Frontiers in NLP I

From BoW to Word2Vec

Agenda

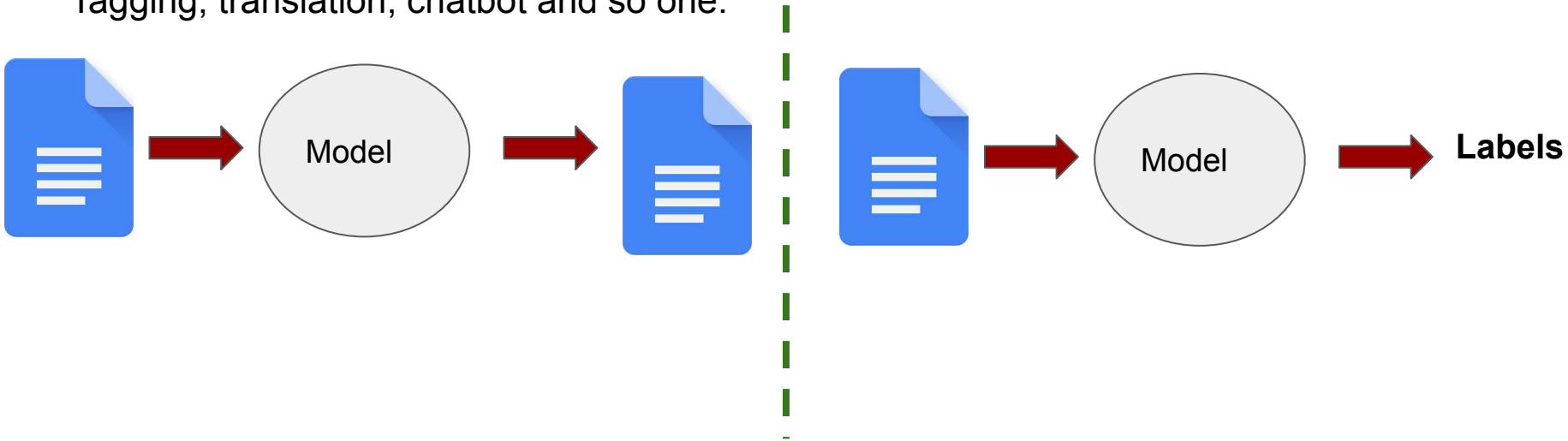
1. NLP and Its Tasks
2. Representation Learning
3. Word Embeddings
4. Neural Networks for NLP
    a. CNN
    b. RNN

NLP and Its Tasks

Natural Language Processing

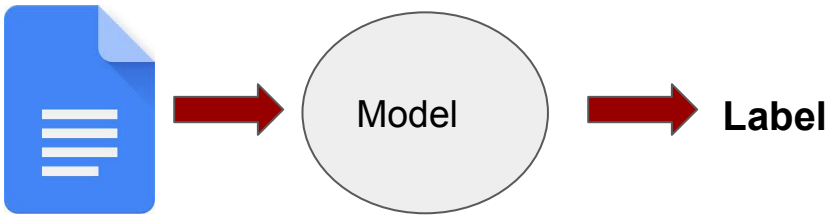
- **NLP** is a subfield of **linguistics**, **computer science**, and **artificial intelligence** concerned with the interactions between computers and **human language**, in particular how to program computers to process and analyze large amounts of natural language data.

- In NLP, there are various NLP tasks such as sentiment analysis, search engine, POS Tagging, translation, chatbot and so one.



