CSE 587: Deep Learning for Natural Language Processing

Lecture 2. Text Classification and Language Modeling

Rui Zhang

Spring 2023

Outline - Key Concepts

NLP

- Text Classification
- Language Modeling
- Word Embedding

ML

- Discriminative Model vs Generative Model
- Objective Function
- Gradient Descent
- Evaluation
- Statistical Testing
- Feed-forward Neural Networks and Recurrent Neural Networks

Text Classification

Input X	Output Y	Task
Text	Label	Text Classification (e.g., Sentiment Analysis)
Text	Linguistic Structure	Structured Prediction (e.g., Part-of-Speech Tagging)
Text	Text	Text Generation (e.g., Translation, Summarization)

Classify sentences/documents into different classes.

- e.g., Sentiment Analysis: Positive, Neutral, Negative

Discriminative Model

Input X	Output Y	Task
Text	Label	Text Classification (e.g., Sentiment Analysis)
Text	Linguistic Structure	Structured Prediction (e.g., Part-of-Speech Tagging)
Text	Text	Text Generation (e.g., Translation, Summarization)

Classify sentences/documents into different classes.

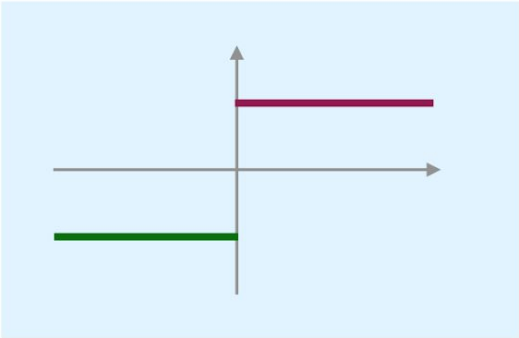
- e.g., Sentiment Analysis: Positive, Neutral, Negative

Discriminative Model: Calculate the conditional probability distribution of class labels Y given the input data X.

$$P(y|X)$$

From Prediction Score to Probability

$\boldsymbol{w}^\top \boldsymbol{x}$ →



→ $y = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} \geq 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases}$

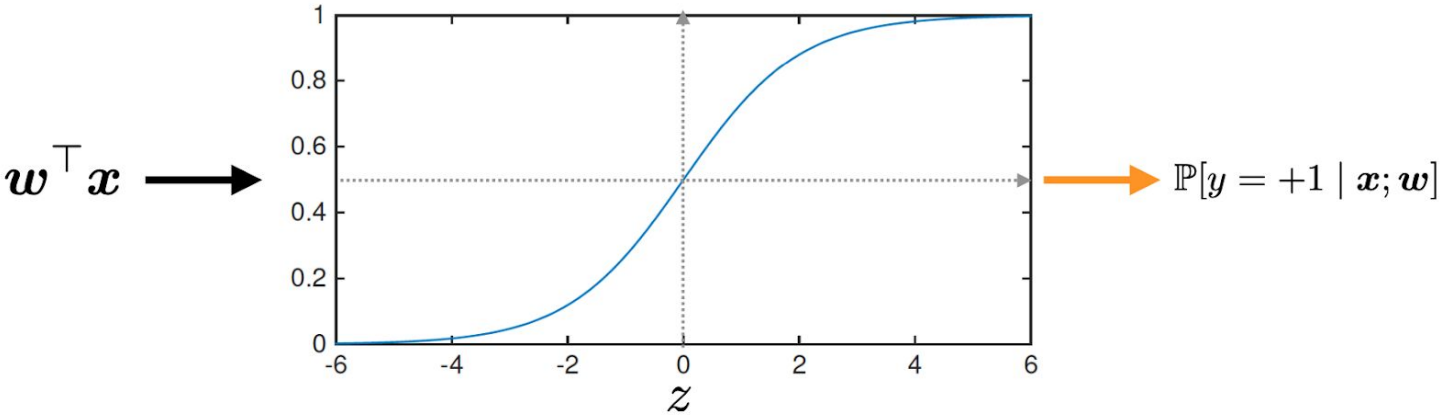
$\boldsymbol{w}^\top \boldsymbol{x}$ →

→ $\mathbb{P}[y = +1]$

$\mathbb{P}[y = -1] = 1 - \mathbb{P}[y = +1]$

Sigmoid for Binary Classification

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



$$\mathbb{P}\left[y = +1 | \boldsymbol{x}; \boldsymbol{w}\right] = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^\top \boldsymbol{x}}}$$

$$\mathbb{P}\left[y = -1 | \boldsymbol{x}; \boldsymbol{w}\right] = 1 - \mathbb{P}\left[y = +1 | \boldsymbol{x}; \boldsymbol{w}\right] = \sigma(-\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + e^{\boldsymbol{w}^\top \boldsymbol{x}}}$$

Softmax for Multi-class Classification

Softmax extends the idea of sigmoid into a multi-class classification.
It converts scores to a probability distribution of class labels.

