Classical NLP

Naive Bayes, TF-IDF, Bag-of-Words, and Logistic Regression

Is this relevant?

PapersWithCode Leaderboard on IMDB Review Classification

RANK	MODEL	ACCURACY	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	NB-weighted-BON + dv-cosine	97.4	✓	Sentiment Classification Using Document Embeddings Trained with Cosine Similarity			2019
2	GraphStar	96.0	✓	Graph Star Net for Generalized Multi-Task Learning			2019
3	BERT large finetune UDA	95.8	✓	Unsupervised Data Augmentation for Consistency Training			2019
4	L MIXED	95.68	✓	Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function			2020
5	BERT large	95.49	✓	Unsupervised Data Augmentation for Consistency Training			2019

Is this relevant?

- Naive-Bayes weighted bag of n-grams trained with cosine similarity
- Link: https://www.aclweb.org/anthology/P19-2057/

- Background:
- Weighted neural bag-of-n-grams
- Link: https://www.aclweb.org/anthology/C16-1150.pdf

- Questions to ponder:
- How do you apply this on new test documents?
- Can you change the GenSim Doc2Vec (link) to implement this paper?

And….

Eric Wallace
@Eric_Wallace_

The state of NLP in 2019.

I'm talking with an amazing undergrad who has already published multiple papers on BERT-type things.

We are discussing deep into a new idea on pretraining.

Me: What would TFIDF do here, as a simple place to start?
Him: ....
Me: ....
Him: What's TFIDF?

10:40 am · 19 Dec 2019 · Twitter Web App

236 Retweets **34** Quote Tweets **1.3K** Likes

Categorization

- Given:
  - A **description of an instance**, $x \in X$, where X is the *instance language* or *instance space*.
  - A **fixed set of categories**: $C = \{c_1, c_2, \ldots c_n\}$
- Determine:
  - The **category of** x: $c(x) \in C$, where $c(x)$ is a categorization function whose domain is X and whose range is C.

Positive or negative movie review?

- unbelievably disappointing

- Full of zany characters and richly applied satire, and some great plot twists

- this is the greatest screwball comedy ever filmed

- It was pathetic. The worst part about it was the boxing scenes.

Text Classification

- Assigning documents to a fixed set of categories, *e.g.*
- Web pages
  - Yahoo-like classification
  - Assigning subject categories, topics, or genres
- Email messages
  - Spam filtering
  - Prioritizing
  - Folderizing
- Blogs/Letters/Books
  - Authorship identification
  - Age/gender identification
- Reviews/Social media
  - Language Identification
  - Sentiment analysis
  - ...

The bag of words representation

Y(

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

)=c

The bag of words representation

γ(

I **love** this movie! It's **sweet,** but with **satirical** humor. The dialogue is **great** and the adventure scenes are **fun…** It manages to be **whimsical** and **romantic** while **laughing** at the conventions of the fairy tale genre. I would **recommend** it to just about anyone. I've seen it **several** times, and I'm always **happy** to see it **again** whenever I have a friend who hasn't seen it yet.

)=c

The bag of words representation: using a subset of words

$Y($

```
x love xxxxxxxxxxxxxxxx sweet
xxxxxxx satirical xxxxxxxxxx
xxxxxxxxxxx great xxxxxxx
xxxxxxxxxxxxxxxxxxx fun xxxx
xxxxxxxxxxxxx whimsical xxxx
romantic xxxx laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxx recommend xxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxxx
xxxxx happy xxxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxx
```

$)=c$


The bag of words representation

Y(

great	2
love	2
recommend	1
laugh	1
happy	1
...	...

)=c

Bayes' Rule Applied to Documents and Classes

- For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Naïve Bayes Classifier (I)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier (II)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Naïve Bayes Classifier (IV)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

O(|X|^n•|C|) parameters

How often does this class occur?

Could only be estimated if a very, very large number of training examples was available.

We can just count the relative frequencies in a corpus

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \quad ., x_n \mid c)$$

- **Bag of Words assumption**: Assume position doesn't matter

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \quad ., x_n \mid c)$$

- **Bag of Words assumption**: Assume position doesn't matter
- **Conditional Independence**: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c.

$$P(x_1, \quad ., x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Multinomial Naïve Bayes Classifier

$$c_{MAP} = \operatorname*{argmax}_{c \in C} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \operatorname*{argmax}_{c \in C} P(c_j) \prod_{x \in X} P(x \mid c)$$

Parameter estimation

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears among all words in documents of topic c_j

- Create mega-document for topic *j* by concatenating all docs in this topic
  – Use frequency of *w* in mega-document

Problem with Maximum Likelihood

Sec.13.3

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up)***?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Naïve Bayes Time Complexity

- **Training Time**: $O(|D|L_d + |C||V|))$
  where L_d is the average length of a document in D.
  - Assumes V and all D_i , n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$
- **Test Time**: $O(|C|\ L_t)$
  where L_t is the average length of a test document.
- Very efficient overall, linearly proportional to the time needed to just read in all the data.

