TTIC 31190: Natural Language Processing

Kevin Gimpel

Spring 2018

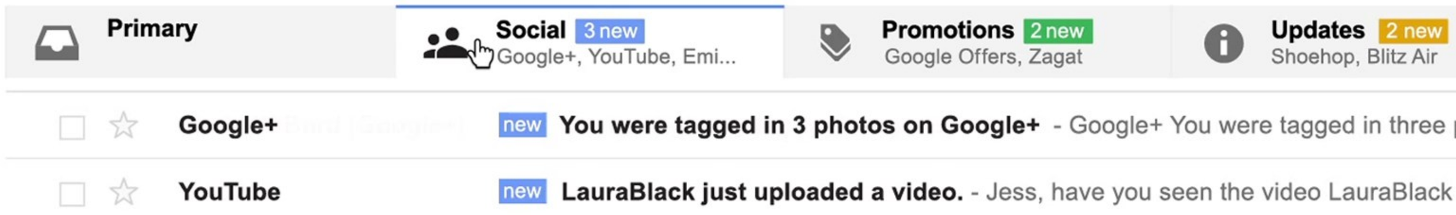
Lecture 4: Text Classification

Roadmap

- words, morphology, lexical semantics
- text classification
- simple neural methods for NLP
- language modeling and word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- semantics, compositionality, semantic parsing
- machine translation and other NLP tasks

Text Classification

- simplest user-facing NLP application
- email (spam, priority, categories):



- sentiment:

- topic classification
- others?

Text Classification

- datasets
- classification
  - modeling
  - inference
  - learning

NLP Datasets

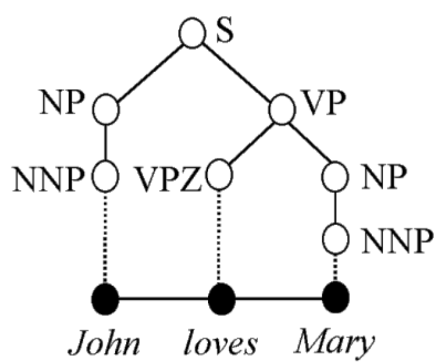
- NLP datasets include inputs (usually text) and outputs (usually some sort of annotation)

Annotation

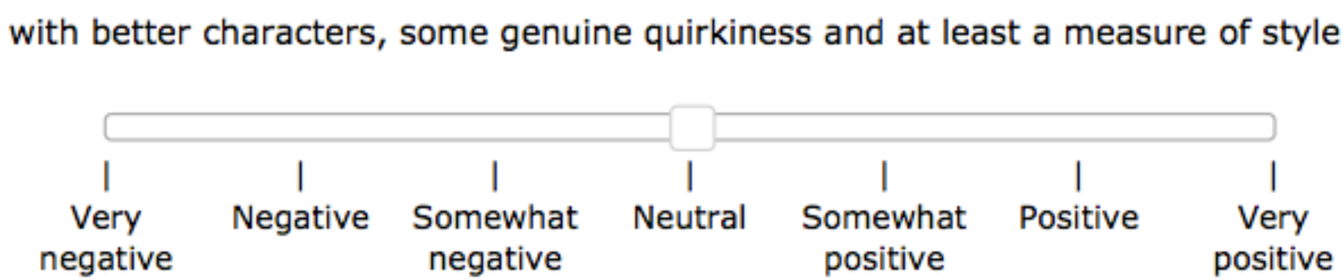
- supervised machine learning needs labeled datasets, where labels are called ground truth
- in NLP, labels are annotations provided by humans
- there is always some disagreement among annotators, even for simple tasks
- these annotations are called a gold standard, not ground truth

