Text Classification

Natural Language Processing

Master in Business Analytics and Big Data

acastellanos@faculty.ie.edu

Examples

Quora Digest <digest-noreply@quora.com> Cancelar suscripcion
para mí

1 feb. (hace 3 días)

Quora

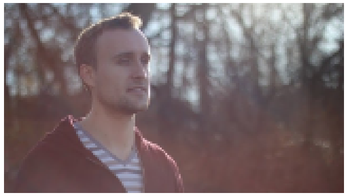
Angel's Digest

TOP STORIES FOR YOU

How can I be a great computer scientist and what courses should I take to move from beginner level to an advanced level?

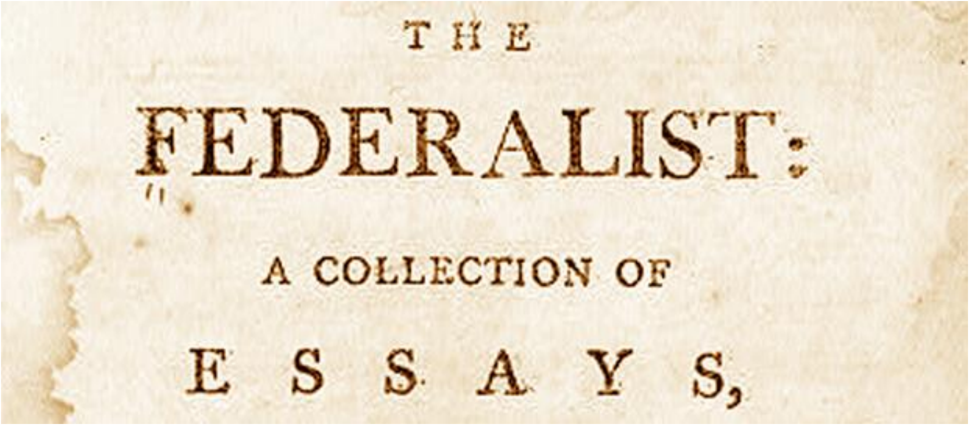
Quincy Larson, teacher at FreeCodeCamp-com
Written May 6, 2015

All of them. This guy just sat down at his desk and cranked through MIT's entire undergraduate Computer Science curriculum in exactly one year, passing all the exams. It's ... Read More »



Examples

Authorship



Examples

Positive or Negative 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.

Examples

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- Sarcasm Detection
- Fake News Detection

Methodologies: Hand-coded Rules

- **Rules** based on combinations of tokens or other features
  - Widely used in industry
  - Codify **domain knowledge**

```
black-list OR (“dollars” AND “have been selected”)
```

- **Accuracy can be high**
  - If rules carefully refined by expert

- **Very Expensive** to build and maintain these rules

Methodologies: Supervised Machine Learning

- Input:
  - a document d
  - a fixed set of classes $C = \{c_1\ c_2, \ldots, c_J\}$
  - A training set of m hand-labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$
- Output:
  - a learned classifier $\gamma : d \rightarrow c$

Naïve Bayes

Likelihood
How probable is the evidence given that our hypothesis is true?

Prior
How probable was our hypothesis before observing the evidence?

$$P(C \mid d) = \frac{P(d \mid C)\, P(C)}{P(d)}$$

Posterior
How probable is our hypothesis given the observed evidence?

Marginal
How probable is the new evidence under all possible hypotheses?

Naïve Bayes

- "Naïve" comes from "too simple to be true"
- Suppose we want to build a text classifier based on Naïve Bayes:
  - The document is made of a series of words $(x_1, \dots, x_n)$
  - The classifier must be able to distinguish between two classes C_1 and C_2, for a given document d.
  - The text will be assigned to the class for which:

$$MAP = \max(P(C \mid x_1, \dots, x_n), P(C \mid x_1, \dots, x_n))$$

This is called the "**Maximum A Posteriori**" (MAP) probability.