Probabilities: Important Detail!

- We are multiplying lots of small numbers
  Danger of underflow!
  - $0.5^{57} = 7$ E -18
- Solution? Use logs and add!
  - $p_1 * p_2 = e^{\log(p1)+\log(p2)}$
  - Always keep in log form

Advantages

- Simple to implement
  - No numerical optimization, matrix algebra, etc
- Efficient to train and use
  - Easy to update with new data
  - Fast to apply
- Binary/multi-class
- Good in domains with many equally important features
  - Decision Trees suffer from fragmentation in such cases – especially if little data
- Comparatively good effectiveness with small training sets
- A good dependable baseline for text classification
  - But we will see other classifiers that give better accuracy

Disadvantages

- Independence assumption wrong
  - Absurd estimates of class probabilities
    - Output probabilities close to 0 or 1
  - Thresholds must be tuned; not set analytically

- Generative model
  - Generally lower effectiveness than discriminative techniques

Generative vs Discriminative

- Generative classifiers
- Assume some functional form for P(Y), P(X|Y)
- Estimate parameters of P(X|Y), P(Y) from training data
- Use Bayes rule to calculate P(Y |X)

- Discriminative Classifiers
- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

Generative vs Discriminative

- Generative classifiers:
- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models (HMM)

- Discriminative Classifiers:
- Logistic regression
- Scalar Vector Machine
- Traditional neural networks
- Nearest neighbour

Exponential Models (log-linear, maxent, Logistic, Gibbs)

- Model: use the scores as probabilities:

$$p(y|x;w) = \frac{\text{exp}\,(w \cdot \phi(x,y))}{\sum_{y'} \text{exp}\,(w \cdot \phi(x,y'))} \begin{array}{l} \longleftarrow \text{Make positive} \\ \longleftarrow \text{Normalize} \end{array}$$

- Learning: maximize the (log) conditional likelihood of training data $\{(x_i, y_i)\}_{i=1}^n$

$$L(w) = \sum_{i=1}^{n} \log p(y_i|x_i;w) \qquad w^* = \arg\max_w L(w)$$

- Prediction: output argmax_y p(y|x;w)

Logistic Regression - Brief Intro

- $p(y=1|x,w) = \sigma(w^T x) = 1/(1+\exp(-w^T x)) = \exp(w^T x)/(1+\exp(w^T x))$
- $p(y=0|x,w) = 1/(1+\exp(w^T x))$
- A special case of exponential family of models in which $\boldsymbol{\Phi(x,y) = yx}$
- Commonly used discriminative model on standard datasets

Feature-Based Linear Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
  - Given this model form, we will choose parameters $\{w_i\}$ that *maximize the conditional likelihood* of the data according to this model.
  - We construct not only classifications, but probability distributions over classifications.
    - There are other (good!) ways of discriminating classes – SVMs, boosting, even perceptrons – but these methods are not as trivial to interpret as distributions over classes.

Derivative of Log-linear Model

$$p(y|x;w) = \frac{\exp\left(w \cdot \phi(x,y)\right)}{\sum_{y'} \exp\left(w \cdot \phi(x,y')\right)}$$

- Unfortunately, argmax_w L(w) doesn't have a close formed solution
- We will have to differentiate and use gradient ascent

$$L(w) = \sum_{i=1}^{n} \log p(y_i|x_i;w)$$

$$L(w) = \sum_{i=1}^{n} \left(w \cdot \varphi(x_i,y_i) - \log \sum_{y} \exp(w \cdot \varphi(x_i,y)) \right)$$

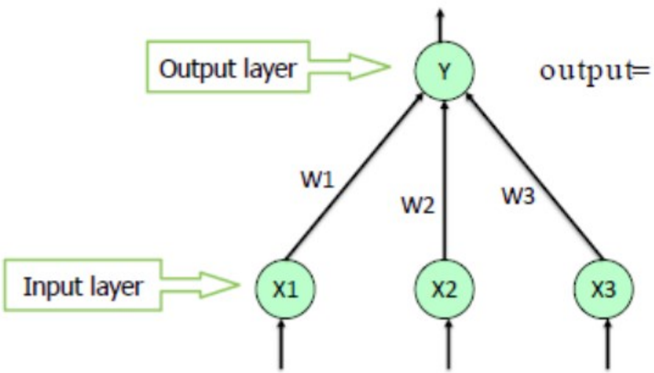
$$\frac{\partial L(w)}{\partial w_{jk}} = \sum_{i=1}^{n} \left(\varphi_{jk}(x_i,y_i) - \mathrm{p}(\mathrm{k}|x_i;\mathrm{w})\varphi_{jk}(x_i,\mathrm{k}) \right)$$