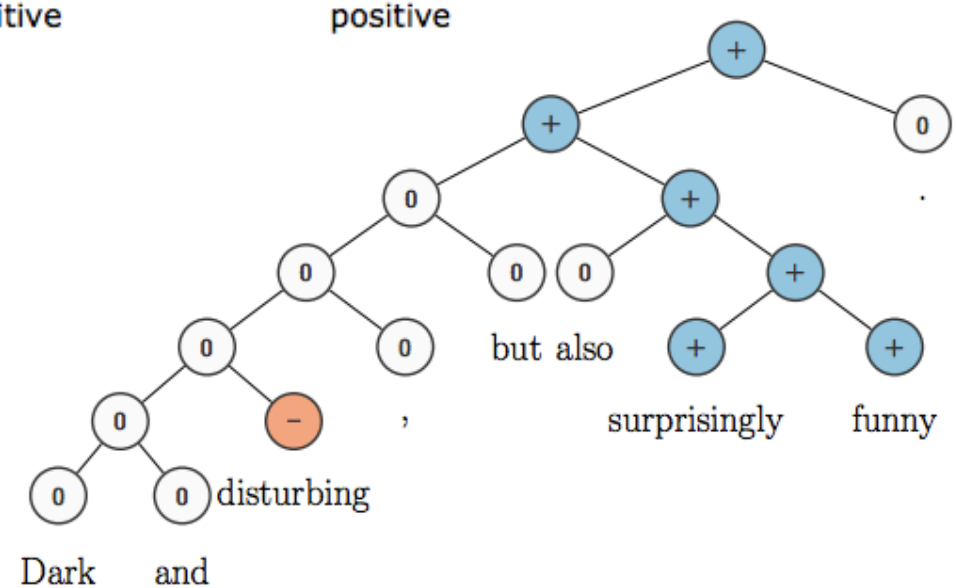
How are NLP datasets developed?

1. paid, trained human annotators
   – traditional approach
   – researchers write annotation guidelines, recruit & pay annotators (often linguists)
   – more consistent annotations, but costly to scale
   – e.g., Penn Treebank (1993)
     • 1 million words, mostly Wall Street Journal, annotated with part-of-speech tags and syntactic parse trees



2. crowdsourcing (e.g., Amazon Mechanical Turk)
   - more recent trend
   - can’t really train annotators, but easier to get multiple annotations for each input (which can then be averaged)
   - e.g., Stanford Sentiment Treebank:





3. naturally-occurring annotation

– long history: used by IBM for speech recognition and statistical machine translation

There's No Data Like More Data

Dick Garwin's correspondence	~2.5M words
Associated Press	20M words
Oil company	25M words
Federal Register	??M words
American Printing House for the Blind	60M words
IBM Deposition	100M words
Canadian Hansard English	100M words

credit: Brown & Mercer, 20 Years of Bitext Workshop, 2013

3. naturally-occurring annotation

– long history: used by IBM for speech recognition and statistical machine translation

There's No Data Like More Data

• Dick Garwin's correspondence	~2.5M words
• Associated Press	20M words
• Oil company	25M words
• Federal Register	??M words
• American Printing House for the Blind	60M words
• IBM Deposition	100M words
• Canadian Hansard English	100M words

credit: Brown & Mercer, 20 Years of Bitext Workshop, 2013

– how might you find naturally-occurring data for:

- conversational agents
- summarization
- coreference resolution

Annotator Agreement

- given annotations from two annotators, how should we measure inter-annotator agreement?

Annotator Agreement

- given annotations from two annotators, how should we measure inter-annotator agreement?
  - percent agreement?

Annotator Agreement

- given annotations from two annotators, how should we measure inter-annotator agreement?
  - percent agreement?
  - Cohen's Kappa (Cohen, 1960) accounts for agreement by chance
  - generalizations exist for more than two annotators (Fleiss, 1971)

Text Classification Data

- There are many annotated datasets
  - Stanford Sentiment Treebank: fine-grained sentiment analysis of movie reviews
  - subjectivity/objectivity sentence classification
  - binary sentiment analysis of customer reviews
  - TREC question classification

- Subjectivity/objectivity classification:

the hulk is an anger fueled monster with incredible strength and resistance to damage .	
in trying to be daring and original , it comes off as only occasionally satirical and never fresh .	
solondz may well be the only one laughing at his own joke	
obstacles pop up left and right , as the adventure gets wilder and wilder .	

- Subjectivity/objectivity classification:

the hulk is an anger fueled monster with incredible strength and resistance to damage .	objective
in trying to be daring and original , it comes off as only occasionally satirical and never fresh .	subjective
solondz may well be the only one laughing at his own joke	
obstacles pop up left and right , as the adventure gets wilder and wilder .	