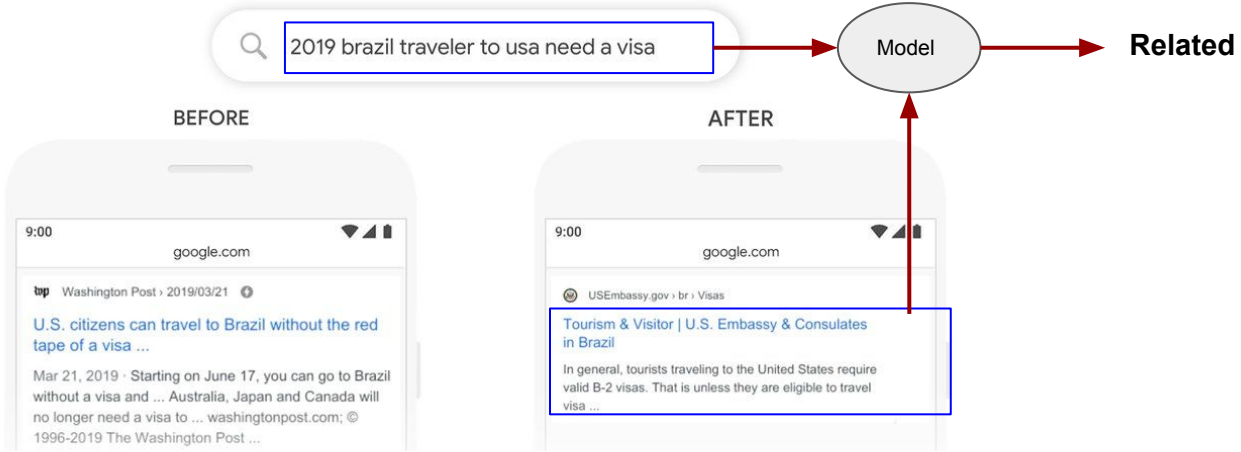
Sentiment Analysis

- The movie: batman begins was spectacular.
- The movie: batman begins was borning in the middle.
- Even the movie: batman begins was boring in the middle, it was spectacular.
- Even the movie: batman begins was spectacular, it was boring in the middle

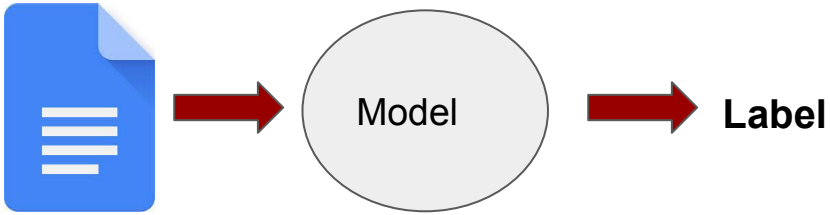


One Sequence of Words

Search Engine



https://www.blog.google/products/search/search-language-understanding-bert/



Two Sequences of Words

Part-of-Speech (POS) Tagging

- Label each word in a sentence with a part-of-speech (e.g. Verb, Adjective, Noun).

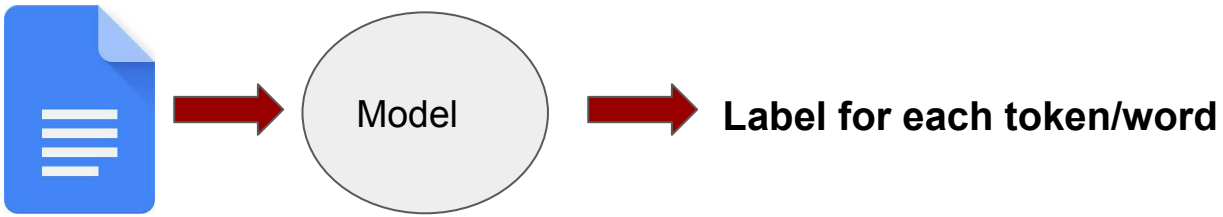
Your query

My dog also likes eating sausage.

Tagging

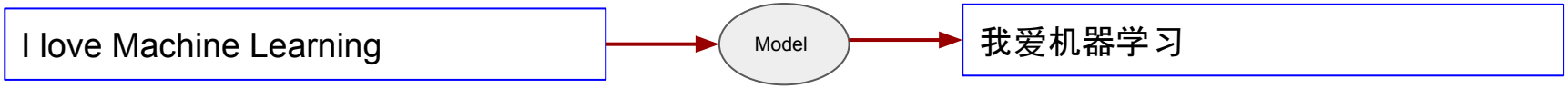
```
My/PRP$  dog/NN  also/RB  likes/VBZ  eating/VBG  sausage/NN  ./.
```

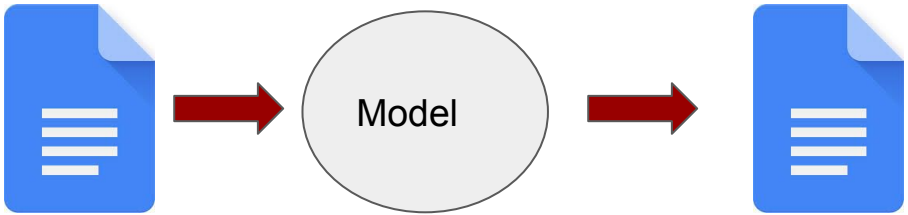
http://nlp.stanford.edu:8080/parser/index.jsp