Naïve Bayes

Given a document and the words present on it $(x_1, \ldots, x_n)$
we can use Bayes to compute the probability that it belongs to class C

$$P(C|x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n|C)\ P(C)}{P(x_1, \ldots, x_n)}$$

Naïve Bayes

$$P(C|x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n|C)\ \mathbf{P(C)}}{P(x_1, \ldots, x_n)}$$

C_1

C_2

$P(C_1) = 5/8 = 0.625$

$P(C_2) = 3/8 = 0.375$

Naïve Bayes

$$P(C|x_1,\dots,x_n) = \frac{\boldsymbol{P(x_1,\dots,x_n|C)}\, P(C)}{P(x_1,\dots,x_n)}$$

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

Naïve Bayes

```
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.
```

Naïve Bayes

```
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.
```

Naïve Bayes

x **love** xxxxxxxxxxxxxxxx **sweet** xxxxxxx **satirical**
xxxxxxxxxx xxxxxxxxxxx **great** xxxxxxx
xxxxxxxxxxxxxxxxxxx **fun** xxxx xxxxxxxxxxxxx **whimsical**
xxxx **romantic** xxxx **laughing**
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
recommend xxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
several xxxxxxxxxxxxxxxxx xxxxx **happy** xxxxxxxxx
again xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Naïve Bayes

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

Naïve Bayes

$$P(C|x_1, \ldots, x_n) = \frac{\boldsymbol{P(x_1, \ldots, x_n|C)}\, P(C)}{P(x_1, \ldots, x_n)}$$

- **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.

$$P(x_1, \ldots, x_n|C) = P(x_1|C) \cdot P(x_2|C) \cdot \ldots \cdot P(x_n|C)$$

Naïve Bayes

$P(x_1|C) \cdot \ldots \cdot P(x_n|C)$ This is a **frequency count problem**, very easy to solve.

- To decide which class a text belongs to, we must compute:

$$\max\Big(\underbrace{P(C_1)P(x_1\,|\,C_1)\cdot\ldots\cdot P(x_n\,|\,C_1)}_{\text{Probability of these events belong to class } k_1},\ \underbrace{P(C_2)P(x_1\,|\,C_2)\cdot\ldots\cdot P(x_n\,|\,C_2)}_{\text{Probability of these events belong to class } k_2}\Big)$$

- And that’s it! No training, etc., just compute probabilities and classify

Naïve Bayes

- **Underflow Prevention: log space**
  - Multiplying lots of probabilities can result in floating-point underflow.
  - Since $\log(xy) = \log(x) + \log(y)$
    - Better to sum logs of probabilities instead of multiplying them
  - Class with highest un-normalized log probability score is still most probable.

$$P(C_1) \sum_{i \in positions} P(x_i \mid C_j) = \log(P(C_j) + \sum_{i \in positions} \log(P(x_i \mid C_j))$$

  - Model is now just max of sum of weights

Naïve Bayes

- 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!

$$P(C_1)P(x_1 \mid C_1) \cdot \ldots \cdot P(x_n \mid C_1)$$

Naïve Bayes

- Laplace Smoothing

Add-one approach to avoid zeroed probabilities

$$P(x_i|C_j) = \frac{count(x_i, C_j)}{\sum count(x, C_j)} \sim \frac{count(x_i, C_j) + 1}{\sum count(x, C_j) + |X|}$$

Naïve Bayes

$$P(C) = \frac{N_C}{N}$$

$$P(x|C) = \frac{count(w,c) + 1}{count(c) + |X|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	ch
	2	Chinese Chinese Shanghai	ch
	3	Chinese Macao	ch
	4	Tokyo Japan Chinese	jp
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$$P(ch) = 3/4$$

$$P(jp) = 1/4$$

Conditional Probabilities:

P(Chinese|ch) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|ch) = (0+1) / (8+6) = 1/14

P(Japan|ch) = (0+1) / (8+6) = 1/14

P(Chinese|jp) = (1+1) / (3+6) = 2/9

P(Tokyo|jp) = (1+1) / (3+6) = 2/9

P(Japan|jp) = (1+1) / (3+6) = 2/9

Choosing a class:

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$

$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

MaxEnt Classifiers

- All models are wrong but some are useful
  - But Naïve Bayes is **too wrong for text**
- **MaxEnt do not assume independence**
  - *"Assume nothing about your probability distribution other than what you have observed"*
- Select the model with the **largest entropy**
  - It is assumed to be the model that best represent the data: the **principle of maximum entropy**
  - **Most Uniform model = Less assumptions**

MaxEnt Classifiers: Example

- We want to classify documents into 4 classes: (economics, sports, politics, art)

- **VSM**: document = vector of words.

$$D = \{w_1, \dots, w_m\}$$

- We want to construct a **probability distribution** that represents the documents

MaxEnt Classifiers

- We want the model that makes the least assumptions
  - As we have 4 clases: (economics, sports, politics, art)

$$P(economics) = P(sports) = P(politics) = P(art) = 0{,}25$$

- Suppose that if the word "ball" appears in the text, then $p(sports|ball) = 0{,}7$
  - Remember that this is a counting problem

$$p(sports|ball) = \frac{count(sports,\ ball)}{\sum_{w \in V} count(sports, w)}$$