The predicted probability for the j'th class given a sample vector $\mathbf{x}$ and a weighting vector $\mathbf{w}$ is

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

Softmax Example

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

$$\begin{bmatrix} 8 \\ 5 \\ 0 \end{bmatrix}$$

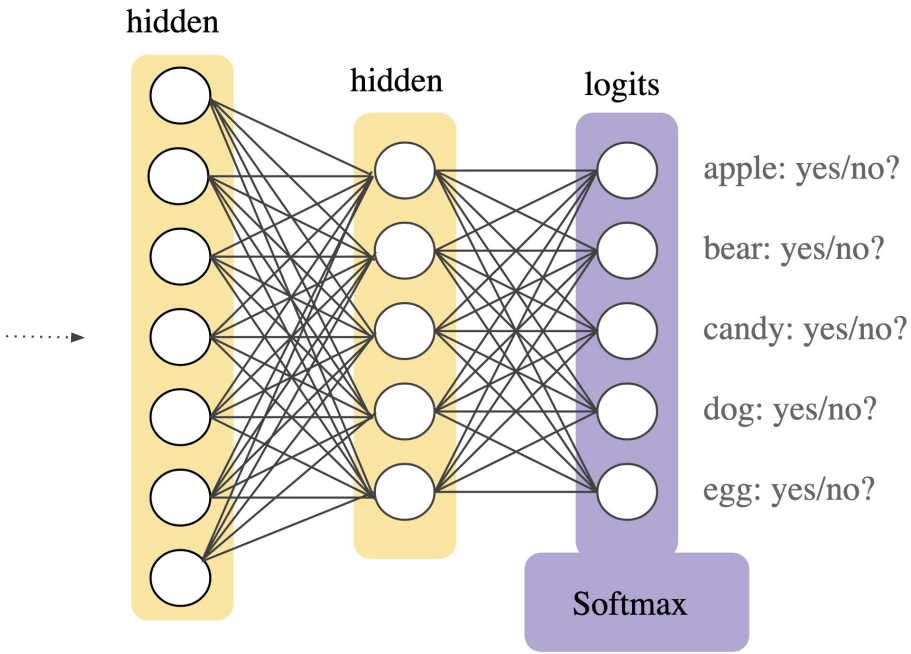
$$\sum_{j=1}^{K} e^{z_j} = e^{z_1} + e^{z_2} + e^{z_3} = 2981.0 + 148.4 + 1.0 = 3130.4$$

$$e^{z_1} = e^{8} = 2981.0$$
$$e^{z_2} = e^{5} = 148.4$$
$$e^{z_3} = e^{0} = 1.0$$

$$\frac{2981.0}{3130.4} = 0.9523$$
$$\frac{148.4}{3130.4} = 0.0474$$
$$\frac{1.0}{3130.4} = 0.0003$$

https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

Softmax in Neural Networks



Class	Probability
apple	0.001
bear	0.04
candy	0.008
dog	0.95
egg	0.001

https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

Discriminative Model

The discriminative model is parameterized by θ

$$P(y|X;\theta)$$

Discriminative Model Objective Function

The discriminative model is parameterized by θ

$$P(y|X;\theta)$$

We often use **negative log likelihood** over training data as our **objective function** or **loss function.** It is a function of θ

$$\mathcal{L}(\theta) = -\sum_{(X,y)\in\mathcal{D}_{\text{train}}} \log P(y|X;\theta)$$

Discriminative Model Objective Function

The discriminative model is parameterized by θ

$$P(y|X;\theta)$$

We often use **negative log likelihood** over training data as our **objective function** or **loss function.** It is a function of θ

$$\mathcal{L}(\theta) = -\sum_{(X,y)\in\mathcal{D}_{\text{train}}} \log P(y|X;\theta)$$

We then optimize the parameters to minimize the loss. Better model has lower loss

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta)$$

Optimize Objective Function by Gradient Descent

Calculate the gradient of the loss function with respect to the parameter

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

Update θ by moving a small step in the gradient direction to decrease the loss

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

η is the **learning rate**

Gradient Descent

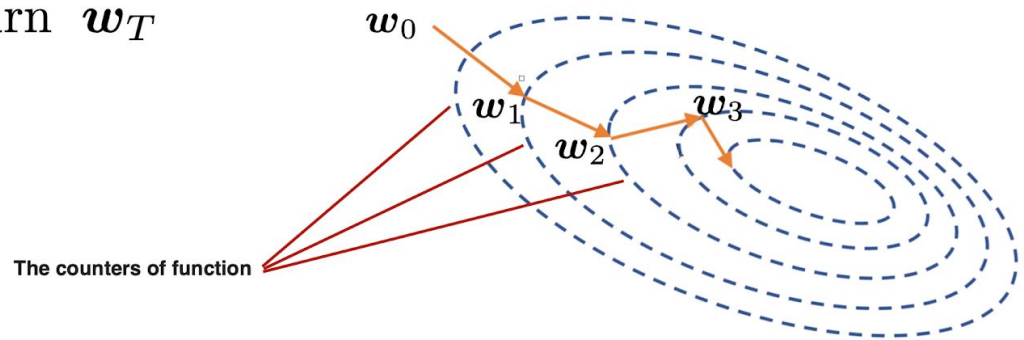
$$\boldsymbol{w}_0 = \mathbf{0}$$

for $t = 1, 2, \ldots, T$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla f(\boldsymbol{w}_t)$$

