}\mathbf{W}_{r_{(F,E)}}\textcolor{blue}{\mathbf{v}_F}) \arrow[ul, "\textsc{Explanation}"'] \\
\textcolor{blue}{\mathbf{v}_B} = f(\mathbf{e}_B) \arrow[u, "\mathbf{W}_{r_{(A,B)}}"'] & & \textcolor{blue}{\mathbf{v}_F} = f(\mathbf{e}_F) \arrow[u, "\mathbf{W}_{r_{(F,E)}}"]
\end{tikzcd}$$

Document Representation through Composition

Composition function

$$f(\text{parent node}, \sum \text{children nodes})$$

$$
\begin{tikzcd}
 & & \mathbf{v}_{doc} = f(\mathbf{e}_C, \sum_{j \in \{A,D,E\}} \alpha_{C,j} \mathbf{W}_{r_{(C,j)}} \mathbf{v}_j) & & \\
\mathbf{v}_A = f(\mathbf{e}_A, \alpha_{A,B} \mathbf{W}_{r_{(A,B)}} \mathbf{v}_B) \arrow[urr, "\mathbf{W}_{r_{(C,A)}}"] & & \mathbf{v}_D = f(\mathbf{e}_D) \arrow[u, "\mathbf{W}_{r_{(C,D)}}"] & & \mathbf{v}_E = f(\mathbf{e}_E, \alpha_{F,E} \mathbf{W}_{r_{(F,E)}} \mathbf{v}_F) \arrow[ull, "\mathbf{W}_{r_{(C,E)}}"'] \\
\mathbf{v}_B = f(\mathbf{e}_B) \arrow[u, "\mathbf{W}_{r_{(A,B)}}"'] & & & & \mathbf{v}_F = f(\mathbf{e}_F) \arrow[u, "\mathbf{W}_{r_{(F,E)}}"]
\end{tikzcd}
$$

Online Resources

- Paper: **Neural Discourse Structure for Text Categorization**
  `https://arxiv.org/pdf/1702.01829.pdf`
- Code:
  - Discourse parsing
    `https://github.com/jiyfeng/DPLP`
  - Classification
    `https://github.com/jiyfeng/disco4textcat`

Overview

- ✓ Case study: Sentiment analysis
  - ✓ Logistic regression
  - ✓ Bag-of-words representation
- ✓ Neural Network Models
  - ✓ Summation over word embeddings
  - ✓ Recurrent neural networks
  - ✓ Recursive neural networks

Conclusion

Summary

Model	Power ranking	Interpretability	Large context	Text priority
LR + BoW	4	Yes	No	No
fastText	3	No	No	No
Recurrent NNs	2	No	Yes	No
recursive NNs	1	Yes*	Yes	Yes

* from discourse parsing

Thank You!

Webpage: `http://yangfengji.net`

Code: `https://github.com/jiyfeng/textclassification`