- Subjectivity/objectivity classification:

the hulk is an anger fueled monster with incredible strength and resistance to damage .	objective
in trying to be daring and original , it comes off as only occasionally satirical and never fresh .	subjective
solondz may well be the only one laughing at his own joke	subjective
obstacles pop up left and right , as the adventure gets wilder and wilder .	objective

- Subjectivity/objectivity classification:

the hulk is an anger fueled monster with incredible strength and resistance to damage .	objective
in trying to be daring and original , it comes off as only occasionally satirical and never fresh .	subjective
solondz may well be the only one laughing at his own joke	subjective
obstacles pop up left and right , as the adventure gets wilder and wilder .	objective

- How was this dataset generated?

- Subjectivity/objectivity classification:

the hulk is an anger fueled monster with incredible strength and resistance to damage .	objective
in trying to be daring and original , it comes off as only occasionally satirical and never fresh .	subjective
solondz may well be the only one laughing at his own joke	subjective
obstacles pop up left and right , as the adventure gets wilder and wilder .	objective

- How was this dataset generated?
  - IMDB plot summaries: objective
  - Rotten Tomatoes snippets: subjective

- customer review sentiment classification:

it works with a minimum of fuss .	
i 've had this thing just over a month and the headphone jack has already come loose .	
size - bigger than the ipod	
you can manage your profile , change the contrast of backlight , make different type of display , either list or tabbed .	
i replaced it with a router raizer and it works much better .	

- customer review sentiment classification:

it works with a minimum of fuss .	positive
i 've had this thing just over a month and the headphone jack has already come loose .	negative
size - bigger than the ipod	
you can manage your profile , change the contrast of backlight , make different type of display , either list or tabbed .	
i replaced it with a router raizer and it works much better .	

- customer review sentiment classification:

it works with a minimum of fuss .	positive
i 've had this thing just over a month and the headphone jack has already come loose .	negative
size - bigger than the ipod	1(a)
you can manage your profile , change the contrast of backlight , make different type of display , either list or tabbed .	1(b)
i replaced it with a router raizer and it works much better .	1(c)

- customer review sentiment classification:

it works with a minimum of fuss .	positive
i 've had this thing just over a month and the headphone jack has already come loose .	negative
size - bigger than the ipod	negative
you can manage your profile , change the contrast of backlight , make different type of display , either list or tabbed .	positive
i replaced it with a router raizer and it works much better .	negative

- question classification:

Who invented baseball ?	human
CNN is an acronym for what ?	abbreviation
Which Latin American country is the largest ?	location
How many small businesses are there in the U.S .	number
What would you add to the clay mixture to produce bone china ?	entity
What is the root of all evil ?	description

Text Classification

- datasets
- classification
  - modeling
  - inference
  - learning

What is a classifier?

What is a classifier?

- a function from inputs $\boldsymbol{x}$ to classification labels y

What is a classifier?

- a function from inputs ***x*** to classification labels *y*
- one simple type of classifier:
  - for any input ***x***, assign a score to each label *y*, parameterized by parameters ***w***:

$$\mathrm{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

Notation

$\mathbf{u} =$ a vector

$u_i =$ entry i in the vector

Notation

$\mathbf{u} =$ a vector

$u_i =$ entry i in the vector

$\boldsymbol{x} =$ a structured object

$x_i =$ entry i in the structured object

What is a classifier?

- a function from inputs $\boldsymbol{x}$ to classification labels y
- one simple type of classifier:
  - for any input $\boldsymbol{x}$, assign a score to each label y, parameterized by parameters $\boldsymbol{w}$:

$$\mathrm{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

  - classify by choosing highest-scoring label:

$$\mathrm{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y} \ \mathrm{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

Course Philosophy

- From reading papers, one gets the idea that machine learning concepts are monolithic, opaque objects
  - e.g., naïve Bayes, logistic regression, SVMs, CRFs, neural networks, LSTMs, etc.
- Nothing is opaque
- Everything can be dissected, which reveals connections
- The names above are useful shorthand, but not useful for gaining understanding

Modeling, Inference, Learning

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{y}{\text{argmax}} \;\; \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

Modeling, Inference, Learning

modeling: define score function

$$\mathrm{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y} \ \mathrm{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

- **Modeling**: How do we assign a score to an ($\boldsymbol{x}$,y) pair using parameters $\boldsymbol{w}$?

