# NLP: Mid Term Review

Dr. Hafiz Khan

mkhan74@Kennesaw.edu

Computer Science, Kennesaw State University

https://ahafizk.github.io/

# Exam Question Pattern

- Question patterns you can expect
  - Multiple choice (MCQ) questions
  - Short questions
  - Simulation/model building based question
    - Similar as homework, i.e., formulating naïve bayes from text, language modeling, etc.
  - Scenario based question (may be)
- Total number of question 10
- Total Points 100 distributed among these questions.

# Regular Expressions: Disjunction

- Letters inside square brackets []
  - []
  - [aA]pple matches apple and Apple
  - [0123456789] matches any digit
- Ranges
  - [0-9] -- matches any digit
  - [a-z] -- matches any lowercase
  - [a-zA-Z] -- matches any uppercase
  - [a-e1-9] -- matches any letter or digit
  - Hyphen only has a special

| Pattern | Matches |
|---|---|
| `[wW]oodchuck` | Woodchuck, woodchuck |
| `[1234567890]` | Any digit |

| Pattern | Matches | |
|---|---|---|
| `[A-Z]` | An upper case letter | `Drenched Blossoms` |
| `[a-z]` | A lower case letter | `my beans were impatient` |
| `[0-9]` | A single digit | `Chapter 1: Down the Rabbit Hole` |

# Regular Expressions: Disjunction

- **Negation in Disjunction [^Ss]**
  - Carat means negation only when first in []
  - ^ right after the square bracket means a negation
  - [^A-Z]
  - [^Aa] means neither a capital A nor a lowercase a
  - [^e^] means not an e, and not ^
- Disjunction for longer strings
  - The pipe | for disjunction
  - a|b|c = [abc]
  - applelpie

| Pattern | Matches | |
|---|---|---|
| `[^A-Z]` | Not an upper case letter | `Oyfn pripetchik` |
| `[^Ss]` | Neither 'S' nor 's' | `I have no exquisite reason"` |
| `[^e^]` | Neither e nor ^ | `Look here` |
| `a^b` | The pattern a | `Look up a^b now` |

| Pattern | Matches |
|---|---|
| `groundhog\|woodchuck` | |
| `yours\|mine` | `yours mine` |
| `a\|b\|c` | = [abc] |
| `[gG]roundhog\|[Ww]oodchuck` | |

# Regular Expressions: ? * + .

- Special characters
  - ? means previous character is optional: **colou?r - color, colour**
  - . matches any character
    - e.g. **beg.n** matches **begun**, **begin, began**
  - Kleene Operators - named after Steven Kleene
  - * matches 0 or more of the previous characters
    - e.g. **oo*h** will match **ooh, oooh**, etc.
  - **(abc)*** will match **abc, abcabc**. etc.

| Pattern | Matches | |
|---|---|---|
| `colou?r` | Optional previous char | color colour |
| `oo*h!` | 0 or more of previous char | oh! ooh! oooh! ooooh! |
| `o+h!` | 1 or more of previous char | oh! ooh! oooh! ooooh! |
| `baa+` | | baa baaa baaaa baaaaa |
| `beg.n` | | begin begun begun beg3n |

| RE | Match |
|---|---|
| ^ | start of line |
| \$ | end of line |
| \b | word boundary |
| \B | non-word boundary |

# Text Normalization

- Every NLP task needs to do text normalization:

1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. Segmenting sentences in running text

# Language Model (LM)

- LM: Assign a probability to a sentence
  - Why?
  - Applications – machine translation, spell correction, speech recognition etc.

  - Machine Translation: return text in the target language
    - P(**high** winds tonight) > P(**large** winds tonight)
  - Spell Correction: return corrected spelling of input
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition: return a transcript of what was spoken
    - P(I saw a van) >> P(eyes awe of an)
  - + Summarization, question-answering, etc., etc.!!

# Language Model

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w1,w2,w3,w4,w5\ldots wn)$

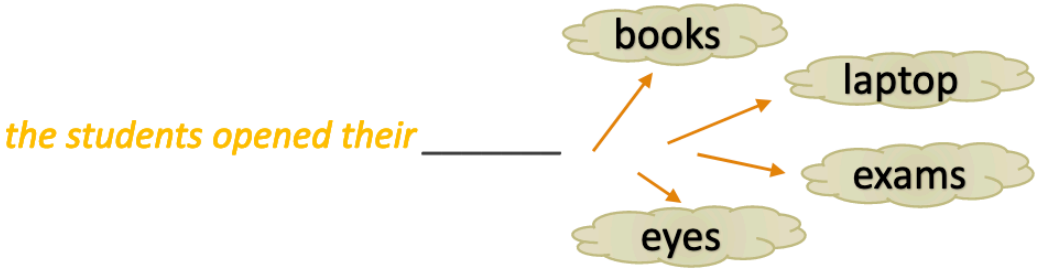
- Related task: probability of an upcoming word:

  $P(w5|w1,w2,w3,w4)$

- A model that computes either of these:

  $P(W)$ or $P(wn|w1,w2\ldots wn\text{-}1)$ is called a **language model**.

- **Language model in short** or **LM** is standard



# Unigram LM

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

- Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

# Example: Unigram LM

- Text: *beginning by, very Alice but was and? reading no tired of to into sitting sister the, bank, and thought of without her nothing: having conversations Alice once do or on she it get the book her had peeped was conversation it pictures or sister in, 'what is the use had twice of a book' 'pictures or' to*

  - P(of) = 3/66.
  - P(her) = 2/66
  - P(Alice) = 2/66
  - P(was) = 2/66
  - P(to) = 2/66
  - P(,) = 4/66
  - P(') = 4/66

# Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

# Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Exampl
e

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

# Practice Example

- Lets calculate the probability of the word “like” occurring after the word “really”.

P(like|really)=?

- *Thank you so much for your help.*
- *I really appreciate your help.*
- *Excuse me, do you know what time it is?*
- *I’m really sorry for not inviting you.*
- *I really like your watch.*

# Practice Example

- P(like|really) = count(really like) / count(really)
  = 1 / 3
  = 0.33
- Similar examples:
  - P(appreciate | really) = count(really appreciate) / count(really)
    = 1 / 3
    = 0.33
  - P(sorry | really) = count(really sorry) / count(really)
    = 1 / 3
    = 0.33

# More Examples:

- Berkeley Restaurant Project sentences
- Total 9222 sentences

can you tell me about any good cantonese restaurants close by
mid priced thai food is what i'm looking for
tell me about chez panisse
can you give me a listing of the kinds of food that are available
i'm looking for a good place to eat breakfast
when is caffe venezia open during the day

# Raw Bigram Count

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| to | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

# Compute bigram probabilities

- Normalize by unigrams:

| i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|
| 2533 | 927 | 2417 | 746 | 158 | 1093 | 341 | 278 |

- Result:

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| want | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| to | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| eat | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| chinese | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| food | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| lunch | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| spend | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

devide by i=2533

# Bigram estimates of sentence probabilities

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

# Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

- Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain Rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

# Perplexity

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

# Perplexity

- Given training set
  - Compute uni-gram, bi-gram mode
- Test set:
  - Utilize previous step statistics to c

# Bayes Rules

- Here C presents class and d represents documents
- We compute conditional probability of the class given the document

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

# Naïve Bayes Classifier

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

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

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

Simplifying the equation by dropping denominator? Why?
Because for all class denominator should be same.

# How to estimate

- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

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

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

fraction of times word *wi* appears
among all words in documents of topic *cj*

# Solutions

- Laplace (add-1) smoothing

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

- Another Issue:
  - Unknown words in the test data.
    - Vocabulary did not occur in the train data in any class.
    - Solution: Remove them or ignore them while computing probability.
  - More frequent words – specially stop words
    - Best practice to remove them

# Naïve Bayes Summary

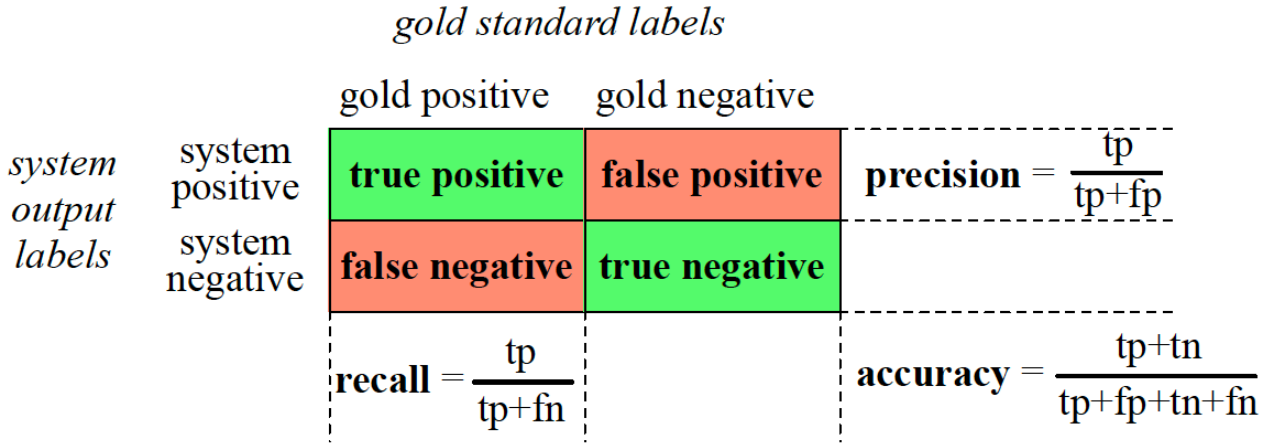
- Very Fast, low storage requirements
- Robust to Irrelevant Features

  Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features

- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification

# Evaluation Metrics

- Performance metrics - Precision, Recall, F measure
- Precision - the fraction of relevant instances among the retrieved instances
- Recall - fraction of relevant instances that were retrieved.