end for

return $\boldsymbol{w}_T$



Evaluation

Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers

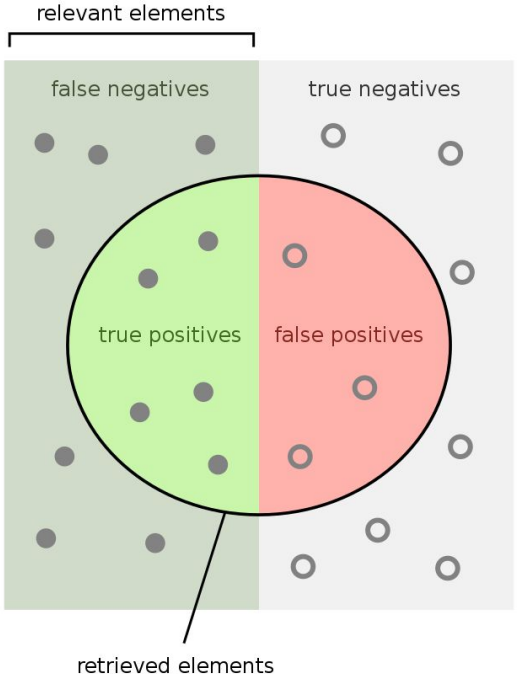
Evaluation

Precision

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall

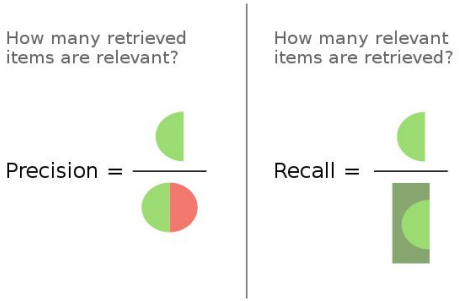
$$\text{Recall} = \frac{tp}{tp + fn}$$



F1 score: the harmonic mean of precision and recall

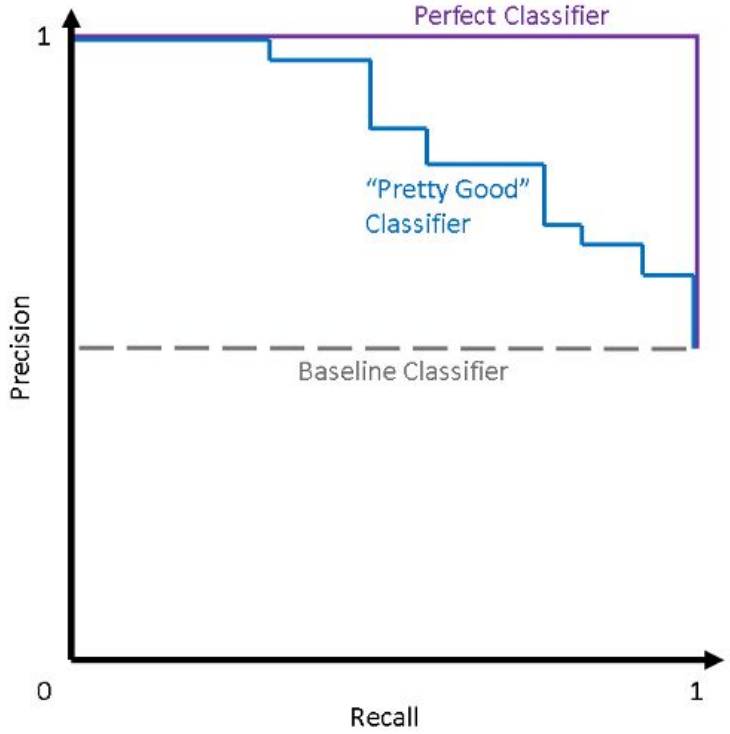
$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- harmonic mean < geometric mean < arithmetic mean



https://en.wikipedia.org/wiki/Precision_and_recall

Precision-Recall Curve



https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248

ROC Curve

Recall is also called **True Positive Rate**

- True Positive / (True Positive + False Negative)

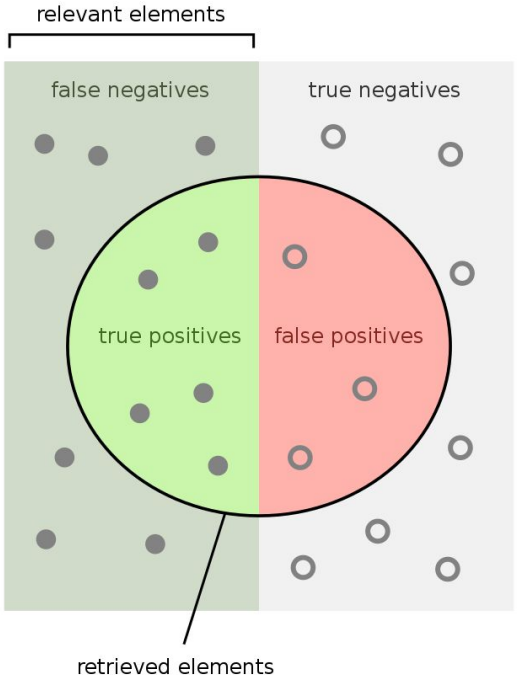
We can also define **False Positive Rate**

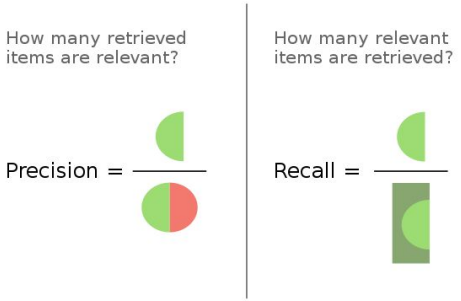
- False Positive / (False Positive + True Negative)

Be default, we can use 0.5 as threshold, but we can use other threshold as well.