MaxEnt Classifiers: Example

- We adjust the distribution, again choosing the model that makes less assumptions
  - $p(sports \mid ball) = 0.7$
  - $p(politics \mid ball) = 0.1$
  - $p(economics \mid ball) = 0.1$
  - $p(art \mid ball) = 0.1$
- We factor in each of the constraints coming from the training data
  - $p(politics \mid Bush) = 0.8$
  - $p(sports \mid game) = 0.6$
  - $p(economic \mid stock) = 0.5, \ldots$

MaxEnt Classifiers

- **Infinite number** of models that satisfy a set of constraints.
- Maximum Entropy modeling lets us create a distribution that **abides by all these constraints**, while being as **uniform** as possible
- **Most uniform = Maximum Entropy**
- We **don’t make any additional assumptions** to what is supported by the data

MaxEnt Classifiers

- More info:
  - http://cseweb.ucsd.edu/~elkan/254/ari_talk.pdf
  - https://nadesnotes.wordpress.com/2016/09/05/natural-language-processing-nlp-fundamentals-maximum-entropy-maxent/
- NLTK
  - http://www.nltk.org/_modules/nltk/classify/maxent.html
- SKLEARN ¿?

MaxEnt Classifiers

From Jurafsky & Martin’s "Speech and Language :

> “Berger et al. (1996)* show that the solution to the proposed optimization problem (i.e., select the model with maximum entropy that abides by all these constraints)turns out to **be exactly the probability distribution of a multinomial logistic regression model whose weights W maximize the likelihood of the training data!**”

*https://aclweb.org/anthology/J/J96/J96-1002.pdf

- Detailed (and very mathematical) justification: http://qr.ae/TU15yf

Support Vector Machines

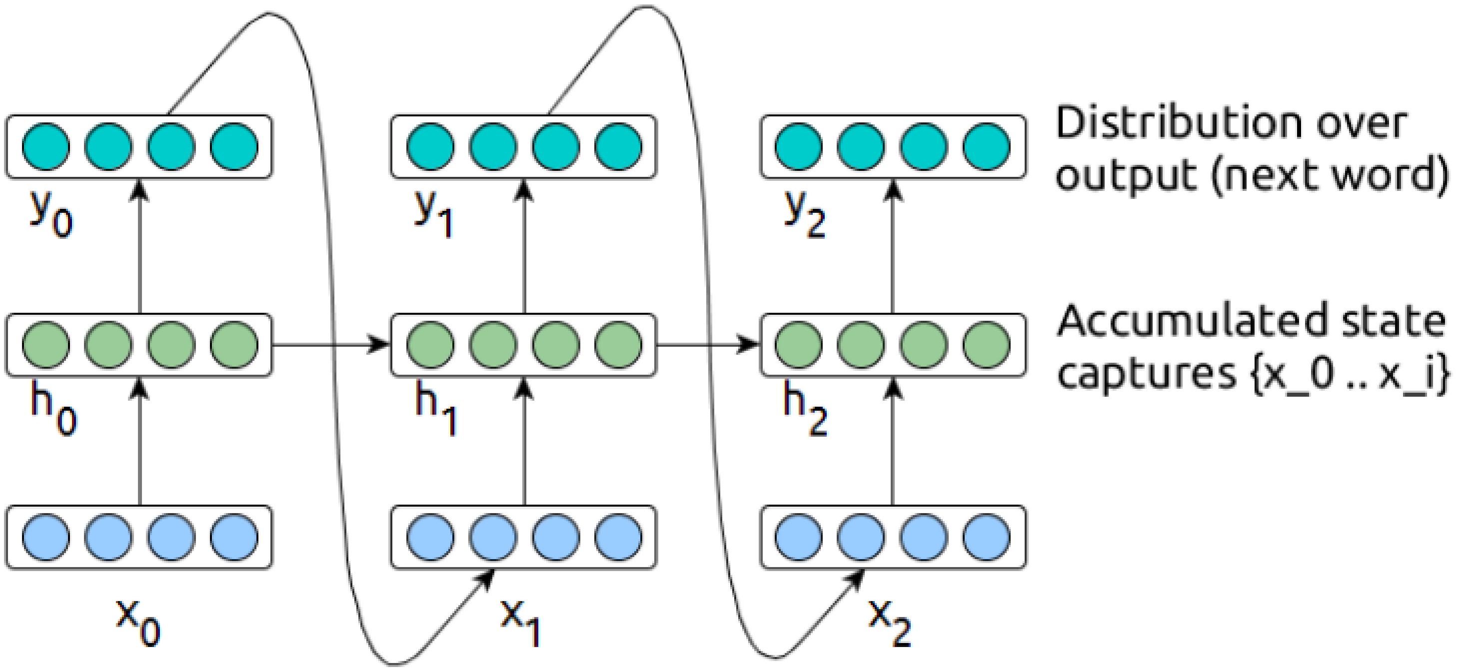
- **Trade-off between robustness and accuracy** (guaranteed by the large margin constraint)
- Well suited for **sparse data**
- Well suited for **high dimensional data**
- **SoftA before Deep Learning**