# Micro- vs. Macro-Averaging: Example

- **Macro averaging**: Compute performance for each class, then average.
- **Micro averaging**: Collect decisions for all classes, compute contingency table, evaluate.

**Class 1: Urgent**

| | true urgent | true not |
|---|---|---|
| system urgent | 8 | 11 |
| system not | 8 | 340 |

$$\text{precision} = \frac{8}{8+11} = .42$$

**Class 2: Normal**

| | true normal | true not |
|---|---|---|
| system normal | 60 | 55 |
| system not | 40 | 212 |

$$\text{precision} = \frac{60}{60+55} = .52$$

**Class 3: Spam**

| | true spam | true not |
|---|---|---|
| system spam | 200 | 33 |
| system not | 51 | 83 |

$$\text{precision} = \frac{200}{200+33} = .86$$

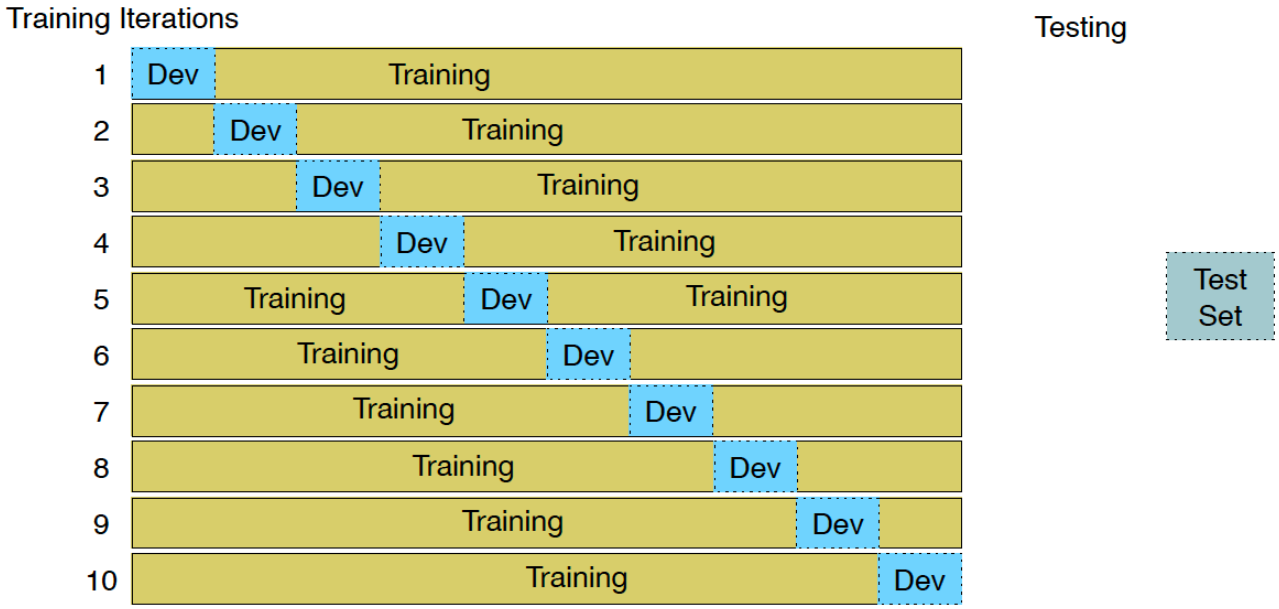
**Pooled**

| | true yes | true no |
|---|---|---|
| system yes | 268 | 99 |
| system no | 99 | 635 |

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

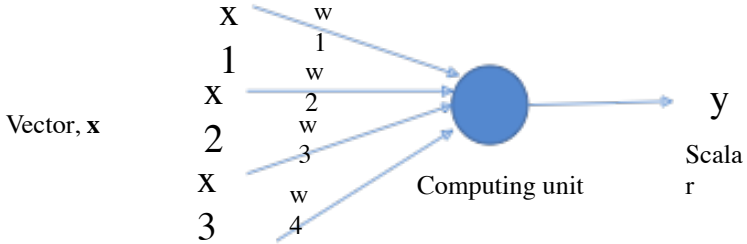
$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

# Cross-validation



# Logistic Regression

- Logistic regression is a discriminative classifier
  - It learns the task of separating classes using given features
  - Learn from a training set
    - Assigns weights (w) to features and also add bias term (intercept)
      - Forms vector
    - Decision during test
      - Learned weight multiplied with the feature to compute score




Score, $z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$

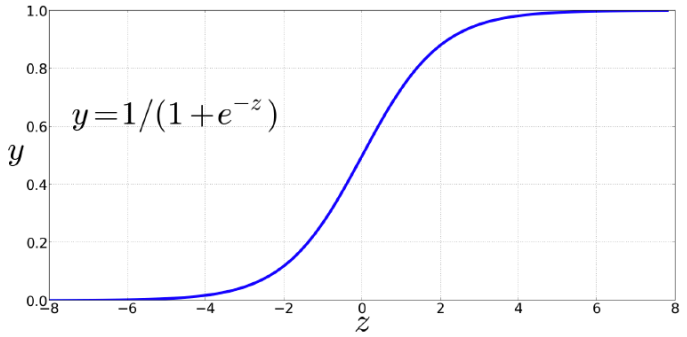
$z = w \cdot x + b$

dot product form.

# Logistic Regression

- We compute scores
- How can we assign probability?
  - Use sigmoid function for binary class
  -

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




Sigmoid function

Property of sigmoid function $1 - \sigma(x) = \sigma(-x)$

# Logistic Regression

- Two classes, p(y = 1) and p(y = 0), sum to 1.

$$
\begin{aligned}
P(y=1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1+\exp\left(-(w \cdot x + b)\right)} \\
P(y=0) &= 1 - \sigma(w \cdot x + b) \\
&= 1 - \frac{1}{1+\exp\left(-(w \cdot x + b)\right)} \\
&= \frac{\exp\left(-(w \cdot x + b)\right)}{1+\exp\left(-(w \cdot x + b)\right)}
\end{aligned}
$$

Decision :

$$
\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}
$$

# Generative vs Discriminative Classifiers

| Generative | Discriminative |
|---|---|
| Probabilistic | Provide a model for the target variable |
| Specify a joint probability distribution over observations and targets: P(c,d) | Use analysis of observed variables<br>Infer outputs based on inputs: P(cld) |
| Bayes rule enables a conditional distribution | Learn boundaries between classes |

# Vector semantics & Word Embedding

- Represents words as vectors such that each vector element corresponds to a different context
- Vector representation strongly associated with the given context
- Example:
  - How tall is Mt. Everest?
  - The official height of Mount Everest is 29029 feet
  - “tall” is similar to “height
- Compute semantic similarity of words as the similarity of the embedded vectors.