As we change the threshold, both TPR and FPR change.

ROC curve: Receiver Operating Characteristic





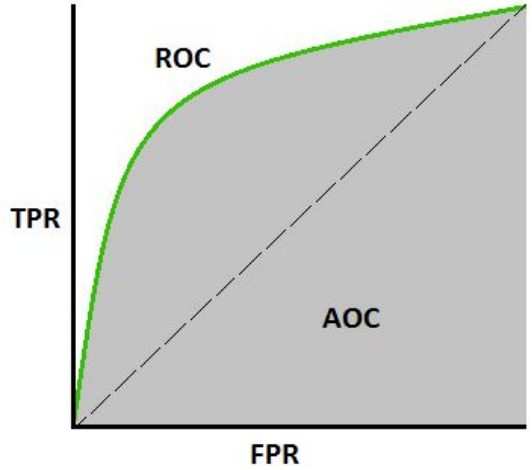
ROC Curve

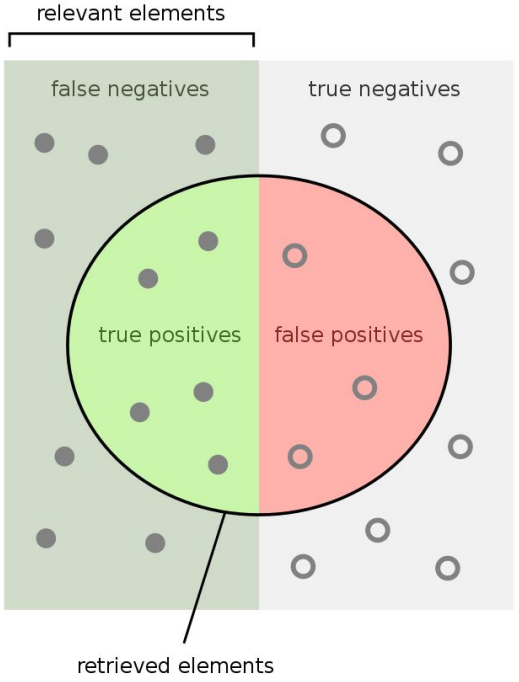
Recall is also called **True Positive Rate**

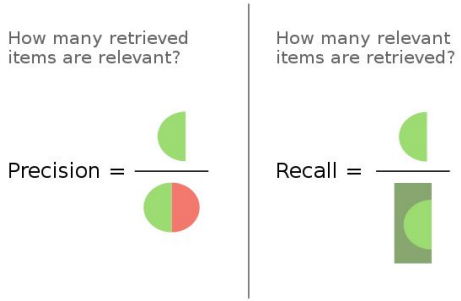
- True Positive / (True Positive + False Negative)

We can also define **False Positive Rate**

- False Positive / (False Positive + True Negative)





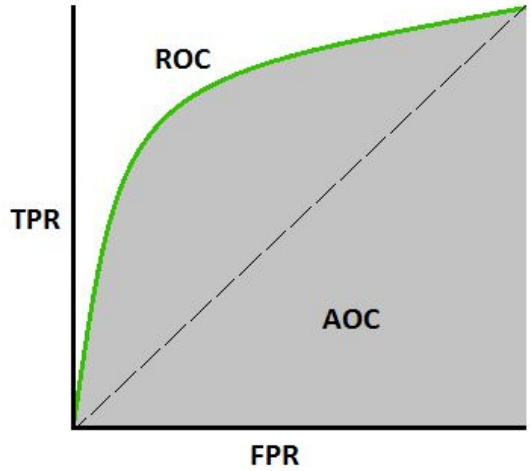


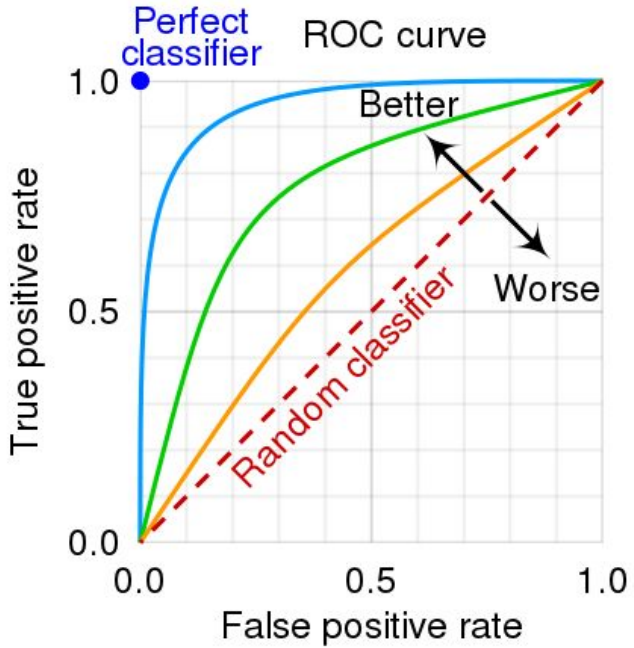
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