Total count of feature j in correct candidates

Expected count of feature j in predicted candidates

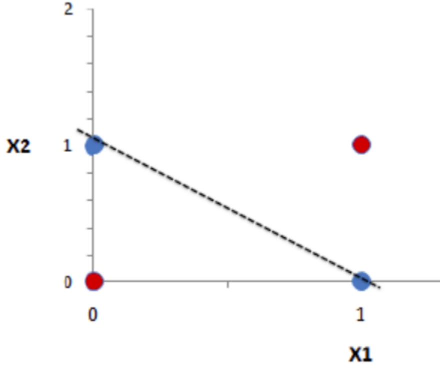
Single-Layer Perceptron

Single Layer Perceptron



$$\text{output} = \begin{cases} 1 & \text{if } \sum w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

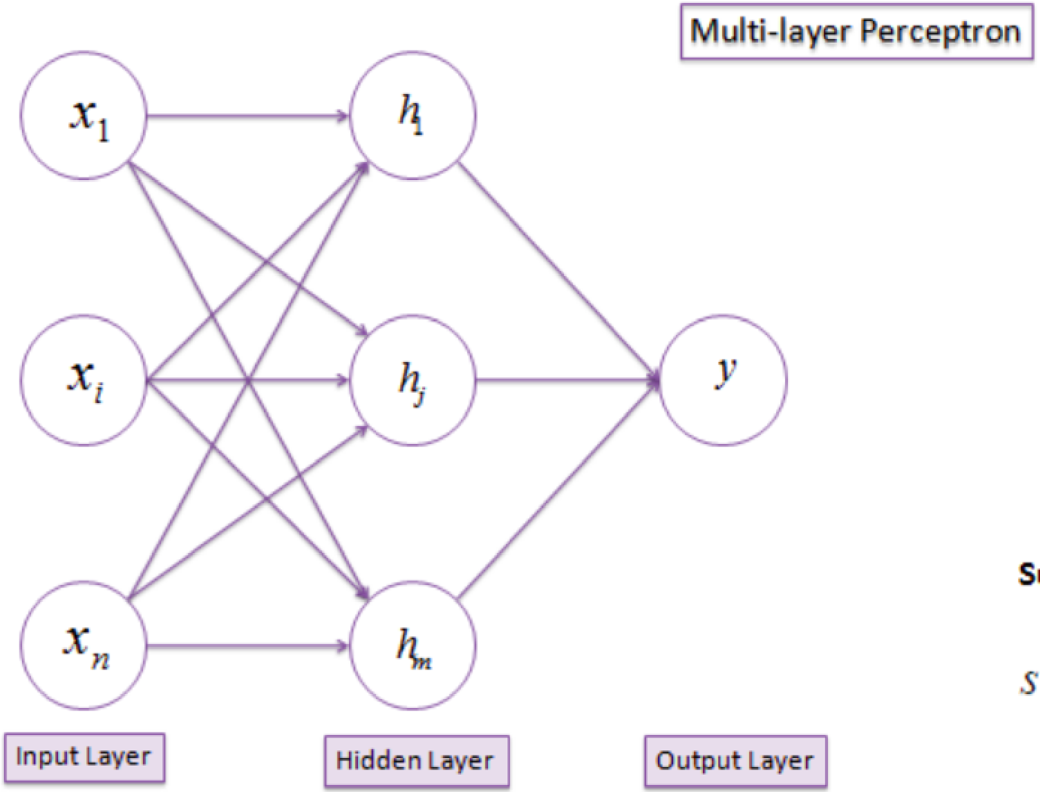
Cannot solve even a XOR problem !!!

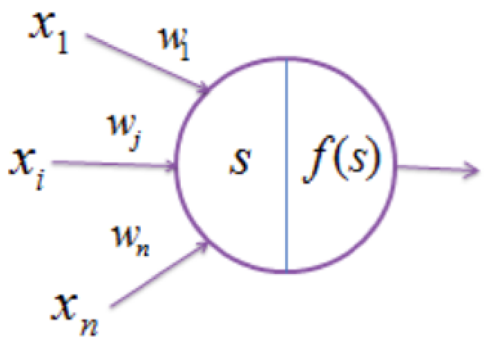


XOR

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Multi-Layer Perceptron (MLP)





Summation

$$s = \sum w \cdot x$$

Transformation

$$f(s) = \frac{1}{1+e^{-s}}$$

MLP

- Idea is to approximate more complicated underlying probability distributions in the dataset
- Hyperparameters are - no. of hidden layers, no. of neurons in each hidden layer, choice of activation function (ReLu, sigmoid, GeLu etc.)
- Practically single-hidden layer neural networks used as basic units in many applications (BiLSTM, wordvec etc.)