Modeling, Inference, Learning

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{y}{\text{argmax}} \ \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

- **Inference**: How do we efficiently search over the space of all labels?

Modeling, Inference, Learning

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{y}{\text{argmax}} \;\; \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

- **Learning**: How do we choose the weights ***w***?

Modeling, Inference, Learning

inference: solve argmax

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{y}{\text{argmax}} \; \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

- We will use this formulation throughout
  - even when output space is exponentially large or unbounded (e.g., machine translation)

Text Classification

- datasets
- classification
  - modeling
  - inference
  - learning

Binary Sentiment Classification

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \underset{y \in \{\text{positive},\text{negative}\}}{\text{argmax}} \quad \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

Binary Sentiment Classification

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y \in \{\text{positive}, \text{negative}\}} \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

a sequence of words
(a sentence or document)

sentiment label

Binary Sentiment Classification

modeling: define score function

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y \in \{\text{positive}, \text{negative}\}} \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

Linear Models

- parameters are arranged in a vector $\mathbf{w}$
- score function is linear in $\mathbf{w}$:

$$\mathrm{score}(\boldsymbol{x}, y, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y) = \sum_i w_i f_i(\boldsymbol{x}, y)$$

- $\mathbf{f}$: vector of feature functions

Linear Models for Binary Sentiment Classification

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y \in \{\text{positive},\text{negative}\}} \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

$$\text{classify}(\boldsymbol{x}, \mathbf{w}) = \operatorname*{argmax}_{y \in \{\text{positive},\text{negative}\}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y)$$

Linear Models for Binary Sentiment Classification

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y \in \{\text{positive}, \text{negative}\}} \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

$$\text{classify}(\boldsymbol{x}, \mathbf{w}) = \operatorname*{argmax}_{y \in \{\text{positive}, \text{negative}\}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y)$$

- How do we define $\mathbf{f}$?

Features for NLP

- NLP datasets include inputs and outputs
- features are usually not included
- you have to define your own features

Features for NLP

- NLP datasets include inputs and outputs
- features are usually not included
- you have to define your own features
- contrast this with UCI datasets, which include a fixed-length dense feature vector for every instance

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Features for NLP

- NLP datasets include inputs and outputs
- features are usually not included
- you have to define your own features
- contrast this with UCI datasets, which include a fixed-length dense feature vector for every instance
- in (traditional) NLP, features usually sparse

Defining Features

- this is a large part of NLP
- last 25 years: **feature engineering**
- last 4 years: **representation learning**

Defining Features

- this is a large part of NLP
- last 25 years: **feature engineering**
- last 4 years: **representation learning**

- In this course, we'll do both
- learning representations doesn't mean that we don't have to look at the data or the output!
- there's still plenty of engineering required in representation learning

Feature Engineering

- Often decried as "costly, hand-crafted, expensive, domain-specific", etc.
- But in practice, simple features typically give the bulk of the performance

- Let's get concrete: how should we define features for text classification?

Unigram Binary Features

- two example features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise

Unigram Binary Features

- two example features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1 \text{ if } S \text{ is true, } 0 \text{ otherwise}$

- we usually think in terms of feature templates

Unigram Binary Features

- two example features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise

- we usually think in terms of feature templates
- unigram binary feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

- to create features, this feature template is instantiated for particular labels and words

Feature Count Cutoffs

- problem: some features are extremely rare
- solution: only keep features that appear at least *k* times in the training data

Feature Count Cutoffs (Example)

- training dataset with 2 examples:

great movie	positive
not so great	negative

- and a single feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

Feature Count Cutoffs (Example)

- training dataset with 2 examples:

great movie	positive
not so great	negative

- and a single feature template:

$$f^{\text{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

- what features would be in the model with a feature count cutoff of 2? **2(a)**

To remind you, here's an example of an instantiation of the feature template above:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

Feature Count Cutoffs (Example)

- training dataset with 2 examples:

great movie	positive
not so great	negative

- and a single feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \mathit{word}]$$

- what features would be in the model with a feature count cutoff of 2? **2(a)**
  – none

Feature Count Cutoffs (Example)

- training dataset with 2 examples:

great movie	positive
not so great	negative

- and a single feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \mathit{word}]$$