Model	Yelp'13	Yelp'14	Yelp'15	IMDB
SVM+TF	59.8	61.8	62.4	40.5
CNN	59.7	61.0	61.5	37.5
Conv-GRNN	63.7	65.5	66.0	42.5
LSTM-GRNN	65.1	67.1	67.6	45.3
fastText	64.2	66.2	66.6	45.2

Source: https://arxiv.org/pdf/1607.01759.pdf

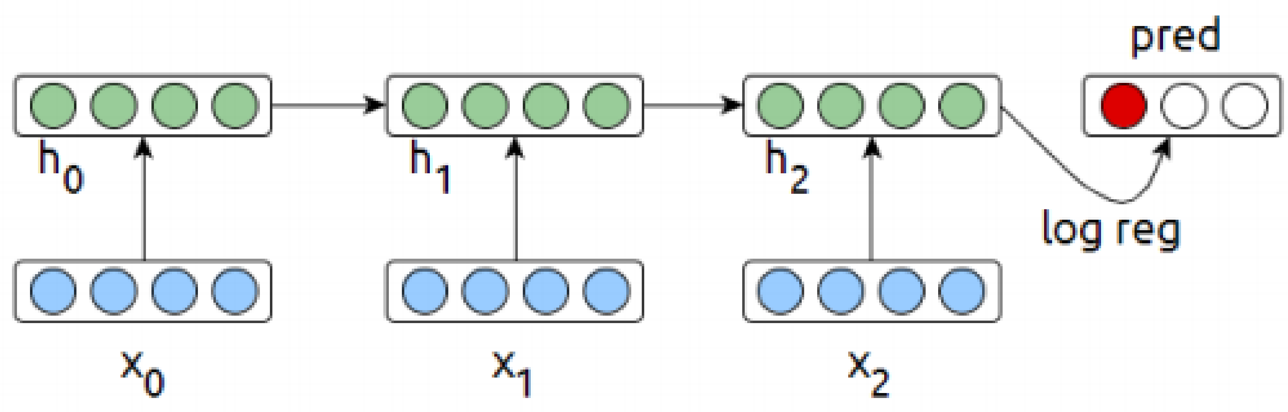
Recurrent Neural Networks (RNN)

- Model Sequences (texts are sequences of terms)



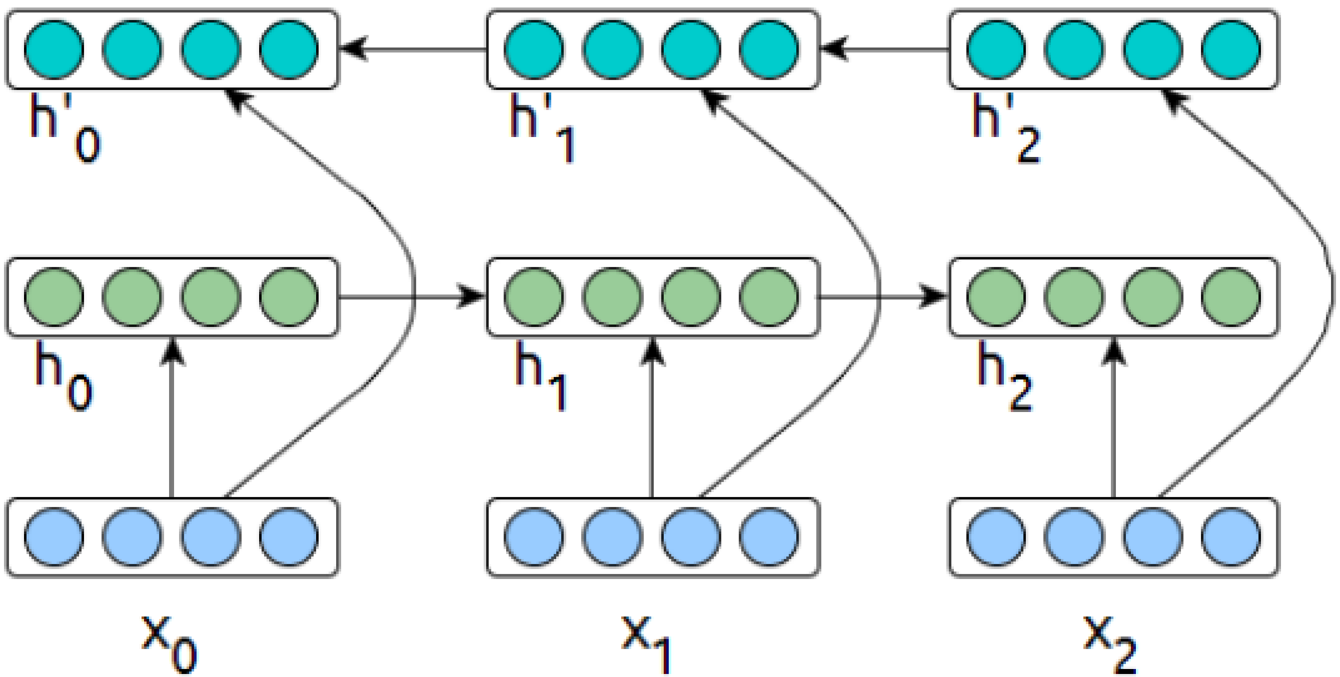
Recurrent Neural Networks (RNN)