AUC

AUC: Area under Curve

- AUC = 1: The perfect classifier
- AUC = 0.5: The random classifier
- AUC = 0: The worst classifier





Evaluation for Ranking

Classification: order of predictions doesn't matter

Ranking: order of predictions does matter

- Search engines: Predict which documents match a query on a search engine.

k	Document ID	Predicted Relevance	Actual Relevance
1	06	0.90	Relevant (1.0)
2	03	0.85	Not Relevant (0.0)
3	05	0.71	Relevant (1.0)
4	00	0.63	Relevant (1.0)
5	04	0.47	Not Relevant (0.0)
6	02	0.36	Relevant (1.0)
7	01	0.24	Not Relevant (0.0)
8	07	0.16	Not Relevant (0.0)

https://queirozf.com/entries/evaluation-metrics-for-ranking-problems-introduction-and-examples

Evaluation for Ranking

Average Precision

- It is the area under the precision-recall curve.
- It is equivalent to compute the following

$$\text{AveP} = \frac{\sum_{k=1}^{n} P(k) \times \text{rel}(k)}{\text{number of relevant documents}}$$

where rel(k) = 1 if the item at rank k is relevant, 0 otherwise.

AveP = (1/1 + 2/3 + 3/4 + 4/6) / 4

k	Document ID	Predicted Relevance	Actual Relevance
1	06	0.90	Relevant (1.0)
2	03	0.85	Not Relevant (0.0)
3	05	0.71	Relevant (1.0)
4	00	0.63	Relevant (1.0)
5	04	0.47	Not Relevant (0.0)
6	02	0.36	Relevant (1.0)
7	01	0.24	Not Relevant (0.0)
8	07	0.16	Not Relevant (0.0)

Evaluation for Ranking

Mean Average Precision

- average of AP over all examples in a test set.

There are many metrics for ranking.

- DCG/NDCG: the document relevance is a real number, not simple 0 or 1.

k	Document ID	Predicted Relevance	Actual Relevance	DCG @k
1	06	0.90	Relevant (1.0)	1.0
2	03	0.85	Not Relevant (0.0)	1.0
3	05	0.71	Relevant (1.0)	1.5
4	00	0.63	Relevant (1.0)	1.93
5	04	0.47	Not Relevant (0.0)	1.93
6	02	0.36	Relevant (1.0)	2.29
7	01	0.24	Not Relevant (0.0)	2.29
8	07	0.16	Not Relevant (0.0)	2.29

Evaluation for Natural Language Generation

(We will talk about this in more detail when we have lectures on NLG and Summarization.)

Automatic Evaluation

- Machine Translation: BLEU
- Summarization: ROUGE, METEOR
- Embedding-based Metric: MoverScore, BERTScore

Human Evaluation

Is Automatic Metric really reliable? Correlation Analysis

Statistical Testing

We have two models with similar accuracies. How can we tell whether the differences are due to consistent trends that hold on other datasets?

	Dataset 1	Dataset 2	Dataset 3
Generative	**0.854**	**0.915**	0.567
Discriminative	0.853	0.902	**0.570**

We need perform **Statistical (significance) testing**!

See The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing (Dror et al., ACL 2018) for a complete overview.

Slides Credit: Graham Neubig

Significance Testing: Basic Idea

Given a quantity, we test certain values of uncertainty with respect to the quantity, e.g.

- **p-value**: what is the probability that a difference with another quantity is by chance (lower = more likelihood of a significant difference).

- **confidence interval**: what is the range under which we could expect another trial to fall?

Slides Credit: Graham Neubig

Unpaired vs. Paired Tests

Unpaired Test: Compare means of a quantity on two unrelated groups

- Example: test significance of difference of accuracies of a model on two datasets

Paired Test: Compare means of a quantity on one dataset under two conditions

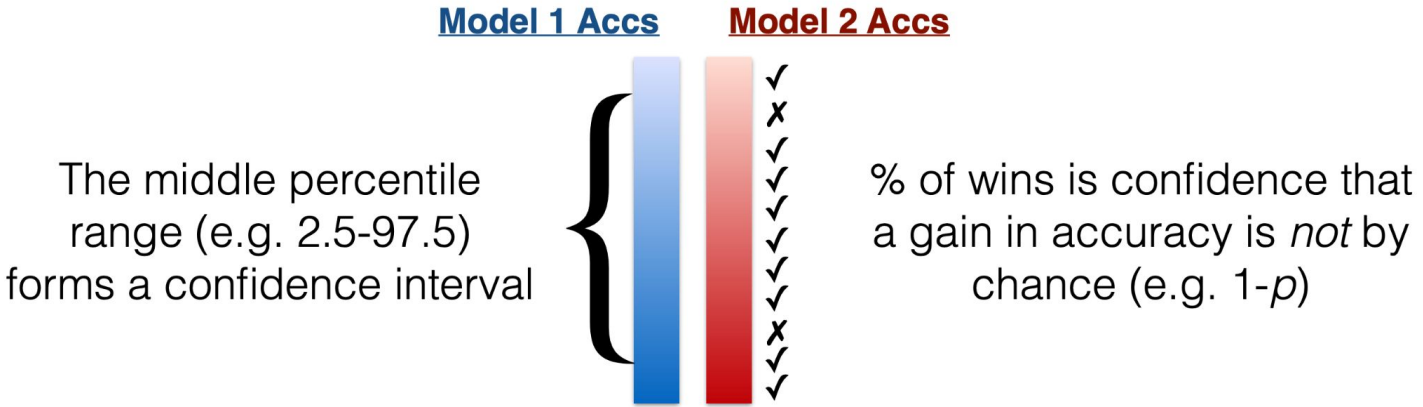
- Example: test significance of difference of accuracies of two models on one dataset

We are most commonly interested in **Paired Test**!