# Vector semantics & Word Embedding

- Vector semantics:
  - Standard way of represent word meaning in NLP
    - Represent word in a multidimensional semantic space that derived from the distributions of word neighbors.
- Word embedding:
  - Represents words as vectors such that each vector element corresponds to a different context
- Vector representation strongly associated with the given context
- Example:


two-dimensional (t-SNE) projection of embeddings for sentiment analysis

# TF-IDF

- IDF: inverse document frequency
  - Document frequency (*dft*) of a term t is the number of documents it occurs in
  - Idf represented as $\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$
  - $N$ – total number of documents
  - *dft* – number of documents in which term t occur.
- Tf-idf computed weighted value $wt,d$ for word t in document *d* thus combines term frequency $tft,d$

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

# Example: TF-IDF

- D1: *Text processing is necessary.*
- D2: *Text processing is necessary and important.*

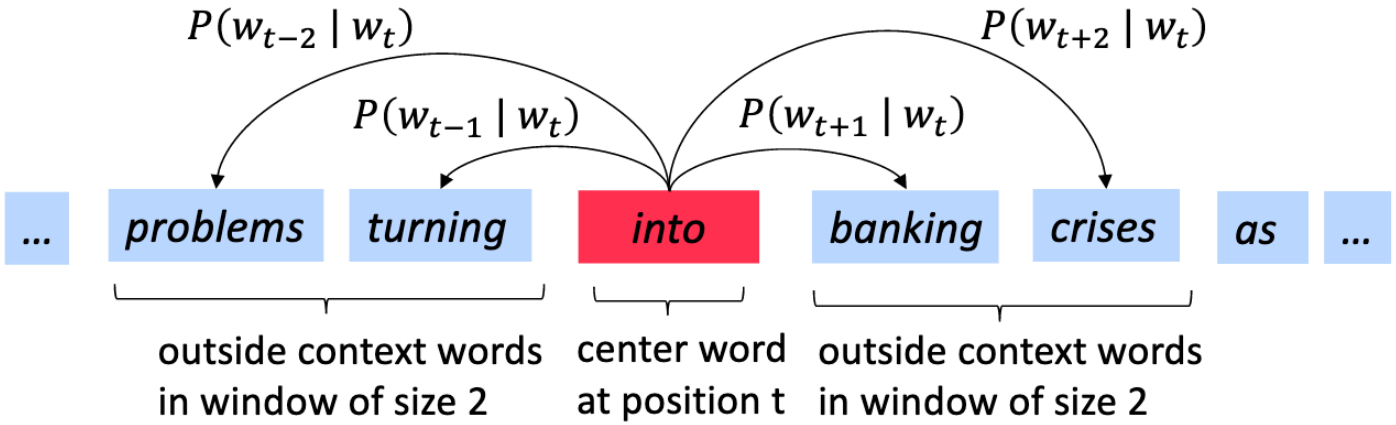
| Word | TF | | IDF | TF-IDF | |
|---|---|---|---|---|---|
| | D1 | D2 | | D1 | D2 |
| Text | 1/4 | 1/6 | Log(2/2)=0 | 0 | 0 |
| processing | 1/4 | 1/6 | Log(2/2)=0 | 0 | 0 |
| is | 1/4 | 1/6 | Log(2/2)=0 | 0 | 0 |
| necessary | 1/4 | 1/6 | Log(2/2)=0 | 0 | 0 |
| and | 0/4 | 1/6 | Log(2/1)=0.3 | 0 | 0.05 |
| important | 0/4 | 1/6 | Log(2/1)=0.3 | 0 | 0.05 |

# Word Embedding

- Word embedding is a way of representing words as vectors.
- Embedding: dense representation of words
  - Goal: goal of word embedding is to **convert the high dimensional feature space of words** into low dimensional feature vectors by preserving the contextual similarity in the corpus
  - We will introduce a dense vector for each word, chosen so that it is similar to vectors of words appearing in similar contexts.
- Popular pre-trained models to create word embedding of a text
  - Word2Vec — From Google
  - Fast text — From Facebook
  - Glove — From Stanford
- These models are called Prediction-based

# Main Idea of Word2Vec

- Iterate through each word of the whole corpus
  - Predict surrounding words using word vectors




$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

# Word2Vec

- Word2vec is a method to efficiently create word embeddings
- Algorithm: skip-gram with negative sampling called (SGNS)
- Word2vec are static embeddings
  - Fixed embedding for each word in the vocabulary

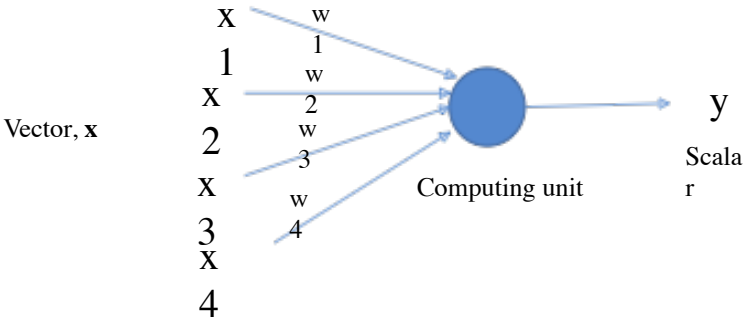
# Skip-gram model

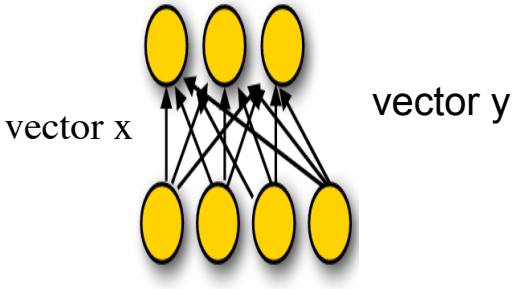
The intuition of skip-gram is:

- Treat the target word and a neighboring context word as positive examples.
- Randomly sample other words in the lexicon to get negative samples.
- Use logistic regression to train a classifier to distinguish those two cases.
- Use the learned weights as the embeddings.

# Neural Networks

- Neural network is a small computing units
  - Input as vector
  - Output – single value
- Building block of NN is single computational unit