- Model Sequences (texts are sequences of terms)



Bi-Directional RNNS

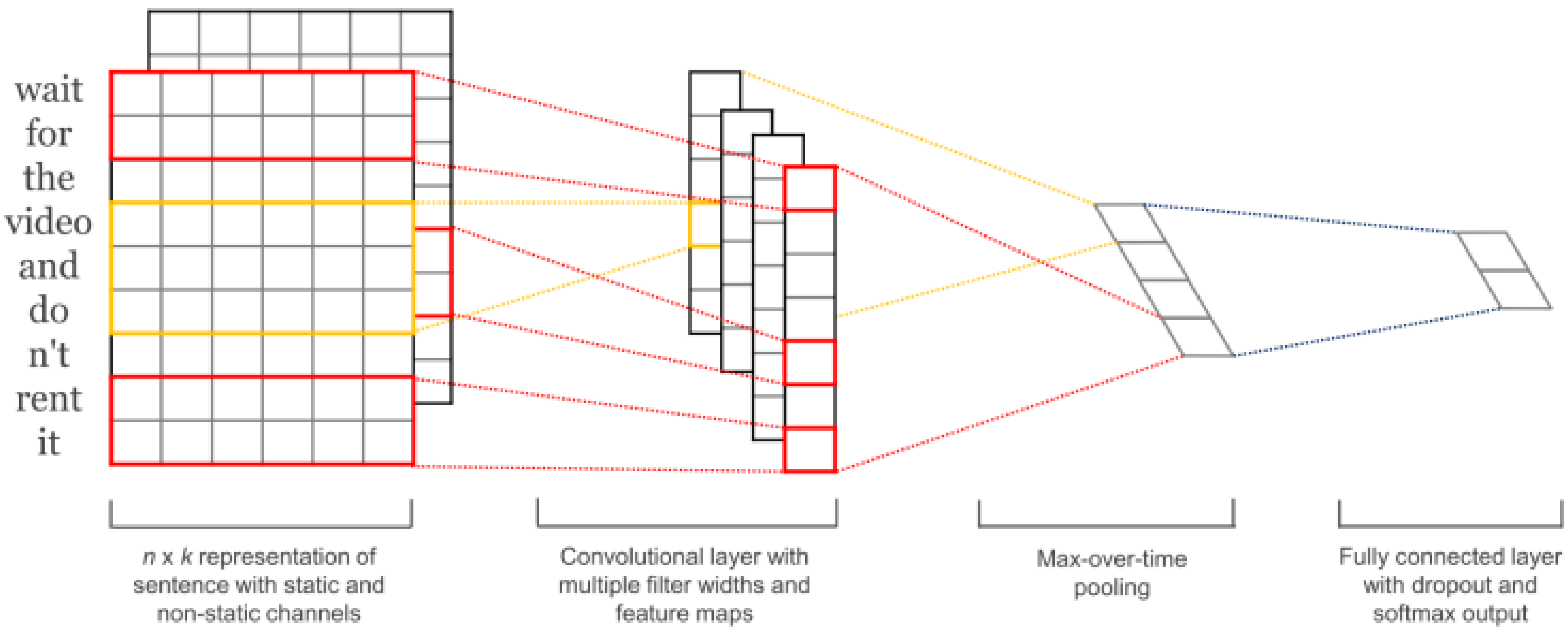
- Capture forward and backward dependencies