Slides Credit: Graham Neubig

Bootstrap Tests

A method that can measure p-values, confidence intervals, etc. by re-sampling data.
Sample many (e.g. 10,000) subsets from your dev/test set with replacement.
Measure accuracies on these many subsets.
Easy to implement, applicable to any evaluation measure, but somewhat biased on small datasets.



Slides Credit: Graham Neubig

Discriminative Model vs Generative Model

Discriminative Model: Calculate the conditional probability distribution of class labels Y given the input data X.

$$P(y|X)$$

Generative Model: Directly Model

$$P(X) \qquad P(X, Y)$$

Language Modeling

Language Modeling: Calculate the probability of a sentence.

Sentence X of L words $X = x_1, \ldots, x_{i-1}, x_i, \ldots, x_L$

Decompose the joint probability by chain rule

$$P(X) = \prod_{i=1}^{L} P(x_i | x_1, \ldots, x_{i-1})$$

Language Modeling

Language Modeling: Calculate the probability of a sentence.

Sentence X of L words $X = x_1, \ldots, x_{i-1}, x_i, \ldots, x_L$

Decompose the joint probability by chain rule

$$P(X) = \prod_{i=1}^{L} P(x_i | x_1, \ldots, x_{i-1}) \qquad \log P(X) = \sum_{i=1}^{L} \log P(x_i | x_1, \ldots, x_{i-1})$$

- We often deal with log probabilities because of numerical stabilities and other benefits.
- Different ways for calculating the probability of next word give different language modeling approaches.

Unigram Language Models

Assumption:

$$P(x_i|x_1, \ldots, x_{i-1}) \approx P(x_i)$$

Count-based Maximum-likelihood estimation (MLE):

$$P(x_i) = \frac{c_{\text{train}}(x_i)}{\sum_{\tilde{x}} c_{\text{train}}(\tilde{x})}$$

Bigram Language Models

Assumption:

$$P(x_i|x_1, \ldots, x_{i-1}) \approx P(x_i|x_{i-1})$$

Count-based Maximum-likelihood estimation (MLE):

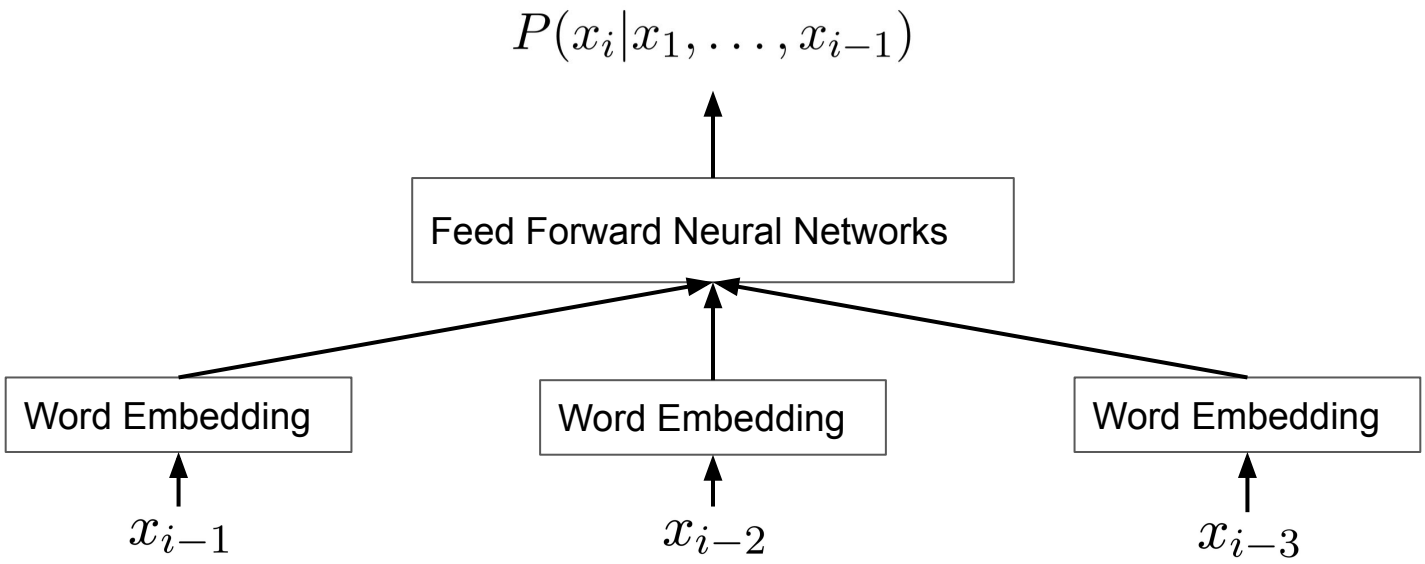
$$P(x_i|x_{i-1}) = \frac{c_{\text{train}}(x_{i-1}, x_i)}{\sum_{\tilde{x}} c_{\text{train}}(x_{i-1}, \tilde{x})}$$

Higher-order n-gram Models

Unigram (1-gram), Bigram (2-gram)
Trigram (3-gram)

Feed-Forward Neural Network Language Models

Assumption: The distribution of next word can be parameterized by feed neural networks.