Document Features

- Binary Features:
- Presence or absence of a word in the document
- D1 = <0, 0, 1, …, 1, 0>

- Count-based Features:
- Number of times a word appears in the document
- D1 = <0, 0, 4, …, 7, 0>

Features

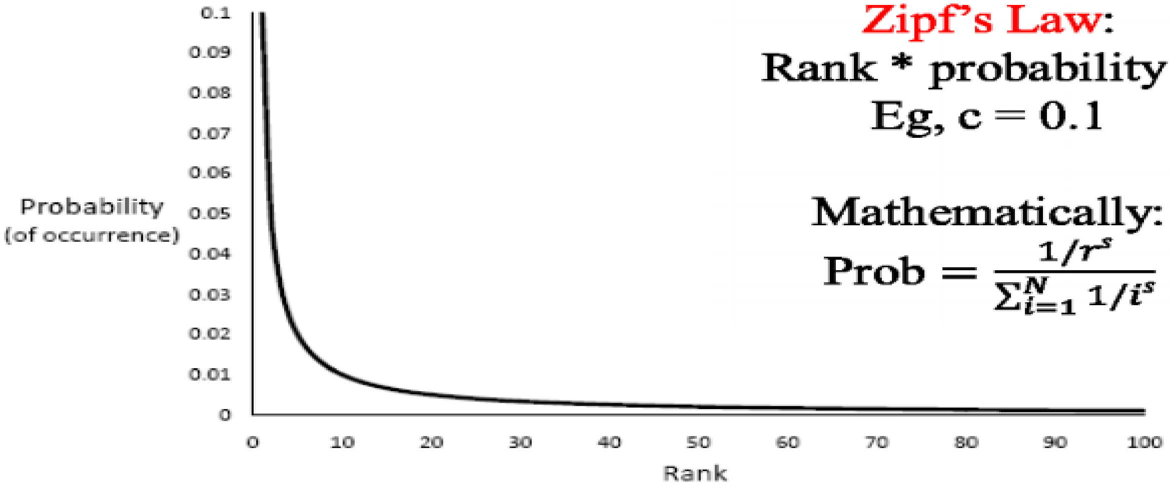
Sec. 15.3.2

- Domain-specific features and weights: *very* important in real performance

- Upweighting: Counting a word as if it occurred twice:
  – title words (Cohen & Singer 1996)
  – first sentence of each paragraph (Murata, 1999)
  – In sentences that contain title words (Ko *et al*, 2002)

Properties of Text

- Word frequencies - skewed distribution
- `The' and `of' account for 10% of all words
- Six most common words account for 40%



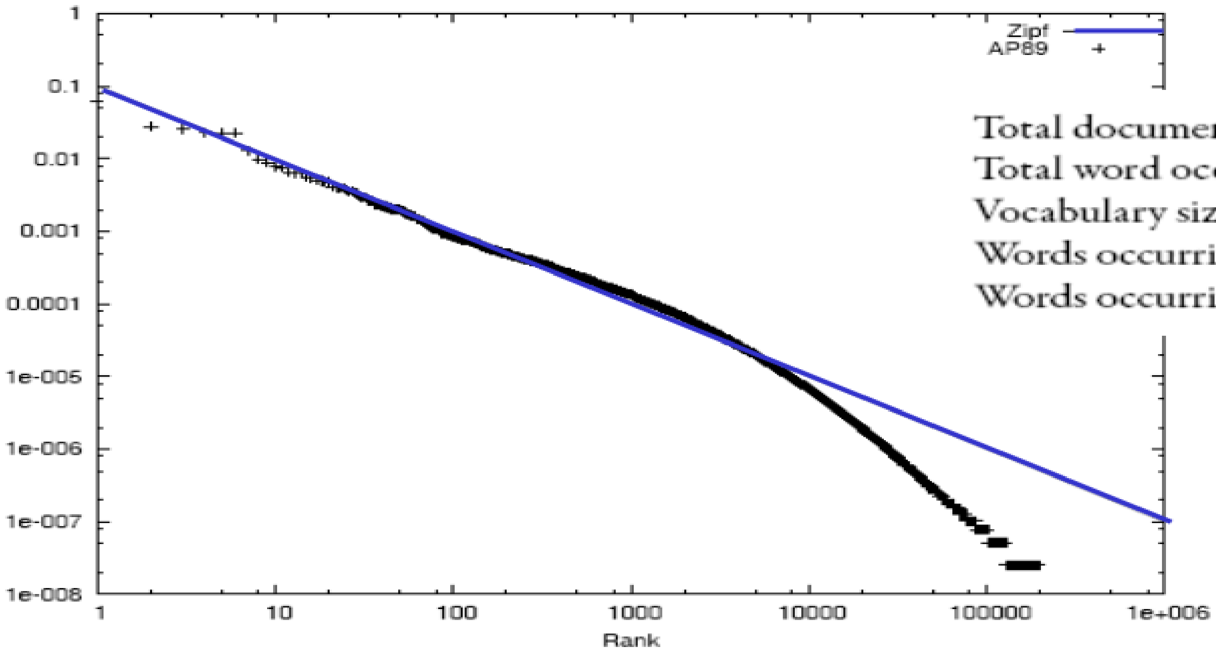
Zipf's Law:
Rank * probability = c
Eg, c = 0.1