Convolutional Neural Networks (CNN)

- **Especially suited for images**
  - Captures spatial relationships (invariant to position or rotation)
  - Detect local features and compose them into higher-level features
- **But for texts?**
  - Can take some textual structure into account
  - Do not capture sequential information
  - But, … they work → All models are wrong, some are useful

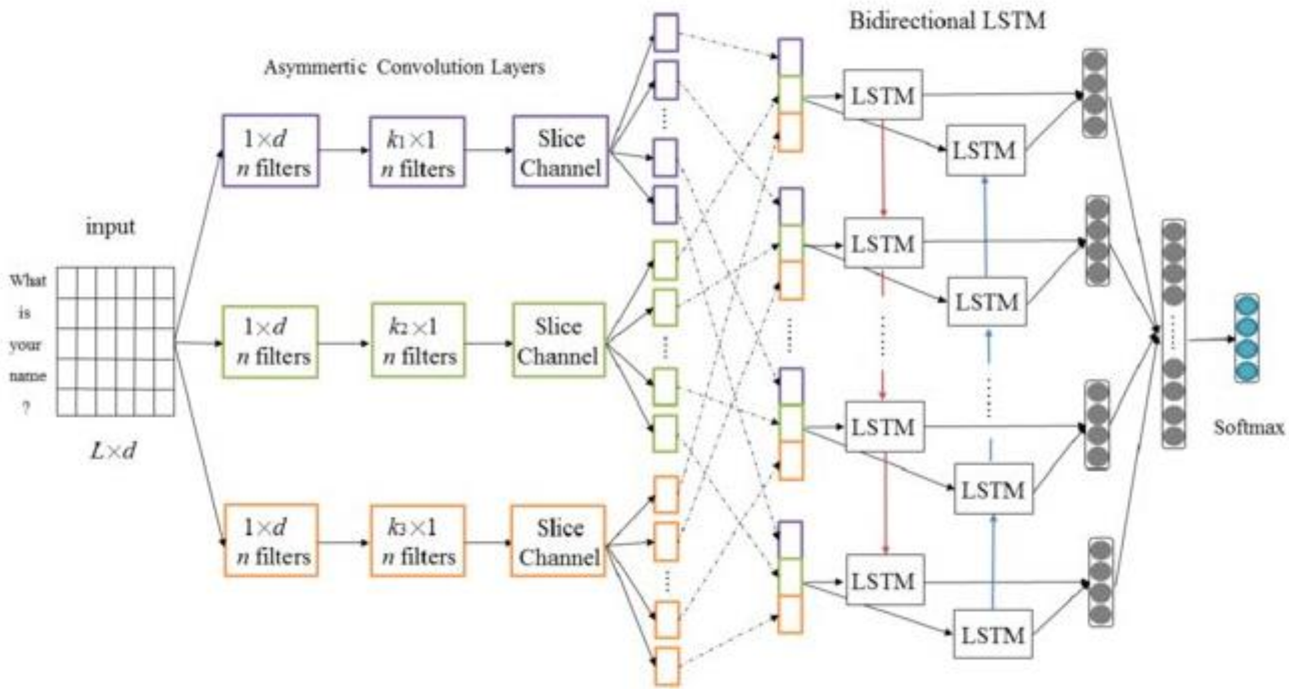
Convolutional Neural Networks (CNN)



Mixed model

- CNN + LSTM
  - Convolution to extract features
  - LSTM to codify sequence

AC-BLSTM: Asymmetric Convolutional Bidirectional LSTM Networks for Text Classification



Transformers: Coming soon

The Real World

No training data?

- Manually written rules
- Need careful crafting
  - Human tuning on development data
  - Time-consuming

The Real World

Very little data?

- **Naïve Bayes**
- Get **more labeled data**
  - DIY
  - Gamification
  - Pay for it (e.g. Mechanical Turk)
- Try **semi-supervised** training methods:
  - Bootstrapping
  - EM over unlabeled documents
  - ...

The Real World

A reasonable amount of data?

- Perfect for all the clever classifiers
  - SVM
  - Regularized Logistic Regression/MaxEnt Models
- User-interpretable **decision trees**
  - Users like to hack
  - Management likes quick fixes

BIG Data?