One Sequence of Words

Machine Translation



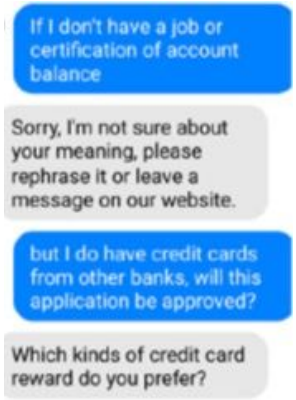


One Sequences of Words

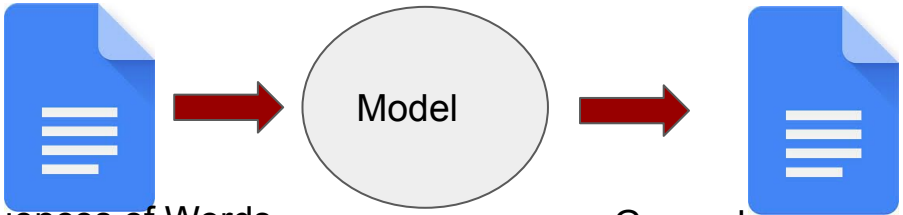
General Sequences of Words

Dialogue Systems

Chatting



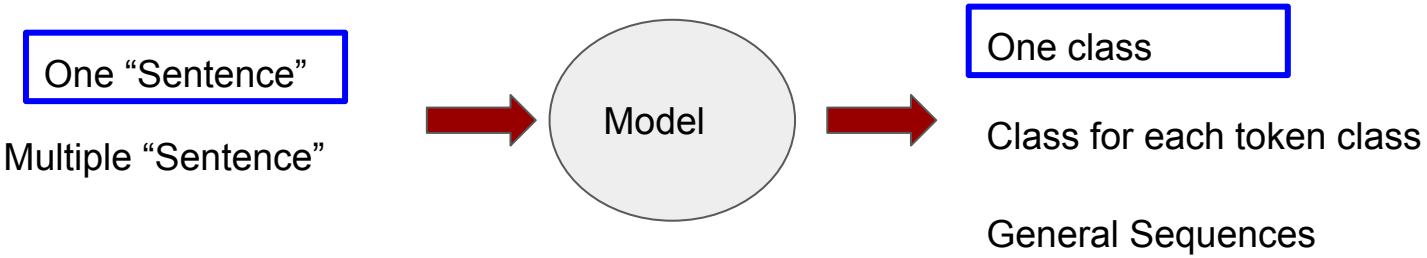
Task-oriented



Several Sequences of Words

General Sequences of Words

NLP Tasks



In practice, most of NLP tasks are **text classification.**

Fraudulent Financial Statement

One “Sentence” → Model → 1 or 0

UNITED STATES
SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549

FORM 10-K

(Mark One)

ANNUAL REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934
For the fiscal year ended December 31, 2016

or

TRANSITION REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934
For the transition period from to

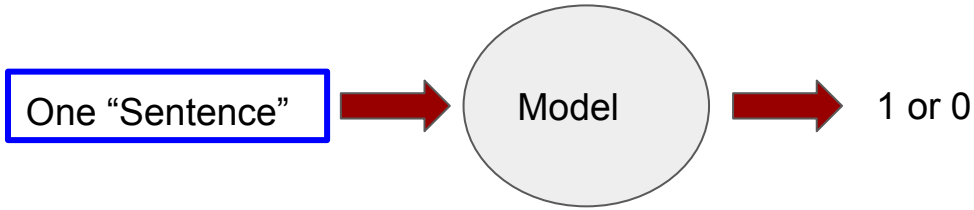
Commission File No. 000-22513

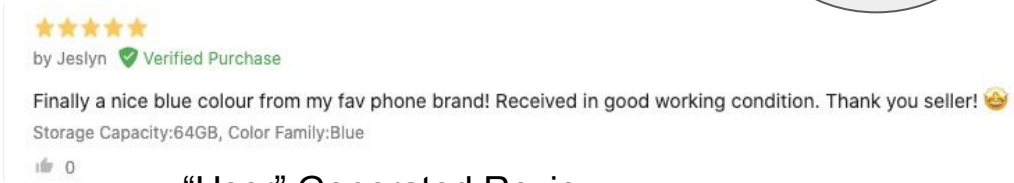
AMAZON.COM, INC.
(Exact name of registrant as specified in its charter)

SEC 10-k

Fraudulent OR Not?