Mathematically:

$$\text{Prob} = \frac{1/r^s}{\sum_{i=1}^{N} 1/i^s}$$

From [Croft, Metzler & Strohman 2010]

Associate Press Corpus `AP89'



Total documents	84,678
Total word occurrences	39,749,179
Vocabulary size	198,763
Words occurring > 1000 times	4,169
Words occurring once	70,064

From [Croft, Metzler & Strohman 2010]

Middle Ground

- Very common words → bad features
  - Language-based stop list:

    words that bear little meaning

    20-500 words

    http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words
  - Subject-dependent stop lists

- Very rare words *also* bad features

  Drop words appearing less than k times / corpus

Word Frequency

- Which word is more indicative of document similarity?
  - 'book,' or 'Rumplestiltskin'?
  - Need to consider "document frequency"--- how frequently the word appears in doc collection.

- Which doc is a better match for the query "Kangaroo"?
  - One with a single mention of Kangaroos… or a doc that mentions it 10 times?
  - Need to consider "term frequency"--- how many times the word appears in the current document.

TF x IDF

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

$T_k = term\ k\ in\ document\ D_i$

$tf_{ik} = frequency\ of\ term\ T_k\ in\ document\ D_i$

$idf_k = inverse\ document\ frequency\ of\ term\ T_k\ in\ C$

$$idf_k = \log\left(\frac{N}{n_k}\right)$$

$N = total\ number\ of\ documents\ in\ the\ collection\ C$

$n_k = the\ number\ of\ documents\ in\ C\ that\ contain\ T_k$

Inverse Document Frequency

- IDF provides high values for rare words and low values for common words

$$\log\left(\frac{10000}{10000}\right) = 0$$

$$\log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698$$

$$\log\left(\frac{10000}{1}\right) = 4$$

- Add 1 to avoid 0.

Tf-idf representation of documents

	w_1	w_2	……	w_j	……	w_N
doc_1						
….						
doc_i				$tf_{ji}*idf_j$		
….						
doc_D						

$tf_{ji} = \log_2(1+w_{ji})$

$idf_j = 1+\log_2(D/D_j)$, where D_j is no. of documents containing term w_j

TF-IDF normalization

- Normalize the term weights
  - so longer docs not given more weight (fairness)
  - force all values to fall within a certain range: [0, 1]

$$w_{ik} = \frac{tf_{ik}(1+\log(N/n_k))}{\sqrt{\sum_{k=1}^{t}(tf_{ik})^2[1+\log(N/n_k)]^2}}$$

N-Gram features

- An n-gram is a subsequence of n items from a given sequence.
- Unigram: n-gram of size 1
- Bigram: n-gram of size 2
- Trigram: n-gram of size 3

- Input: “the dog smelled like a skunk”
- Bigrams:
  # the, the dog, dog smelled, smelled like, like a, a skunk, skunk#
- Trigrams:
  # the dog, the dog smelled, dog smelled like, smelled like a, like a skunk and a skunk #.

Diving Deeper into Feature Engineering

Issues in document representation

Cooper's concordance of Wordsworth was published in 1911. The applications of full-text retrieval are legion: they include résumé scanning, litigation support and searching published journals on-line.

- *Cooper's vs. Cooper vs. Coopers.*
- *Full-text vs. full text vs. {full, text} vs. fulltext.*
- *résumé vs. resume.*

Punctuation

- *Ne'er*: use language-specific, handcrafted "locale" to normalize.
- *State-of-the-art*: break up hyphenated sequence.
- *U.S.A.* vs. *USA*
- *a.out*

Possible Feature Ideas

- Look at capitalization (may indicated a proper noun)

- Look for commonly occurring sequences
  - E.g. New York, New York City
  - Limit to 2-3 consecutive words
  - Keep all that meet minimum threshold (e.g. occur at least 5 or 10 times in corpus)

Case folding

- Reduce all letters to lower case
- Exception: upper case in mid-sentence
  - *e.g., **General Motors***
  - ***Fed*** vs. ***fed***
  - ***SAIL*** vs. ***sail***

Thesauri and Soundex

- Handle synonyms and spelling variations
  - Hand-constructed equivalence classes
    - e.g., ***car*** = ***automobile***

Spell Correction

- Look for all words within (say) edit distance 3 (Insert/Delete/Replace) at query time
  - e.g., ***arfiticial inteligence***
- Spell correction is expensive and slows the processing significantly
  - Invoke only when index returns zero matches?