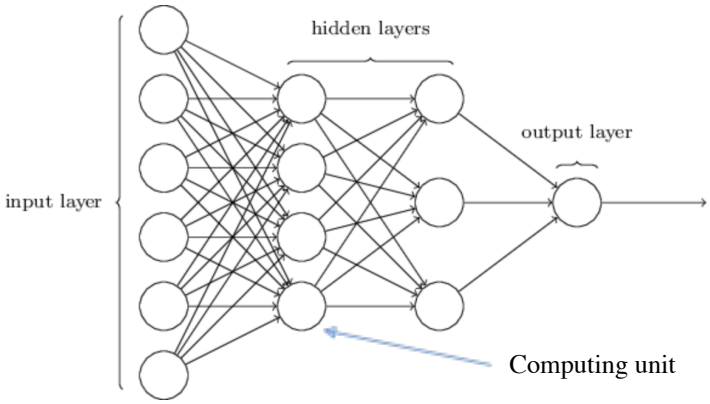


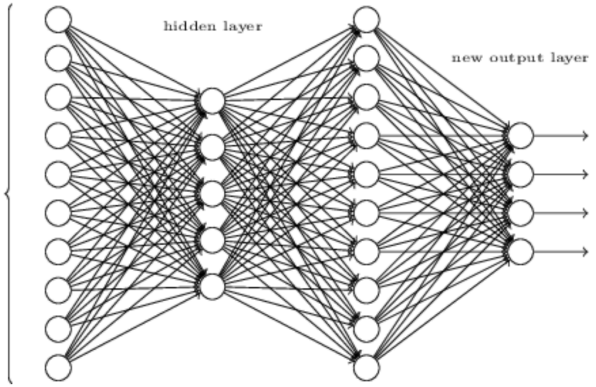

Neural Networks

# Feed-forward Neural Network (NN)

- Combine multiple units
- Feed output of one unit to another unit iteratively



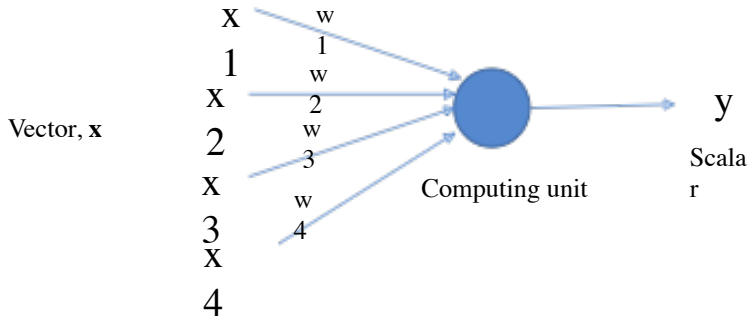

Single Output NN




Multiple Output NN

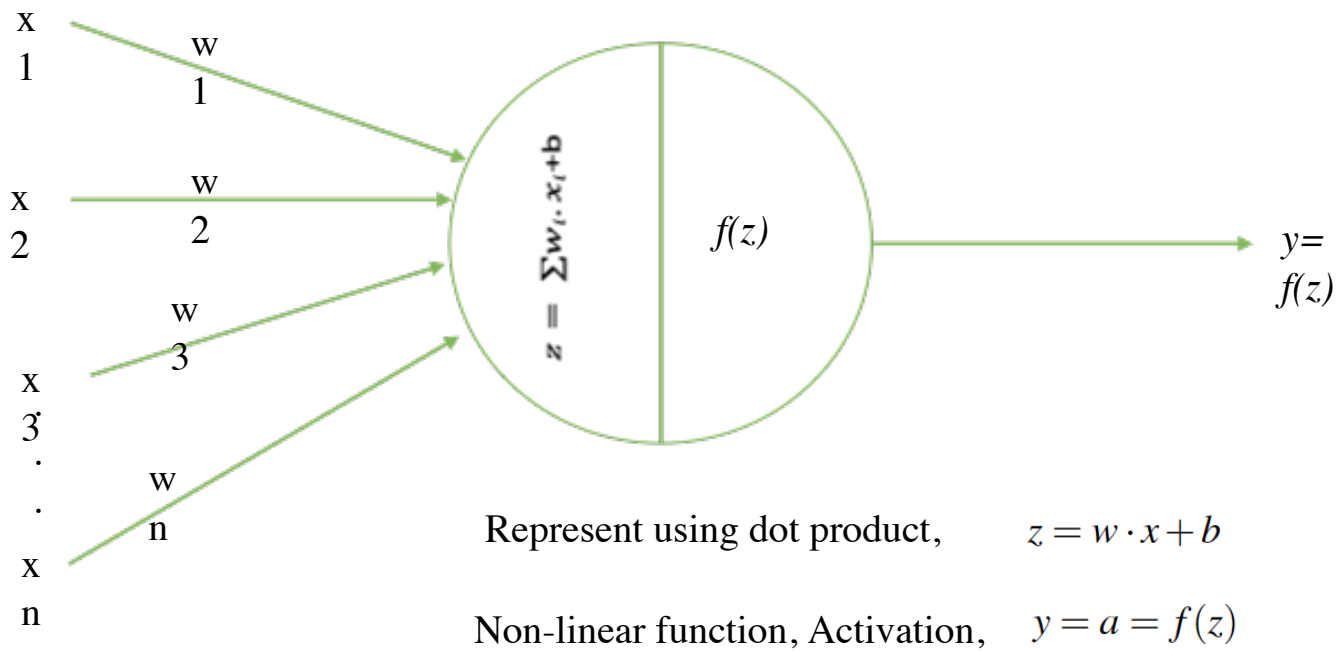
# Computing Unit

- Single computational unit
  - Input vector,
  - Weights,
  - Computing unit/functions
  - Output




- Takes real valued numbers as input → Perform Computation → output

# Computing Unit




Represent using dot product, $z = w \cdot x + b$

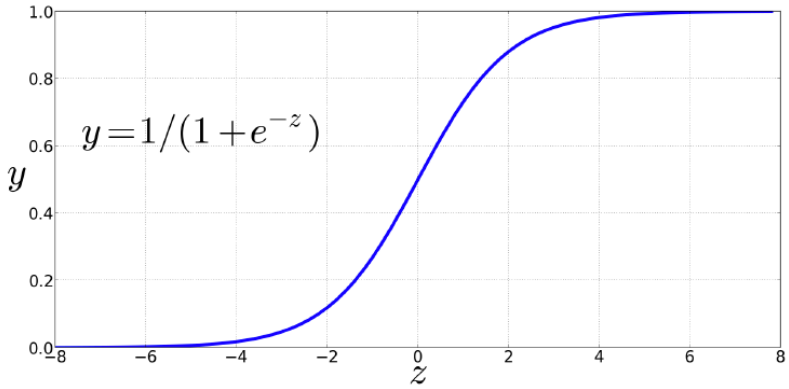
Non-linear function, Activation, $y = a = f(z)$

# Activation Function - sigmoid

- Popular activation functions – sigmoid, Relu, Tanh

**Sigmoid Characteristics:**

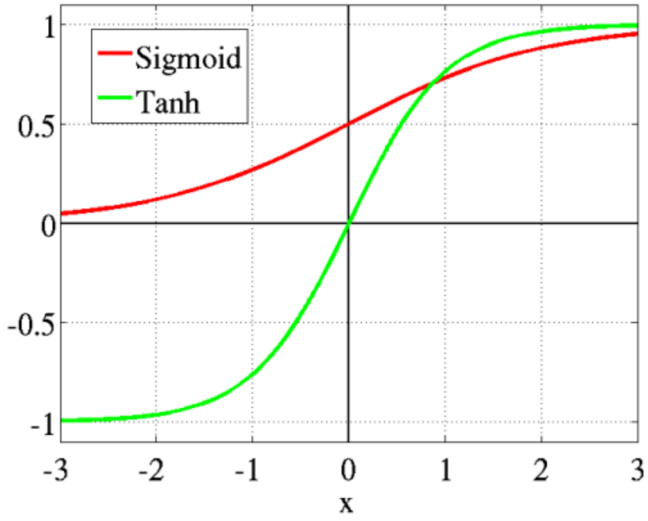
- S-shaped,
- Non-linear function
- Output range [0,1]
- Linear near 0
- Outlier values squashed towards 0 or 1
- Function is differentiable
- function is monotonic but function's derivative is not.
- This is the Non zero Centered Activation Function
- The model Learning rate is slow
- This function may cause a neural network to get stuck at the training time.
- Create a Vanishing gradient problem.




In the sigmoid or tanh functions, very high saturated values of z result in values of y that are saturated, i.e., extremely close to 1, and have derivatives very close to 0. Zero derivatives cause problems for learning.

# Activation Function - tanh

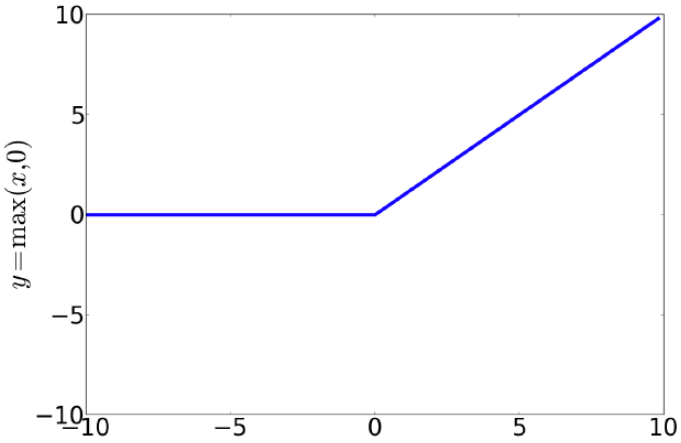
- Hyperbolic tangent activation function
- Range of tanh function is [-1, 1]
- S-shaped
- Non-linear
- Negative input mapped strongly negative
- Zero input mapped near zero in the graph
- Differentiable
- Tanh activation function is also known as the **zero centered activation function.**
  - the minimum range is -1
- function is **monotonic** while its **derivative is**




$$y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

# Activation Function - relu

- Linear activation
- function and its derivative **both are monotonic**.
- Ranges [0, infinity)
- Convert negative values to zero immediately
- Removed Vanishing Gradient Problem completely
  - Maximum Threshold values are Infinity
- Speed is fast compare to other activation function



# Details of the softmax classifier

- We can tease apart the prediction function into two steps:

  Take the $y^{th}$ row of $W$ and multiply that row with $x$:

$$W_{y.}x = \sum_{i=1}^{d} W_{yi}x_i = f_y$$

  Compute all $f_c$ for $c = 1, \ldots, C$

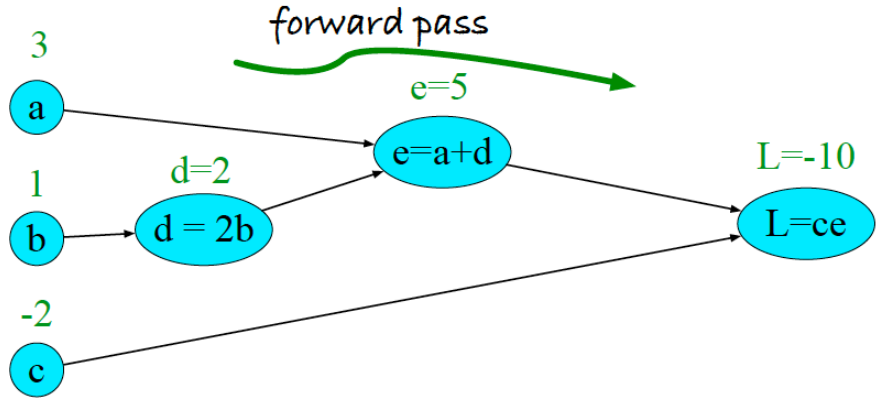
  Apply softmax function to get normalized probability:

$$p(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)} = \text{softmax}(f_y)$$

# Backpropagation Intuition

- Given equations:

$$d = 2 * b$$
$$e = a + d$$
$$L = c * e$$




- Computation graph:

- Chain rule (partial derivative):
  - f (x) = u(v(w(x)))

$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

# Backpropagation Intuition

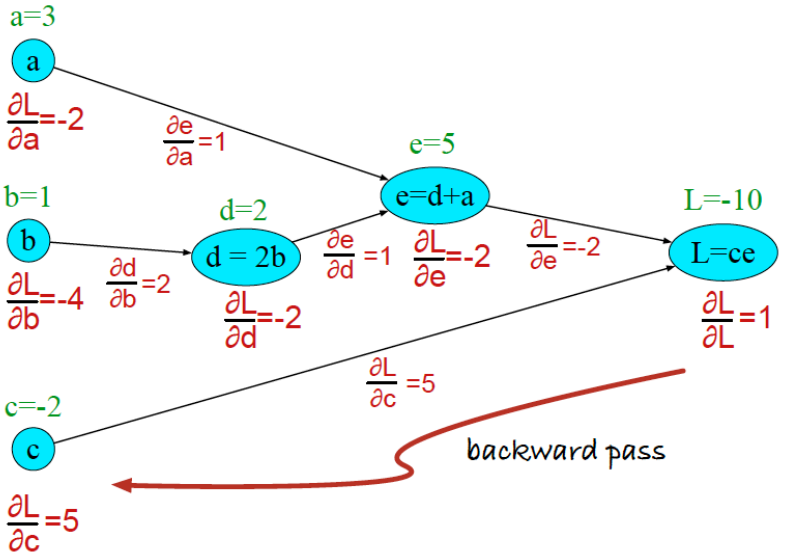
- Loss function
  - L(a,b,c) = c(a+2b) = ce

$$
\begin{aligned}
d &= 2*b \\
e &= a+d \\
L &= c*e
\end{aligned}
$$

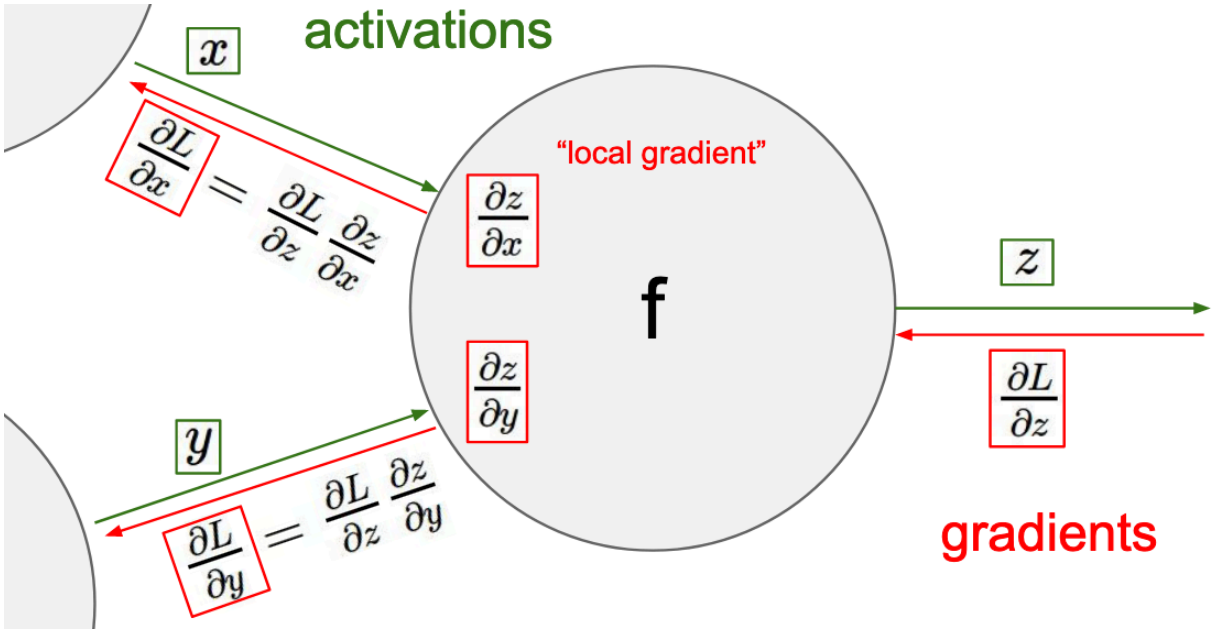
$$
\frac{\partial L}{\partial c} = e
$$

$$
\begin{aligned}
\frac{\partial L}{\partial a} &= \frac{\partial L}{\partial e}\frac{\partial e}{\partial a} \\
\frac{\partial L}{\partial b} &= \frac{\partial L}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial b}
\end{aligned}
$$

$$
\begin{aligned}
L = ce &: \quad \frac{\partial L}{\partial e} = c, \frac{\partial L}{\partial c} = e \\
e = a+d &: \quad \frac{\partial e}{\partial a} = 1, \frac{\partial e}{\partial d} = 1 \\
d = 2b &: \quad \frac{\partial d}{\partial b} = 2
\end{aligned}
$$