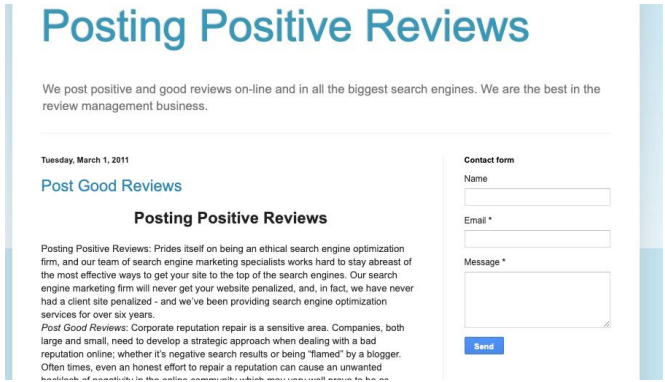
Fake Review Detection





“User” Generated Review

True or Fake



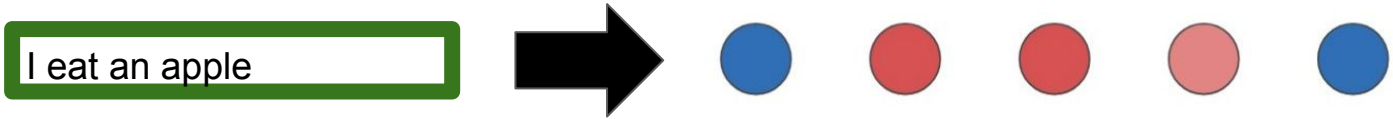
Representation Learning

Key Task: Representation Learning

- We need to develop systems that read and understand text the way a person does, by forming a representation of the text, and other context information that humans create to understand a piece of text.

Kevin Gimpel

Computational linguist



Text

Numerical Representation

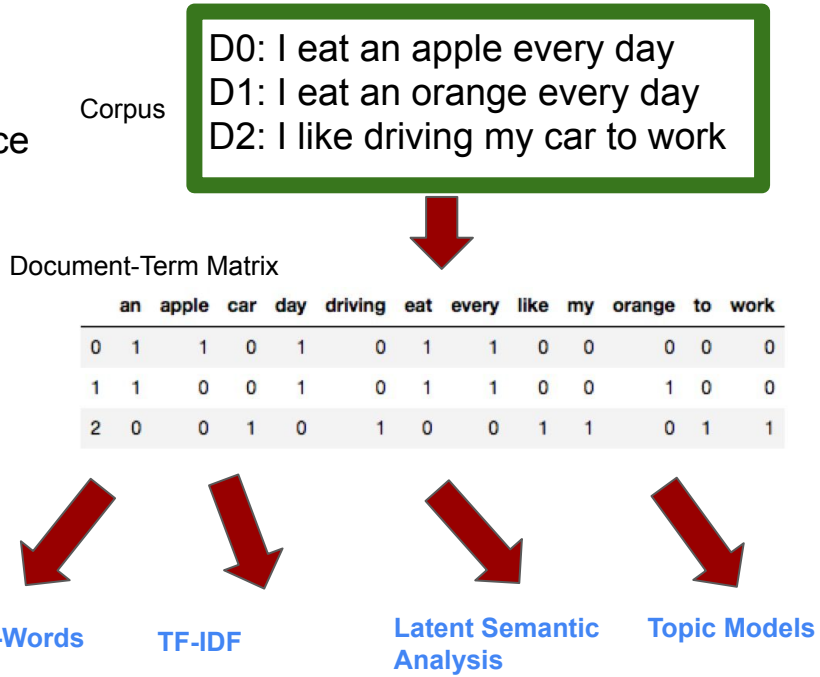
The learned representation should capture high-level semantic and syntactic information.

History of NLP

- Now, neural nlp models are able to achieve state-of-arts results in all tasks.
- Before neural nlp:
    - Symbolic NLP: rule-based system (derived from linguistic)
    - Statistical NLP: data-driven and use statistical methods

Statistical NLP

- Starting from Document-Term Matrix
  - It contains the co-occurrence information
  - Bag-of-Words: n-gram as features
  - TF-IDF: frequency of words to measure importance
  - Matrix Decomposition:
    - SVD -> Latent Semantic Analysis
    - Probabilistic model-> Topic Model