- what features would be in the model with a feature count cutoff of 1? **2(b)**

- training dataset with 2 examples:

  great movie positive

  not so great negative

- and a single feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

- what features would be in the model with a feature count cutoff of 1? **2(b)**

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{movie}]$$
$$f_3(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{not}]$$
$$f_4(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{so}]$$
$$f_5(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

- training dataset with 2 examples:

  great movie positive

  not so great negative

- and a single feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

- what additional features would be in the model with a feature count cutoff of 0? **2(c)**

- training dataset with 2 examples:

great movie	positive
not so great	negative

- and a single feature template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

- what additional features would be in the model with a feature count cutoff of 0?

2(c)

$$f_6(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{movie}]$$
$$f_7(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{not}]$$
$$f_8(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{so}]$$

Higher-Order Binary Feature Templates

unigram binary template:

$$f^{\mathrm{u,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{word}]$$

bigram binary template:

$$f^{\mathrm{b,b}}(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ label}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{``word1 word2''}]$$

trigram binary template:

…

Unigram **Count** Features

- a "count" feature returns the count of a particular word in the text
- unigram count feature template:

$$f^{\mathrm{u,c}}(\boldsymbol{x}, y) = \begin{cases} \sum_{i=1}^{|\boldsymbol{x}|} \mathbb{I}[x_i = \ \mathit{word}], & \text{if } \mathbb{I}[y = \ \text{label}] \\ 0, & \text{otherwise} \end{cases}$$

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y) = \sum_i w_i f_i(\boldsymbol{x}, y)$$

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y) = \sum_i w_i f_i(\boldsymbol{x}, y)$$

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1 \text{ if } S \text{ is true, } 0 \text{ otherwise}$

- What do you expect the weights to be? **3**

$$w_1 > w_2? \qquad w_1 = w_2? \qquad w_1 < w_2?$$

Feature Engineering for Text Classification

$$\text{score}(\boldsymbol{x}, y, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y) = \sum_i w_i f_i(\boldsymbol{x}, y)$$

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

where $\mathbb{I}[S] = 1 \text{ if } S \text{ is true, } 0 \text{ otherwise}$

- What do you expect the weights to be? **3**

$\boxed{w_1 > w_2?}$ $\quad w_1 = w_2? \quad w_1 < w_2?$

Feature Engineering for Text Classification

- Two features:

$$f_1(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$
$$f_2(\boldsymbol{x}, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

- Let's say we set $w_1 > w_2$
- On sentences containing "***great***" in the Stanford Sentiment Treebank training data, this would get us an accuracy of 69%
- But "***great***'' only appears in 83/6911 examples

Feature Engineering for Text Classification

- Two features:

$$f_1(x, y) = \mathbb{I}[y = \text{ positive}] \wedge \mathbb{I}[x \text{ contains } \textit{great}]$$
$$f_2(x, y) = \mathbb{I}[y = \text{ negative}] \wedge \mathbb{I}[x \text{ contains } \textit{great}]$$

ambiguity: "***great***" may not mean positive sentiment

- On sentences containing "***great***" in the Stanford Sentiment Treebank training data, this would get us an accuracy of 69%
- But "***great***'' only appears in 83/6911 examples

variability: many other words can indicate positive sentiment

- Usually, ***great*** indicates positive sentiment:

  The most wondrous love story in years, it is a ***great*** *film.*

  A ***great*** *companion piece to other Napoleon films .*

- Usually, ***great*** indicates positive sentiment:

  The most wondrous love story in years, it is a ***great*** *film.*

  A ***great*** *companion piece to other Napoleon films .*

- Sometimes not. Why? List 3 reasons. $\boxed{4}$

- Usually, ***great*** indicates positive sentiment:

  *The most wondrous love story in years, it is a **great** film.*

  *A **great** companion piece to other Napoleon films .*

- Sometimes not. Why? List 3 reasons. **4**

  Negation: *It's not a **great** monster movie .*

  Different sense: *There's a **great** deal of corny dialogue and preposterous moments .*

  Multiple sentiments: *A **great** ensemble cast can't lift this heartfelt enterprise out of the familiar.*

- What about a feature like the following?

$$f_3(\boldsymbol{x}, y) = \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