- Try Deep Learning

The Real World

Text Categorization Based on Regularized Linear Classification Methods (Reuters Dataset)

- Naïve Bayes: 77.0%
- Linear regression: 86.0%
- Logistic regression: 86.4%
- Support vector machine: 86.5%

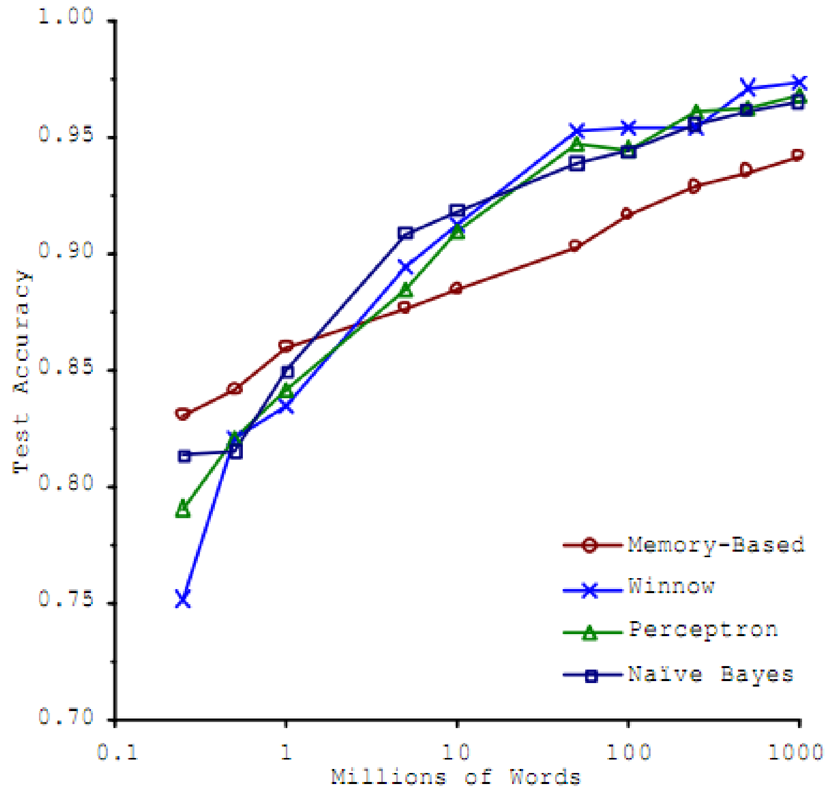
More on Reuters

- LSTM: 88%

The Real World

Accuracy as a function of data size

- With enough data Classifier may not matter



Brill and Banko on spelling correction

How to tweak performance

- **Domain-specific features** and weights:
  - Very important in real performance
- Sometimes need to **collapse terms**
  - Part numbers, chemical formulas, …
- **Upweighting**: Counting a word as if it occurred twice
  - **Title words**: William W. Cohen and Yoram Singer (1996) Learning to Query the Web
  - **First sentence of each paragraph**: Murata (1999) A design method for continuous deadbeat control systems
  - **Sentences that contain title words**: Ko et al, (2002) Improving text categorization using the importance of sentences
- **N-Grams**
  - Lexical information (negation, object-action, …)
  - Semantic Information (Named Entities, compound names)

How to tweak performance

Table 1. Results comparison (MacroAveraged F_1) on reuters 21578

	N=2	N=3	N=4	N=5	N=6	N=7
K=100	0.458	0.643	0.701	0.704	0.680	0.629
K=200	0.462	0.650	0.702	0.702	0.681	0.626
K=300	0.462	0.648	0.703	0.707	0.685	0.626
K=400	0.462	0.646	0.701	0.704	0.686	0.624
K=500	0.462	0.649	0.700	0.703	0.685	0.621
K=600	0.462	0.648	0.699	0.702	0.685	0.620
K=700	0.462	0.648	0.697	0.701	0.685	0.622
K=800	0.462	0.648	0.695	0.701	0.685	0.622

A Study of Text Classification Natural Language Processing Algorithms for Indian Languages

How to tweak performance

- NLP-specific **Feature Engineering**
  - POS Tags
    - First Level word disambiguation

      Book (NN) → *Books Sales*

      Book (VB)) → *Travel Agency*
    - Extract proper nouns (NP) as entities or events

How to tweak performance

Table 5. Rocchio and *PRC* performances on different feature sets of the ANSA corpus