Lemmatization

- Reduce inflectional/variant forms to base form
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*

the boy's cars are different colors

$\rightarrow$

the boy car be different color

Scikit example

```
>>> from sklearn.datasets import fetch_20newsgroups
>>> twenty_train = fetch_20newsgroups(subset='train',
...     categories=categories, shuffle=True, random_state=42)
```

Loading data

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> count_vect = CountVectorizer()
>>> X_train_counts = count_vect.fit_transform(twenty_train.data)
>>> X_train_counts.shape
(2257, 35788)
```

```
>>> from sklearn.feature_extraction.text import TfidfTransformer
>>> tf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)
>>> X_train_tf = tf_transformer.transform(X_train_counts)
>>> X_train_tf.shape
(2257, 35788)
```

Extracting feature vector

```
>>> from sklearn.naive_bayes import MultinomialNB
>>> clf = MultinomialNB().fit(X_train_tfidf, twenty_train.target)
```

```
>>> from sklearn.linear_model import LogisticRegression
```

```
>>> from sklearn.svm import SVC
```

```
>>> from sklearn.neural_network import MLPClassifier
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(5, 2), random_state=1)
...
```

fitting model
on train set

Prediction
on test set

```
>>> import numpy as np
>>> twenty_test = fetch_20newsgroups(subset='test',
...     categories=categories, shuffle=True, random_state=42)
>>> docs_test = twenty_test.data
>>> predicted = text_clf.predict(docs_test)
>>> np.mean(predicted == twenty_test.target)
0.8348...
```

Next Class

- "CBOW - Continuous Bag of Words"
- Models for learning feature vectors

- "Word2Vec and Glove"
- Models for learning word vectors

References

- http://www.cse.iitd.ac.in/~mausam/courses/col772/spring2019/lectures/06-loglinear.pdf
- http://www.cse.iitd.ac.in/~mausam/courses/col772/spring2019/lectures/04-textcat.pdf
- http://spring2015.cs-114.org/wp-content/uploads/2016/01/NgramModels.pdf

Multi-class Problems

Evaluation: Classic Reuters-21578 Data Set

Sec. 15.2.4

- Most (over)used data set, 21,578 docs (each 90 types, 200 tokens)
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document (with at least one category) has 1.24 classes
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Reuters Text Categorization data set (**Reuters-21578**) document

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

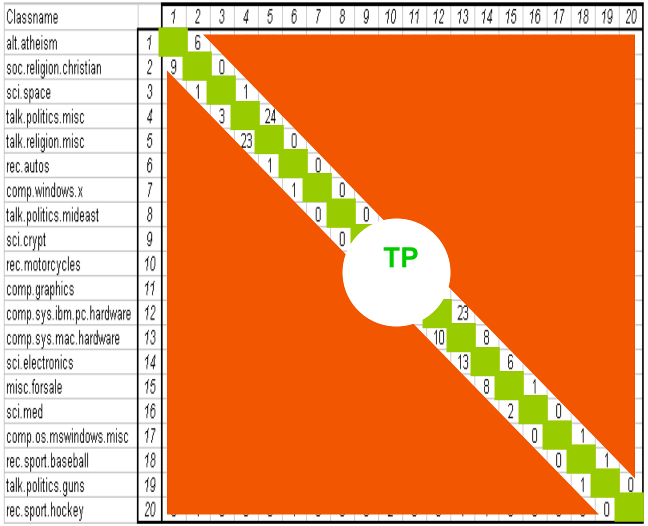
&#3;</BODY></TEXT></REUTERS>

Precision & Recall

Two class situation

	Predicted	
Actual		
	“P”	**“N”**
P	**TP**	**FN**
N	**FP**	**TN**

Multi-class situation:



Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
F-measure = 2pr/(p+r)

Micro-- vs. Macro--Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?

- Macroaveraging
  - Compute performance for each class, then average.

- Microaveraging
  - Collect decisions for all classes, compute contingency table, evaluate

Precision & Recall

Multi-class Multi-label situation:

Classname		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
alt.atheism	1		6	1	3	32	1	1	2	1	2	0	0	0	0	0	0	0	0	0	0
soc.religion.christian	2	9		0	1	6	0	0	1	0	0	0	0	0	1	2	2	0	0	0	1
sci.space	3	3	1		1	0	1	2	0	1	1	9	0	0	1	2	3	0	0	1	1