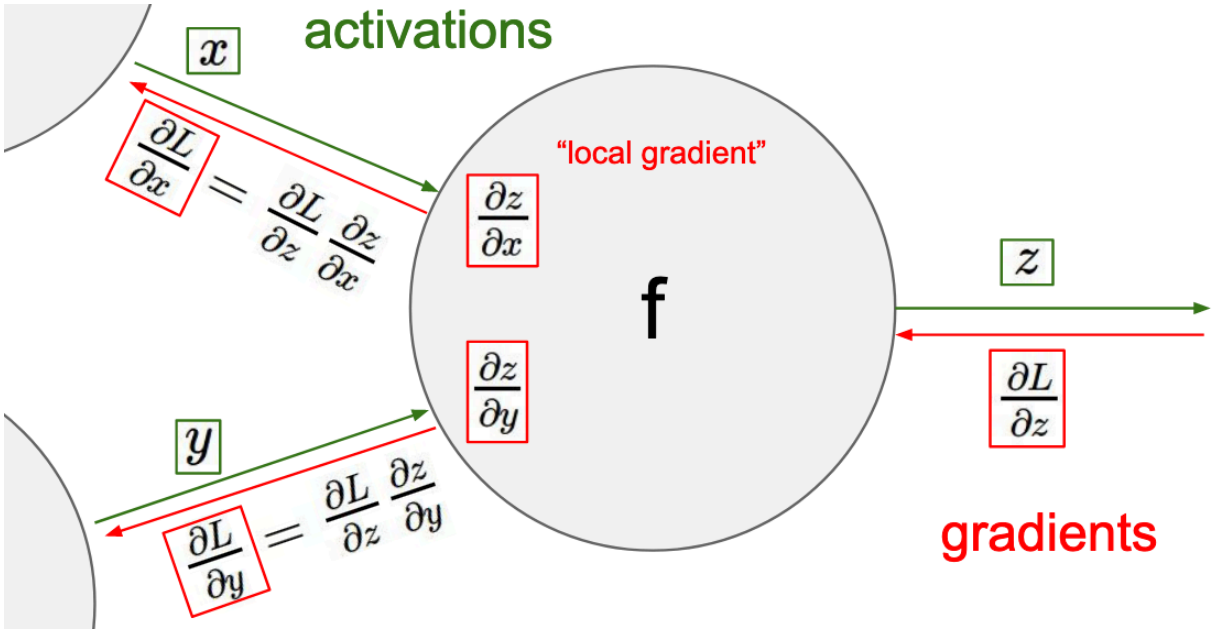


# Backpropagation - Intuition




Source: Fei-Fei Li & Andrej ~~Karpathy~~ & Justin Johnson

# Backpropagation - Intuition



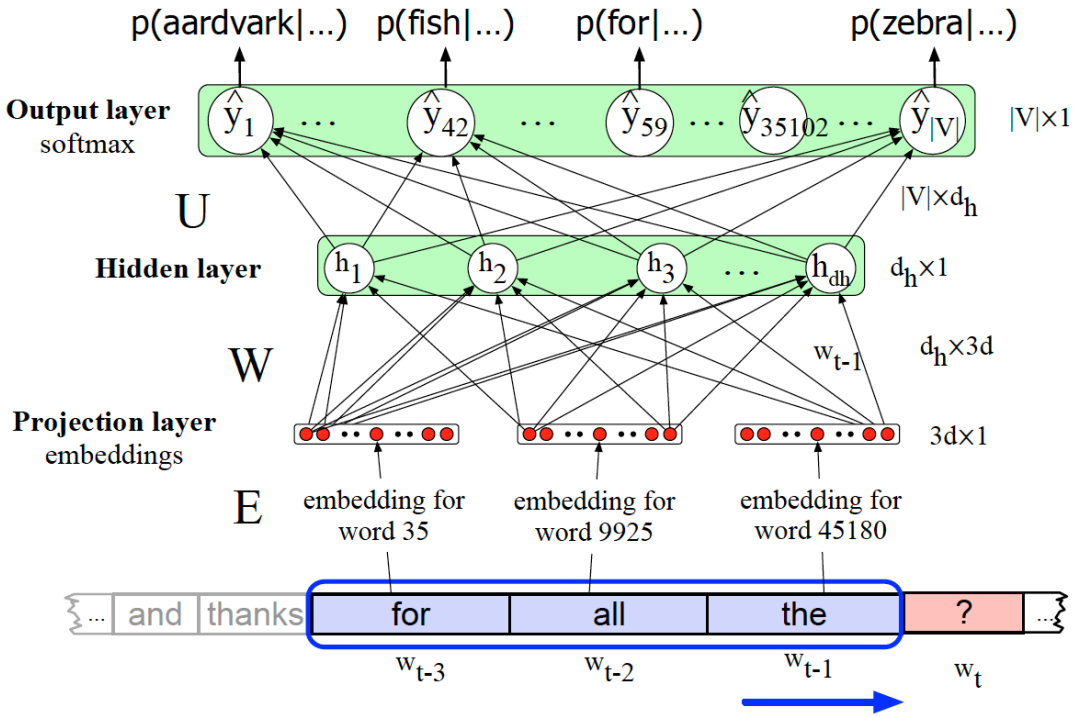

Source: Fei-Fei Li & Andrej Karpathy & Justin Johnson

# Neural Language Models

- Language Modeling: predicting upcoming words from prior word context
- Feedforward neural LM is a standard network
  - takes as input at time t a representation of some number of previous words
    - Wt-1; wt-2
- Feedforward neural LM approximates the probability of a word given the entire prior context $P(w_t|w_1 : t-1)$
- LM: $P(w_t|w_1,\ldots,w_{t-1}) \approx P(w_t|w_{t-N+1},\ldots,w_{t-1})$

# Neural Language Model

- The prior context is represented by embeddings of the previous words
- At each timestep t the network takes the 3 context words
- Converts each to a d-dimensional embedding
- Concatenates the 3 embeddings together to get the 1xN - d unit input layer x for the network.
- Units are multiplied by a




Assume embedding comes from word2vec

# NN Language Model: Learning word embedding

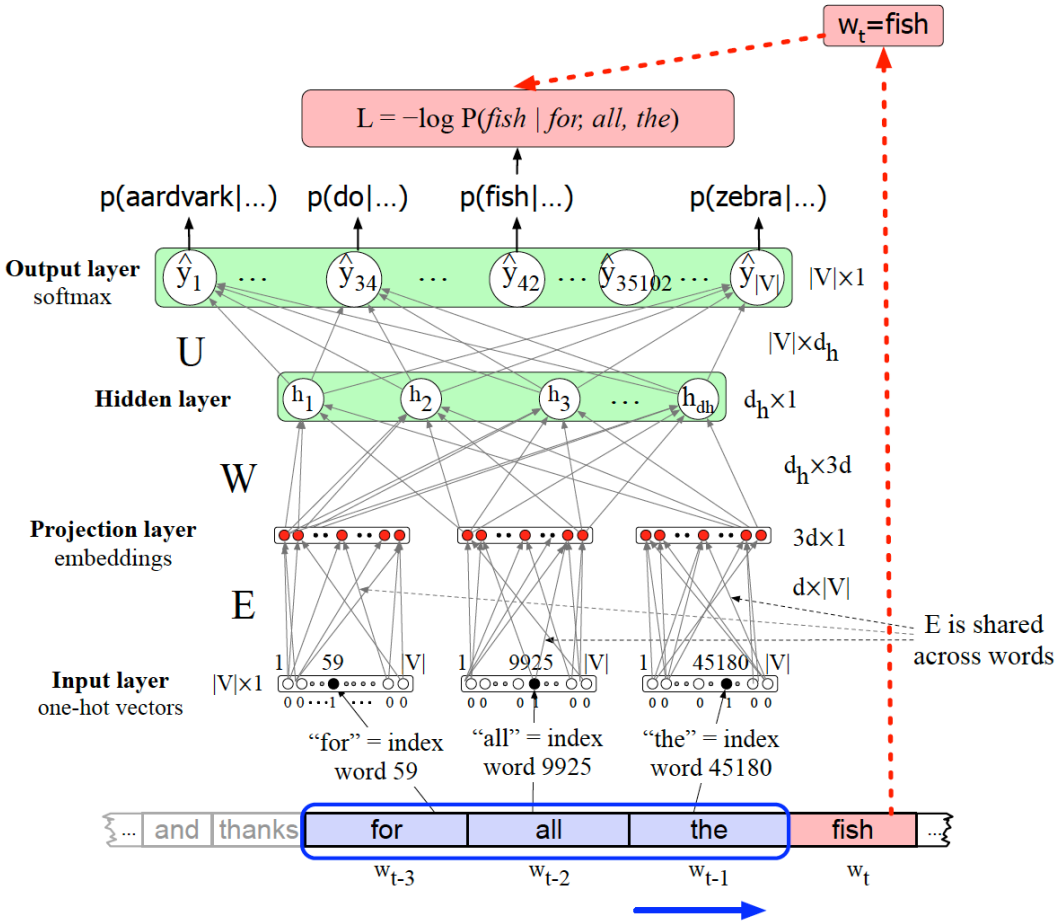
- N=3 context words
- One hot encoding of input words
- Single embedding dictionary E that's shared among these three

Language Model:

$$e = (Ex_1, Ex_2, ..., Ex)$$
$$h = \sigma(We + b)$$
$$z = Uh$$
$$\hat{y} = \text{softmax}(z)$$

Training: Back-pr

$$\theta = E, W, U, b$$



# Sequence Labeling

- Sequence Labeling is a task of assigning labels to each input sequence
  - Output sequence length is same as input sequence.
  - Algorithms: Hidden Markov Model (HMM), Conditional Random Field (CRF), Recurrent Neural Network (RNN)

- Two task falls in this category
  - **Parts of Speech (POS)** - noun, verb, pronoun, preposition, adverb, conjunction, participle, and article.
  - **Name Entity recognition** – proper names – multiword phrase
- Parts of Speech and Name entities are useful clues for sentence structure and meaning
  - Example: given a word noun, what is the likelihood of neighboring word.

# Parts of Speech Tagging

- A POS tag (or part-of-speech tag) is a special label assigned to each token (word) in a text corpus to indicate the part of speech
- A set of all POS tags used in a corpus is called a tagset
  - A tagset is a list of part-of-speech tags (POS tags for short)
  - Basic tagset includes- (N for noun, V for verb, A for adjective etc.)
- Two categories
  - Closed class, open class
- Closed classes are those with relatively fixed membership, such as prepositions— new prepositions are rarely coined. By contrast, nouns and verbs are open classes—new nouns and verbs like iPhone or to fax are continually being created or borrowed.

# Parts of Speech Tagging

- Closed class words are generally function
  - very short, occur frequently, and often have structuring uses in grammar
  - Example: like of, it, and, or you

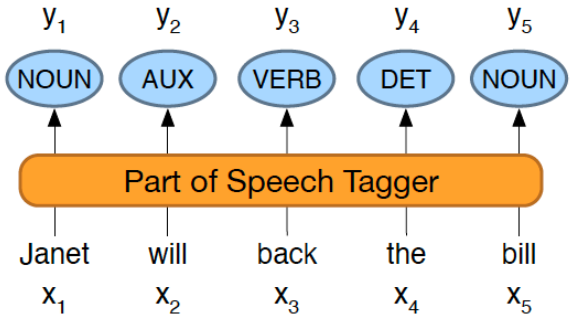
- Four major open classes
  - nouns (including proper nouns), verbs, adjectives, and adverbs,

# Example POS Tags

- **Sentence**: *There are 70 children there*.
- Tagging:
  - There/PRO/EX are/VERB/VBP 70/NUM/CD children/NOUN/NNS there/ADV/RB ./PUNC/.
  - [PRO, VERB, NUM, NOUN, ADV, PUNC] – UD
  - [EX, VBP, CD, NNS, RB, .] --- Penn Tree

# Parts of Speech tagging

- Part-of-speech tagging is the process of assigning a part-of-speech to each word in a text.
- **Input:-** a sequence x1, x2; …,xn of (tokenized) words and a tagset, and
- **Output:**- is a sequence y1,y2, …,yn of tags, each output yi corresponding exactly to one input xi,
- General:
  - words are ambiguous—have more than one possible part-of-speech
  - Tagging is a disambiguation task
    - Goal: find the correct tag for the context/situation
  - **Example**:



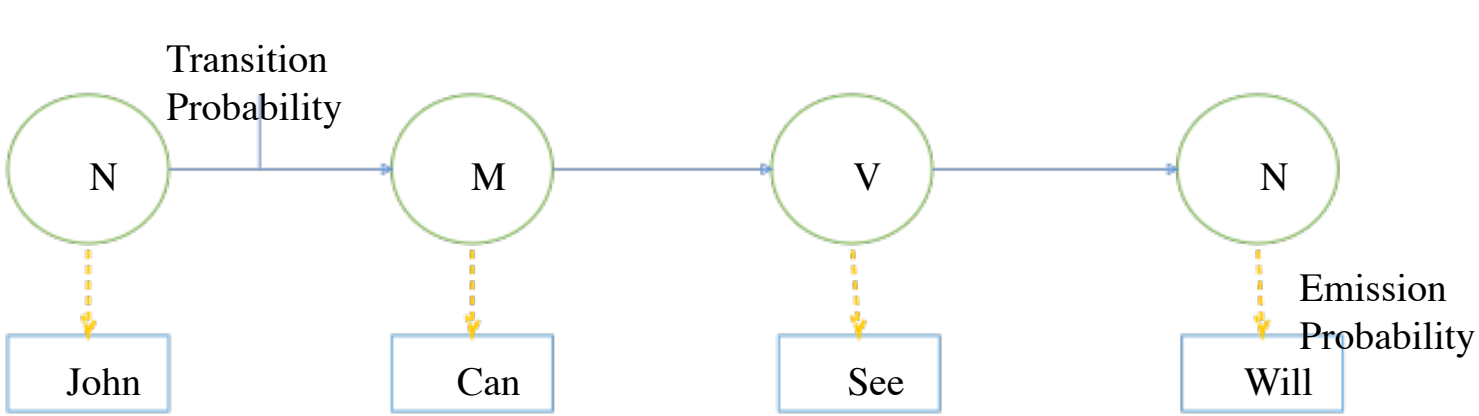
# POS - Ambiguity Resolution

- Ambiguous common words include
  - that, back, down, put and set;
- 6 different parts of speech for the word back
  1. earnings growth took a back/JJ seat
  2. a small building in the back/NN
  3. a clear majority of senators back/VBP the bill
  4. Dave began to **back/VB** toward the door
  5. enable the country to buy **back/RP** debt
  6. I was twenty-one **back/RB** then

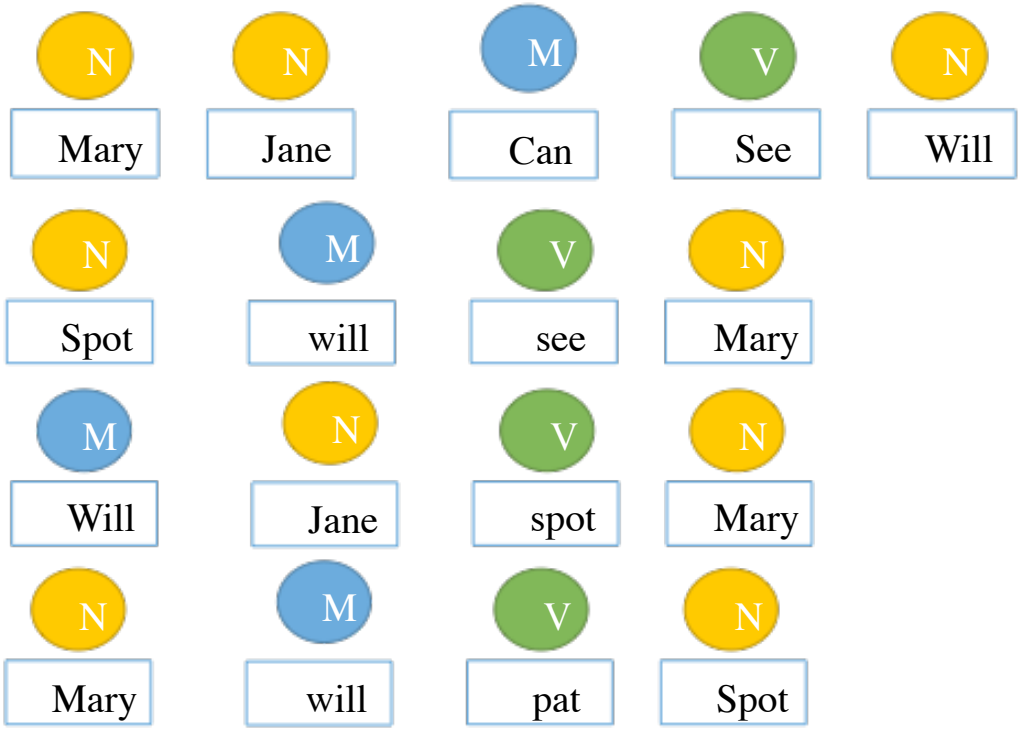
- Solution:
  - Baseline: given an ambiguous word, choose the tag which is most frequent in the training corpus.
  - How to create baseline?

# Hidden Markov Model (HMM)

- Hidden Markov Model (HMM) – stochastic process
  - Computes probability distribution over possible sequences of labels and chooses the best sequence
- Applications – reinforcement learning, temporal pattern recognition - speech, handwriting, gesture recognition, bioinformatics.



# Hidden Markov Model




**Example Sentences:**

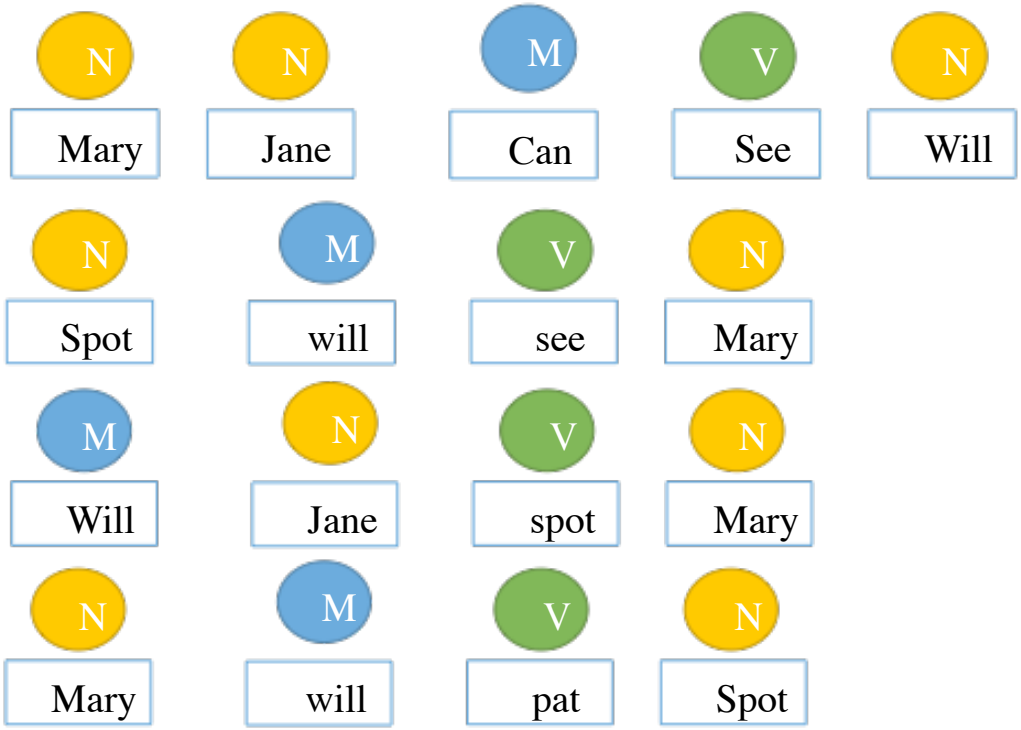
- Mary Jane can see Will
- Spot will see Mary
- Will Jane spot Mary?
- Mary will pat Spot

**How to get Emission Probability?**

# Hidden Markov Model

| Word | N | M | V |
|---|---|---|---|
| Mary | 4 | 0 | 0 |
| Jane | 2 | 0 | 0 |
| Will | 1 | 3 | 0 |
| Spot | 2 | 0 | 1 |
| Can | 0 | 1 | 0 |
| See | 0 | 0 | 2 |
| pat | 0 | 0 | 1 |
| | 9 | 4 | 4 |