	Rocchio	*PRC*		
	Tokens	Tokens	+CN	+POS+CN
Category	*BEP*	f_1	f_1	f_1
News	50.35	68.99	68.58	69.30
Economics	53.22	76.03	75.21	75.39
Politics	60.19	59.58	62.48	63.43
Entertainment	75.91	77.63	76.48	76.27
Sport	67.80	80.14	79.63	79.67
μf_1 (8 cat.)	61.76±0.5	71.00±0.4	71.80±0.4	72.37±0.4

Complex Linguistic Features for Text Classification: a comprehensive study

How to tweak performance

- NLP-specific **Feature Engineering**
  - POS Tags
  - Dependency Parsing
    - Better relationships than those discovered by n-grams

	Reuters			NSF			MiniNg20		
	Key#	microF	macroF	Key#	microF	macroF	Key#	microF	macroF
AWDCP	4138	86.03	45.26	3908	66.01	47.68	2914	54.23	51.65
AWDP	4198	85.96	45.07	2829	65.07	47.10	3114	54.13	51.53
AWP	3976	85.84	44.85	2478	64.58	46.49	2863	53.62	51.02
AW	20292	85.58	43.83	13424	64.46	46.11	30970	46.42	43.44

IMPROVING TEXT CLASSIFICATION PERFORMANCE WITH THE ANALYSIS OF LEXICAL DEPENDENCIES AND CLASS-BASED FEATURE SELECTION

Additional Resources

- TREC: Text Retrieval Conference
  - https://trec.nist.gov/data/qa.html
- Text Classification Systems Compilation (mostly DL)
  - https://github.com/brightmart/text_classification
- Bag of Tricks for Efficient Text Classification
  - Paper of FAIR for Text classification
  - Several well-known dataset
  - Make sense of the state of the art
- Fine-tuned Language Models for Text Classification
  - Another FAIR review on text classification
- Extensive review of text classification methods (mostly DL)
  - https://github.com/fendouai/Awesome-Text-Classification
- Text Classifier Algorithms in Machine Learning
  - Key text classification algorithms with use cases and tutorials
  - **https://blog.statsbot.co/text-classifier-algorithms-in-machine-learning-acc115293278**

Practical

- **Use NTLK classifiers:** http://www.nltk.org/book/ch06.html
  - **Some classifiers**
    - ConditionalExponentialClassifier
    - DecisionTreeClassifier
    - MaxentClassifier
    - NaiveBayesClassifier
    - WekaClassifier
  - **Tricky input format**
  - **Cross-Validation, Model Optimization, …. ¿?**

Practical

- **Use NTLK classifiers**

```
>>> featuresets = [(gender_features(n), gender) for (n, gender) in labeled_names]
>>> train_set, test_set = featuresets[500:], featuresets[:500]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
>>> classifier.classify(gender_features('Neo'))
'male'
>>> classifier.classify(gender_features('Trinity'))
'female'
```

```
>>> print(nltk.classify.accuracy(classifier, test_set))
0.77
```

Practical

- **Use NTLK classifiers**

```
>>> classifier.show_most_informative_features(5)
Most Informative Features
             last_letter = 'a'            female : male   =     33.2 : 1.0
             last_letter = 'k'              male : female =     32.6 : 1.0
             last_letter = 'p'              male : female =     19.7 : 1.0
             last_letter = 'v'              male : female =     18.6 : 1.0
             last_letter = 'f'              male : female =     17.3 : 1.0
```

Practical

- **Use sklearn classifiers**
  - **Many (and I mean) many methodologies implemented**
    - Logistic Regression
    - LDA and QDA
    - Kernel Rigde Regression
    - SVM
    - SGD
    - NN
    - Gaussian Processes
    - Naïve Bayes
    - Decision Trees
    - Ensembles
    - Neural nets....

Practical

- **Use sklearn classifiers**
  - **Many (and I mean) many methodologies implemented**
  - **Includes all what we learned in ML2**
    - Optimization
    - Cross-validation
    - Regularization
  - **Take a look to the sklearn examples**
    - http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
    - https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f
    - https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

Practical

- **Deep Learning**
  - **Hugging Face:** https://huggingface.co/
    - Easy Python-Integration of Transformer-based Models
    - Pre-trained SoftA Models
  - **fastText**: https://github.com/facebookresearch/fastText
    - Facebook Research library for text classification
    - Includes pre-trained models and representations
  - Implement **your own models**
    - **Tensorflow**: Low-level library
      - Complex to implement
      - More flexibility
    - **Keras**: Wrapper on top lower-level libraries
      - API including many precompiled models and tools
      - Less flexible