- What do you expect its weight to be? $\boxed{\mathbf{5}}$

- What about a feature like the following?

$$f_3(\boldsymbol{x}, y) = \mathbb{I}[\boldsymbol{x} \text{ contains } \textit{great}]$$

- What do you expect its weight to be? $\boxed{\mathbf{5}}$
  - Doesn't matter.
  - Why?

$$\text{classify}(\boldsymbol{x}, \mathbf{w}) = \underset{y \in \{\text{positive}, \text{negative}\}}{\text{argmax}} \mathbf{w}^\top \mathbf{f}(\boldsymbol{x}, y)$$

  - a feature with the same value for all outputs will not affect the argmax

Text Classification

- datasets
- classification
  - modeling
  - inference
  - learning

inference: solve argmax

$$\mathrm{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y} \ \mathrm{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

- **Inference**: How do we efficiently search over the space of all labels?

Inference for Text Classification

$$\text{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y \in \{\text{positive}, \text{negative}\}} \text{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

- trivial (loop over labels)

Text Classification

- datasets
- classification
  - modeling
  - inference
  - learning

Modeling, Inference, Learning

inference: solve argmax

modeling: define score function

$$\mathrm{classify}(\boldsymbol{x}, \boldsymbol{w}) = \operatorname*{argmax}_{y} \ \mathrm{score}(\boldsymbol{x}, y, \boldsymbol{w})$$

learning: choose $\boldsymbol{w}$

- **Learning**: How should we choose values for the weights?

Experimental Practice

- in the beginning, we just had data

Experimental Practice

- in the beginning, we just had data
- first innovation: split into **train** and **test**
  - motivation: simulate conditions of applying system in practice

Experimental Practice

- in the beginning, we just had data
- first innovation: split into **train** and **test**
  - motivation: simulate conditions of applying system in practice
- but, there's a problem with this…

Experimental Practice

- in the beginning, we just had data
- first innovation: split into **train** and **test**
  - motivation: simulate conditions of applying system in practice
- but, there's a problem with this…
  - we need to explore and evaluate methodological choices
  - after multiple evaluations on **test**, it is no longer a simulation of real-world conditions

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on **train**, but this is slow and doesn't quite simulate real-world settings (why?)

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on **train**, but this is slow and doesn't quite simulate real-world settings (why?)
- second innovation: divide data into **train**, **test**, and a third set called development (**dev**) or validation (**val**)
  - use **dev**/**val** to evaluate choices
  - then, when ready to write the paper, evaluate the best model on **test**

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on **train**, but this is slow and doesn't quite simulate real-world settings (why?)
- second innovation: divide data into **train**, **test**, and a third set called development (**dev**) or validation (**val**)
  - use **dev**/**val** to evaluate choices
  - then, when ready to write the paper, evaluate the best model on **test**
- are we done yet? no! there's still a problem:

Experimental Practice

- we need to explore/evaluate methodological choices
- what should we do?
  - some use cross validation on **train**, but this is slow and doesn't quite simulate real-world settings (why?)
- second innovation: divide data into **train**, **test**, and a third set called development (**dev**) or validation (**val**)
  - use **dev**/**val** to evaluate choices
  - then, when ready to write the paper, evaluate the best model on **test**
- are we done yet? no! there's still a problem:
  - overfitting to **dev**/**val**

Experimental Practice

- best practice: split data into **train**, development (**dev**), development test (**devtest**), and **test**
  - train model on **train**, tune hyperparameters on **dev**, do preliminary testing on **devtest**, do final testing on **test** a single time when writing the paper
  - Even better to have even more test sets! **test1**, **test2**, etc.

- experimental credibility is a huge component of doing useful research
- when you publish a result, it had better be replicable without tuning anything on **test**

Don’t Cheat!

SECTIONS

The New York Times

SUBSCRIBE

LOG IN

TECHNOLOGY

Computer Scientists Are Astir After Baidu Team Is Barred From A.I. Competition

By JOHN MARKOFF JUNE 3, 2015

Email

Share

Tweet

Save

SAN FRANCISCO — A group of researchers at the Chinese web services company Baidu have been barred from participating in an international competition for artificial intelligence technology after organizers discovered that the Baidu scientists broke the contest’s rules.