talk.politics.misc	4	2	0	3		24	3	0	17	3	0	0	0	0	0	0	1	0	1	33	0
talk.religion.misc	5	88	36	2	23		0	1	0	0	0	0	0	0	0	0	2	0	1	15	0
rec.autos	6	0	0	0	3	1		0	0	0	7	1	2	1	6	4	1	0	0	2	0
comp.windows.x	7	1	1	2	1	0	1		0	2	2	30	5	3	1	1	2	1	1	0	0
talk.politics.mideast	8	0	3	1	18	0	0	0		0	1	0	0	0	0	0	0	0	1	1	0
sci.crypt	9	1	0	1	2	1	0	3	0		0	3	0	1	0	0	1	0	0	3	0
rec.motorcycles	10	0	0	0	1	0	4	1	0	0		1	2	0	1	2	1	0	0	1	0
comp.graphics	11	0	1	2	1	1	0	10	1	2	0		23	7	3	3	3	0	0	0	0
comp.sys.ibm.pc.hardware	12	0	0	0	0	0	2	7	0	1	0	5		23	12	3	1	3	0	0	0
comp.sys.mac.hardware	13	0	0	1	1	0	2	1	0	0	0	7	10		8	9	1	0	0	0	0
sci.electronics	14	1	0	1	0	1	5	2	0	2	0	7	13	13		6	3	0	1	0	0
misc.forsale	15	0	1	4	2	0	12	1	0	0	4	1	19	10	8		1	0	1	1	2
sci.med	16	0	1	5	0	1	1	0	0	0	1	2	0	2	7	2		0	1	1	1
comp.os.mswindows.misc	17	1	0	2	0	1	1	58	1	3	0	38	71	17	3	6	0		1	0	0
rec.sport.baseball	18	2	1	1	0	0	0	0	0	0	0	4	0	0	0	1	1	0		1	7
talk.politics.guns	19	0	0	0	9	5	1	0	0	1	0	0	0	0	1	0	0	1	1		0
rec.sport.hockey	20	0	1	0	0	0	1	0	0	0	2	0	0	1	1	0	0	0	3	0	

Aggregate

Average Macro Precision = $\Sigma p_i/N$

Average Macro Recall = $\Sigma r_i/N$

Average Macro F-measure = $2p_M r_M/(p_M+r_M)$

Average Micro Precision = $\Sigma TP_i/ \Sigma_i Col_i$

Average Micro Recall = $\Sigma TP_i/ \Sigma_i Row_i$

Average Micro F-measure = $2p_\mu r_\mu /(p_\mu+r_\mu)$

Precision(class i) = $TP_i/(TP_i+FP_i)$

Recall(class i) = $TP_i/(TP_i+FN_i)$

F-measure(class i) = $2p_i r_i/(p_i+r_i)$

Precision(class 1) = $251/(Column_1)$

Recall(class 1) = $251/(Row_1)$

F-measure(class 1)) = $2p_i r_i/(p_i+r_i)$

Precision & Recall

Multi-class situation:

Classname		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
alt.atheism	1		6	1	3	32	1	1	2	1	2	0	0	0	0	0	0	0	0	0	0
soc.religion.christian	2	9		0	1	6	0	0	1	0	0	0	0	0	1	2	2	0	0	0	1
sci.space	3	3	1		1	0	1	2	0	1	1	9	0	0	1	2	3	0	0	1	1
talk.politics.misc	4	2	0	3		24	3	0	17	3	0	0	0	0	0	0	1	0	1	33	0
talk.religion.misc	5	88	36	2	23		0	1	0	0	0	0	0	0	0	0	2	0	1	15	0
rec.autos	6	0	0	0	3	1		0	0	0	7	1	2	1	6	4	1	0	0	2	0
comp.windows.x	7	1	1	2	1	0	1		0	2	2	30	5	3	1	1	2	1	1	0	0
talk.politics.mideast	8	0	3	1	18	0	0	0		0	1	0	0	0	0	0	0	0	1	1	0
sci.crypt	9	1	0	1	2	1	0	3	0		0	3	0	1	0	0	1	0	0	3	0
rec.motorcycles	10	0	0	0	1	0	4	1	0	0		1	2	0	1	2	1	0	0	1	0
comp.graphics	11	0	1	2	1	1	0	10	1	2	0		23	7	3	3	3	0	0	0	0
comp.sys.ibm.pc.hardware	12	0	0	0	0	0	2	7	0	1	0	5		23	12	3	1	3	0	0	0
comp.sys.mac.hardware	13	0	0	1	1	0	2	1	0	0	0	7	10		8	9	1	0	0	0	0
sci.electronics	14	1	0	1	0	1	5	2	0	2	0	7	13	13		6	3	0	1	0	0
misc.forsale	15	0	1	4	2	0	12	1	0	0	4	1	19	10	8		1	0	1	1	2
sci.med	16	0	1	5	0	1	1	0	0	0	1	2	0	2	7	2		0	1	1	1
comp.os.mswindows.misc	17	1	0	2	0	1	1	58	1	3	0	38	71	17	3	6	0		1	0	0
rec.sport.baseball	18	2	1	1	0	0	0	0	0	0	0	4	0	0	0	1	1	0		1	7
talk.politics.guns	19	0	0	0	9	5	1	0	0	1	0	0	0	0	1	0	0	1	1		0
rec.sport.hockey	20	0	1	0	0	0	1	0	0	0	2	0	0	1	1	0	0	0	3	0	

Missed predictions

Classifier hallucinations

Aggregate

Average Macro Precision = $\Sigma p_i/N$

Average Macro Recall = $\Sigma r_i/N$

Average Macro F-measure = $2p_M r_M/(p_M+r_M)$

Average Micro Precision = $\Sigma TP_i / \Sigma_i Col_i$

Average Micro Recall = $\Sigma TP_i / \Sigma_i Row_i$

Average Micro F-measure = $2p_\mu r_\mu /(p_\mu+r_\mu)$