**Step 1: Frequency Count**



# Hidden Markov Model

| Word | N | M | V |
|---|---|---|---|
| Mary | 4 | 0 | 0 |
| Jane | 2 | 0 | 0 |
| Will | 1 | 3 | 0 |
| Spot | 2 | 0 | 1 |
| Can | 0 | 1 | 0 |
| See | 0 | 0 | 2 |
| pat | 0 | 0 | 1 |
| | **9** | **4** | **4** |

| Word | N | M | V |
|---|---|---|---|
| Mary | 4/9 | 0 | 0 |
| Jane | 2/9 | 0 | 0 |
| Will | 1/9 | 3/4 | 0 |
| Spot | 2/9 | 0 | 1/4 |
| Can | 0 | 1/4 | 0 |
| See | 0 | 0 | 2/4 |
| pat | 0 | 0 | 1/4 |
| | **9** | **4** | **4** |

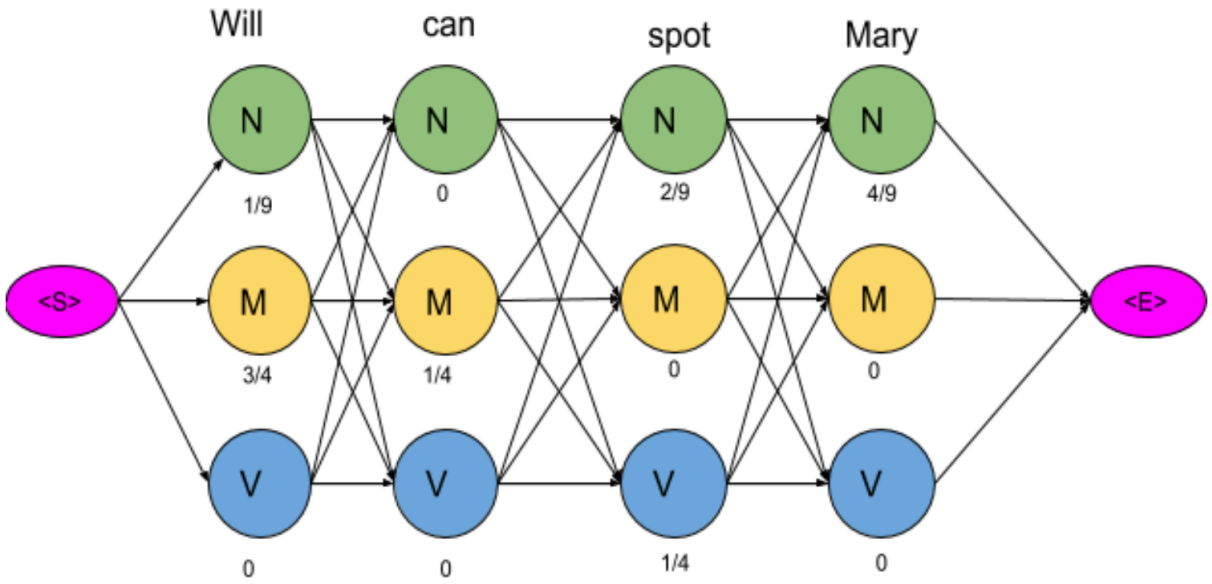
**Step 2: Normalization (column wise) --> emission probability**

# Tagging using HMM

- Tagging goal: find the the tag sequence which has maximum likelihood.
  - Step 1: Find all possible tag sequences for the given word sequence
  - Step 2: Delete the vertices and edges that has zero probability (optimization)
  - Step 3: Compute likelihood for the remaining sequences
  - Step 4: Choose the tag sequence which has maximum likelihood.

# HMM Tagging

Step 1: All Possible tag Sequences




Note: For simplicity, we are omitting transition probability along the edges

Transition Probability

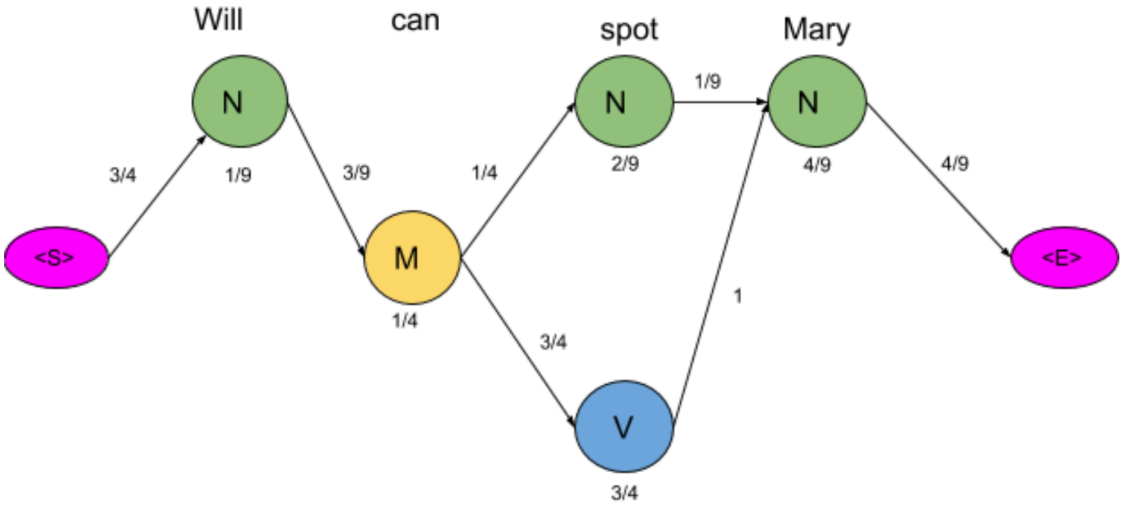
| | N | M | V | <E> |
|---|---|---|---|---|
| <S> | 3/4 | 1/4 | 0 | 0 |
| N | 1/9 | 3/9 | 1/9 | 4/9 |
| M | 1/4 | 0 | 3/4 | 0 |
| V | 4/4 | 0 | 0 | 0 |

Emission Probability

| Word | N | M | V |
|---|---|---|---|
| Mary | 4/9 | 0 | 0 |
| Jane | 2/9 | 0 | 0 |
| Will | 1/9 | 3/4 | 0 |
| Spot | 2/9 | 0 | 1/4 |
| Can | 0 | 1/4 | 0 |
| See | 0 | 0 | 2/4 |
| pat | 0 | 0 | 1/4 |

# HMM Tagging

- Compute likelihood for the remaining tag sequences




<S>→N→M→N→N→<E> = **3/4*1/9*3/9*1/4*1/4*2/9*1/9*4/9*4/9=0.00000846754**

<S>→N→M→N→V→<E> = **3/4*1/9*3/9*1/4*3/4*1/4*1*4/9*4/9 = 0.00025720164**

# Name Entity Recognition (NER)

- The task: find and classify names in text, for example:

  The European Commission [ORG] said on Thursday it disagreed with German [MISC] advice. Only France [LOC] and Britain [LOC] backed Fischler [PER] 's proposal .

  “What we have to be extremely careful of is how other countries are going to take Germany 's lead”, Welsh National Farmers ' Union [ORG] ( NFU [ORG] ) chairman John Lloyd Jones [PER] said on BBC [ORG] radio .

- Possible purposes:
  - Tracking mentions of particular entities in documents
  - For question answering, answers are usually named entities
  - A lot of wanted information is really associations between named entities
  - The same techniques can be extended to other slot-filling classifications

# NER on word sequences

We predict entities by classifying words in context and then extracting entities as word subsequences

| | | |
|---|---|---|
| Foreign | ORG | B-ORG |
| Ministry | ORG | I-ORG |
| spokesman | O | O |
| Shen | PER | B-PER |
| Guofang | PER | I-PER |
| told | O | O |
| Reuters | ORG | B-ORG |
| that | O | O |
| : | : | 👆 BIO encoding |

# Why NER is Hard?

- Hard to work out boundaries of entity

  First National Bank Donates 2 Vans To Future School of Fort Smith

  - Is the first entity “First National Bank” or “National Bank”
- Hard to know if something is an entity
  - Is there a school “Future School” or is it a future school?
- Hard to know class of unknown/novel entity:
  - *To find out more about Zig Ziglar and read features by other Creators Syndicate writers*
- What class is “Zig Ziglar”? (A Person)
- Entity class is ambiguous and depends on context
  - Where Larry Ellison and Charles Schwab can live discreetly amongst wooded estates.

# Summary Topics

- Regular Expression
- Basic Text Processing
- Language Modeling –
  - Building our own LM
  - Evaluate LM
- Classifiers – Naïve Bayes, Logistic regression, Softmax classifier
- Classifier Evaluation
  - Precision, Recall, F1, Accuracy
  - Micro and macro average, etc.
- Word Embedding – TF-IDF, Word2Vec, Skip-gram, NN-Language Model
- Neural Network – basics, backpropagation intuition