A Neural probabilistic language model. Bengio et al., 2003

A fixed-window neural language model

output distribution

$$\hat{y} = \text{softmax}(W_2 h)$$

hidden layer

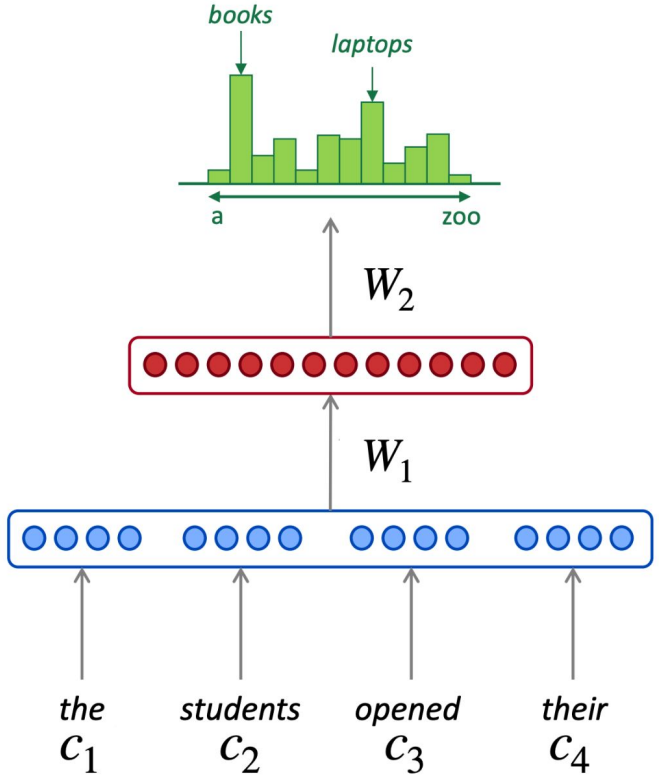
$$h = f(W_1 x)$$

concatenated word embeddings

$$x = [c_1; c_2; c_3; c_4]$$

words / one-hot vectors

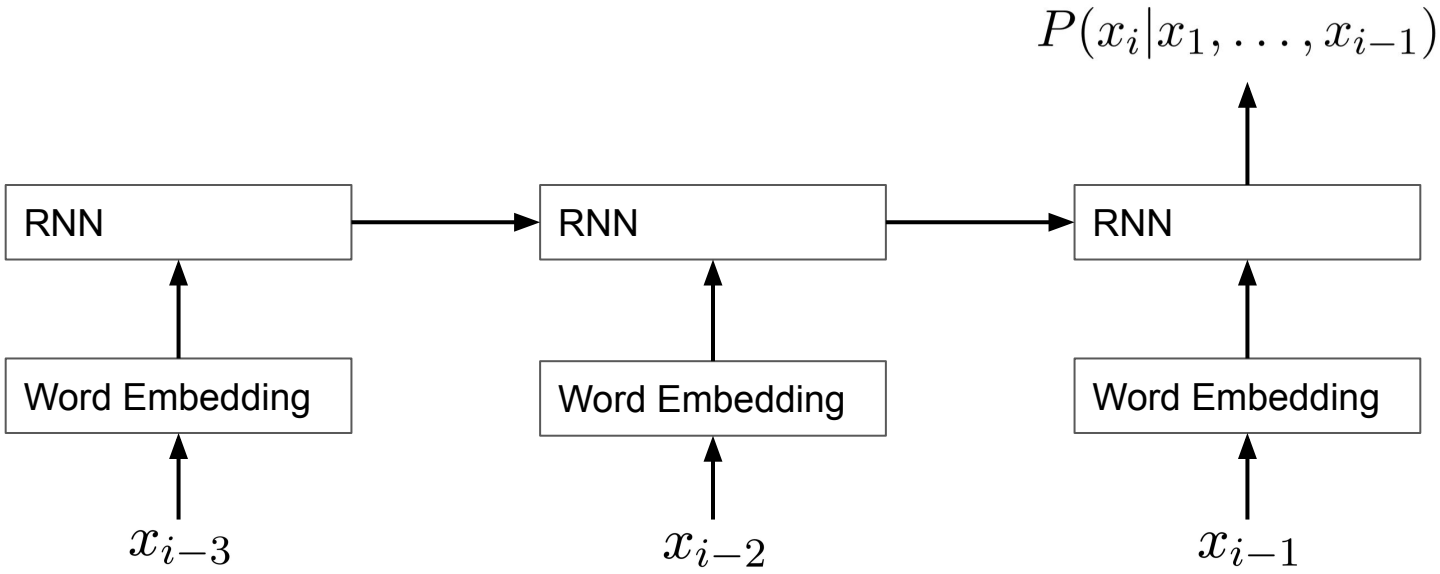
$$c_1, c_2, c_3, c_4$$



https://people.cs.umass.edu/~miyyer/cs685/slides/02-neural-lms.pdf

Recurrent Neural Network Language Models

Assumption: The distribution of next word can be parameterized by recurrent neural networks.