Aren't μ prec and μ recall the same?

Precision(class i) = $TP_i/(TP_i+FP_i)$

Recall(class i) = $TP_i/(TP_i+FN_i)$

F-measure(class i) = $2p_i r_i/(p_i+r_i)$

Precision(class 1) = $251/(\text{Column}_1)$

Recall(class 1) = $251/(\text{Row}_1)$

F-measure(class 1)) = $2p_i r_i/(p_i+r_i)$

Multi-Class Classification

- What?
  - Converting a k-class problem to a binary problem.
- Why?
  - For some ML algorithms, a direct extension to the multiclass case may be problematic.
- How?
  - Many methods

Methods

- One-vs-all
- All-pairs
- Error-correcting Output Codes (ECOC)
- …

One-vs-all

- Create many 1 vs other classifiers

  Classes = City, County, Country

  - Classifier 1 = {City} {County, Country}, Classifier 2 = {County} {City, Country}, Classifier 3 = {Country} {City, County}

- Training time:
  - For each class c_m, train a classifier $cl_m(x)$
    - replace (x,y) with

      $(x, 1)$ if $y = c_m$

      $(x, -1)$ if $y \neq c_m$

An example: training

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

for c1-vs-all:

x1 1 …

x2 -1 …

x3 1 …

x4 -1 …

for c2-vs-all:

x1 -1

x2 1 …

x3 -1 …

x4 -1 …

for c3-vs-all:

x1 -1…

x2 -1…

x3 -1 …

x4 1 …

One-vs-all (cont)

- Testing time: given a new example x
  - Run each of the k classifiers on x
  - Choose the class c_m with the highest confidence score $cl_m(x)$:

$$c^* = \arg\max_m cl_m(x)$$

An example: testing

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

➔ three classifiers

Test data:
x ?? f1 v1 …

for c1-vs-all:
x ?? 1 0.7 -1 0.3

for c2-vs-all
x ?? 1 0.2 -1 0.8

for c3-vs-all
x ?? 1 0.6 -1 0.4

=> what's the system prediction for x?

All-pairs (All-vs-All (AVA))

- Idea:
  - For each pair of classes build a classifier
  - {City vs. County}, {City vs Country}, {County vs. Country}
  - C_k^2 classifiers: one classifier for each class pair.
- Training:
  - For each pair (c_m, c_n) of classes, train a classifier cl_{mn}
    - replace a training instance (x,y) with (x, 1) if $y = c_{m,}$ (x, -1) if $y = c_n$

      otherwise ignore the instance

An example: training

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

for c1-vs-c2:
x1 1 …
x2 -1 …
x3 1 …

for c2-vs-c3:
x2 1 …
x4 -1 …

for c1-vs-c3:
x1 1…
x3 1 …
x4 -1 …

All-pairs (cont)

- Testing time: given a new example x
  - Run each of the C_k^2 classifiers on x
  - Max-win strategy: Choose the class c_m that wins the most pairwise comparisons:
  - Other coupling models have been proposed: e.g., (Hastie and Tibshirani, 1998)

An example: testing

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

➔ three classifiers

Test data:

x ?? f1 v1 …

for c1-vs-c2:

x ?? 1 0.7 -1 0.3

for c2-vs-c3

x ?? 1 0.2 -1 0.8

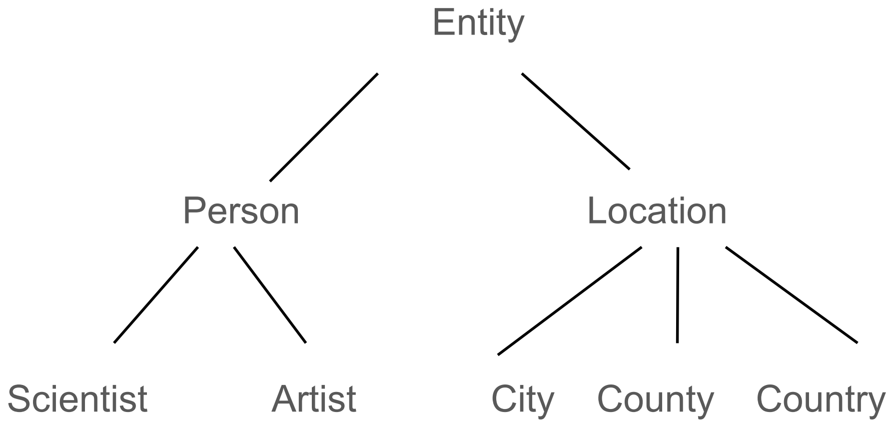
for c1-vs-c3

x ?? 1 0.6 -1 0.4

=> what's the system prediction for x?

Hierarchical Categorization

- Pick the category with max probability
- Create many OVA/AVA classifiers
- Use a hierarchical approach (wherever hierarchy available)



Summary

- Different methods:
    - Direct multiclass, if possible
    - One-vs-all (a.k.a. one-per-class): k-classifiers
    - All-pairs: C_k^2 classifiers
    - Hierarchical classification (logC classifiers)

- Some studies report that All-pairs works better than one-vs-all.