Recurrent neural network based language model. Mikolov et al., 2010

A RNN Language Model

output distribution

$$\hat{y} = \text{softmax}(W_2 h^{(t)})$$

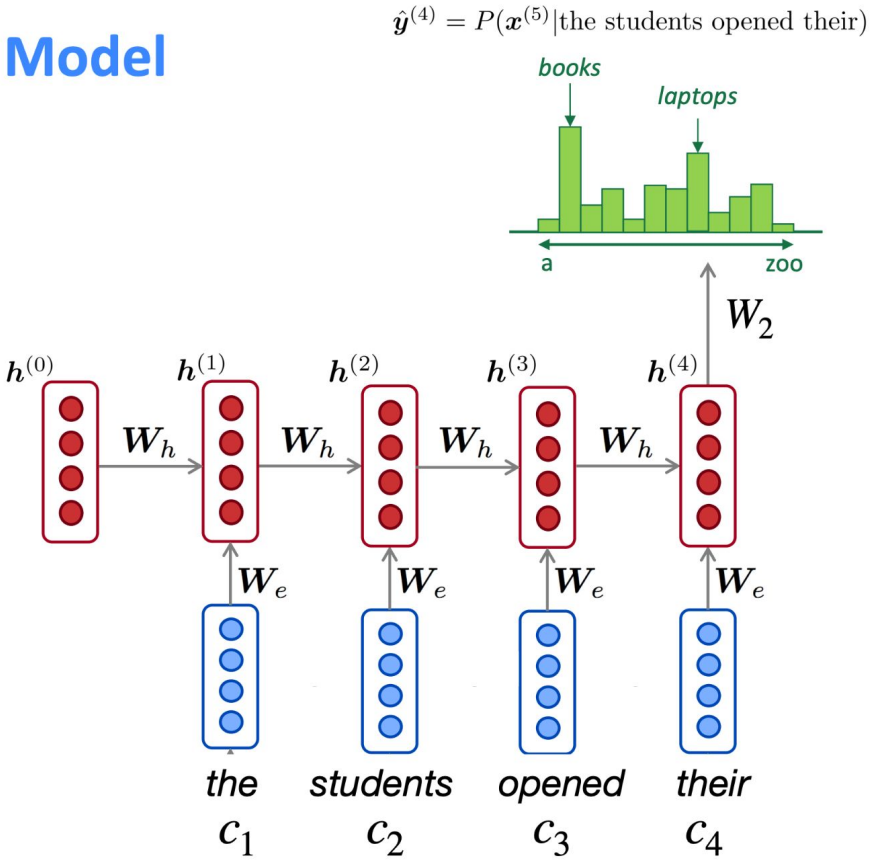
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

h(0) is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



https://people.cs.umass.edu/~miyyer/cs685/slides/02-neural-lms.pdf

Evaluation of Language Modeling

We train our language model on **training set**. How good is our language model?
A good language model should assign a high probability to text in **test set**!

Log-Likelihood:

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

Evaluation of Language Modeling

We train our language model on **training set**. How good is our language model?
A good language model should assign a high probability to text in **test set**!

Log-Likelihood:

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

Per-word Log Likelihood:

$$WLL(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

Per-word Cross Entropy

$$H(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} -\log_2 P(E)$$

Evaluation of Language Modeling - Perplexity

Perplexity is the inverse probability of the test set.
Lower perplexity, higher probability, better model.

$$ppl(\mathcal{E}_{test}) = 2^{H(\mathcal{E}_{test})} = e^{-WLL(\mathcal{E}_{test})}$$

Evaluation of Language Modeling - Perplexity

Perplexity is the inverse probability of the test set.
Lower perplexity, higher probability, better model.

$$ppl(\mathcal{E}_{test}) = 2^{H(\mathcal{E}_{test})} = e^{-WLL(\mathcal{E}_{test})}$$

Perplexity is the exponentiated token-level negative log-likelihood.
e.g., for bigram LM:

$$PP(W) = \exp\Big(-\frac{1}{N}\sum_{i}^{N}\log p(w_i|w_{<i})\Big)$$

Usage of LMs

1. Score given sentences

2. Generate new sentences

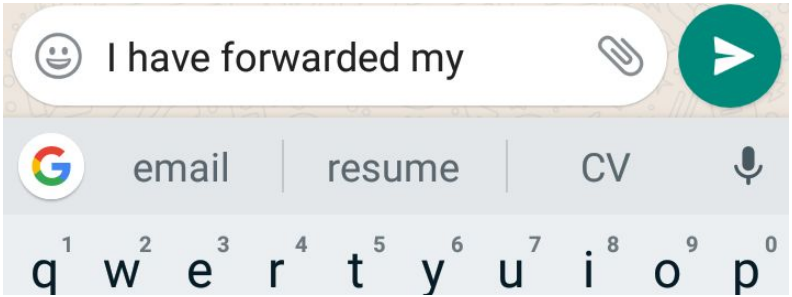
    while didn’t choose end-of-sentence symbol:

        calculate the probability distribution of next word

        sample a new word from the probability distribution of next word

3. Testbed for a neural networks

Usage of LMs



https://devopedia.org/language-modelling

Usage of LMs

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Generate Text from LM trained on Shakespeare