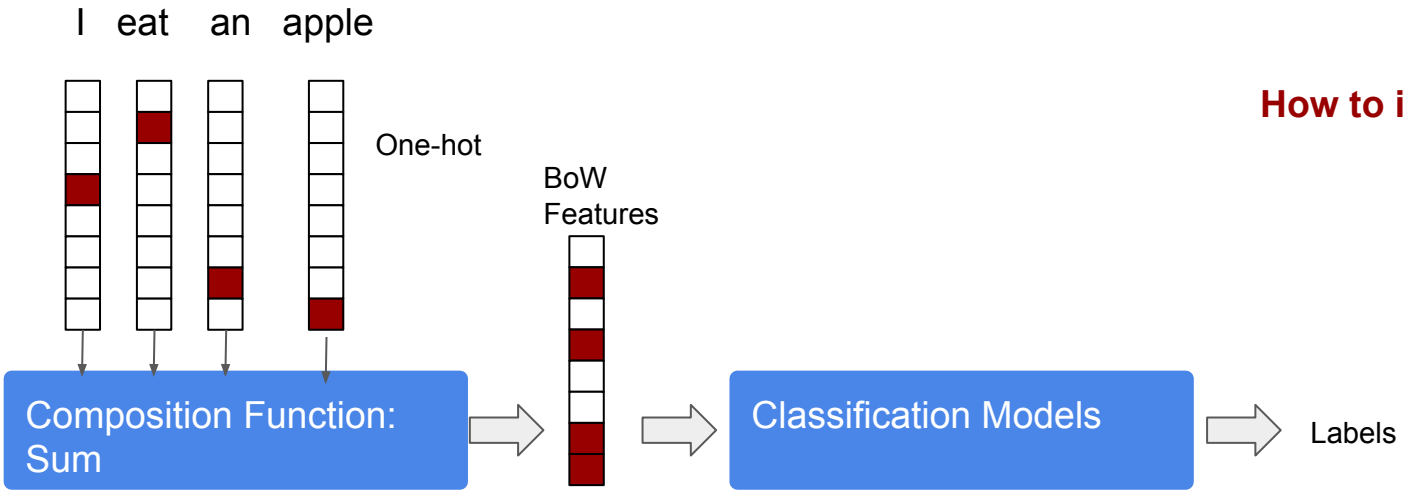
Limitations of Document-Term Matrix

- Too strong assumption: all words are independent of each other
    - | *orange* - *peach* | < | *orange* - *car* |
- Can not capture the order information in the sequence
- High dimensionality due to large size of vocabulary

	an	apple	car	day	driving	eat	every	like	my	orange	to	work
0	1	1	0	1	0	1	1	0	0	0	0	0
1	1	0	0	1	0	1	1	0	0	1	0	0
2	0	0	1	0	1	0	0	1	1	0	1	1

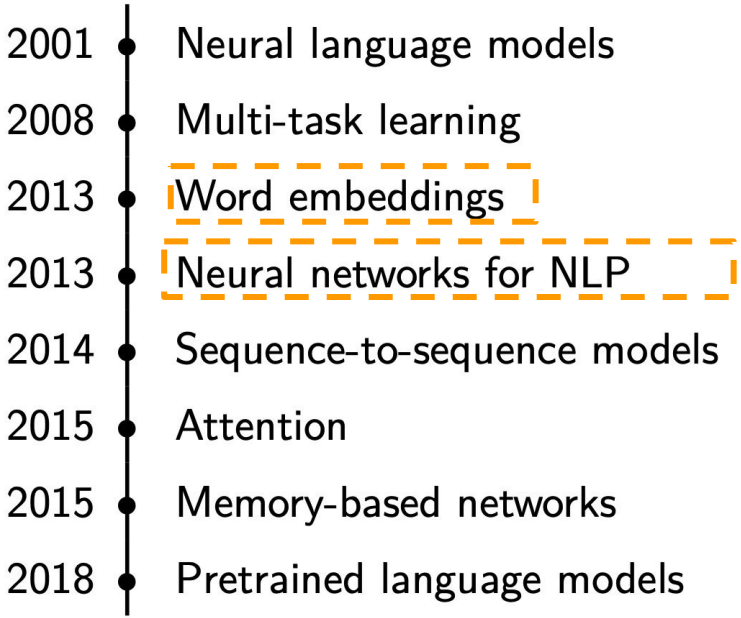
A New Perspective on BoW

- Each word in vocab is represented in one-hot embedding
- Sum one-hot vectors of the words in a sentence
- The final vector is the representation for the given sentence and then fed into a classifier.



How to improve?

Neural NLP



https://www.kamperh.com/slides/ruder+kamper_indaba2018_talk.pdf

Word Embeddings

Word Representation

- How to represent word in a vector space

apple [0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

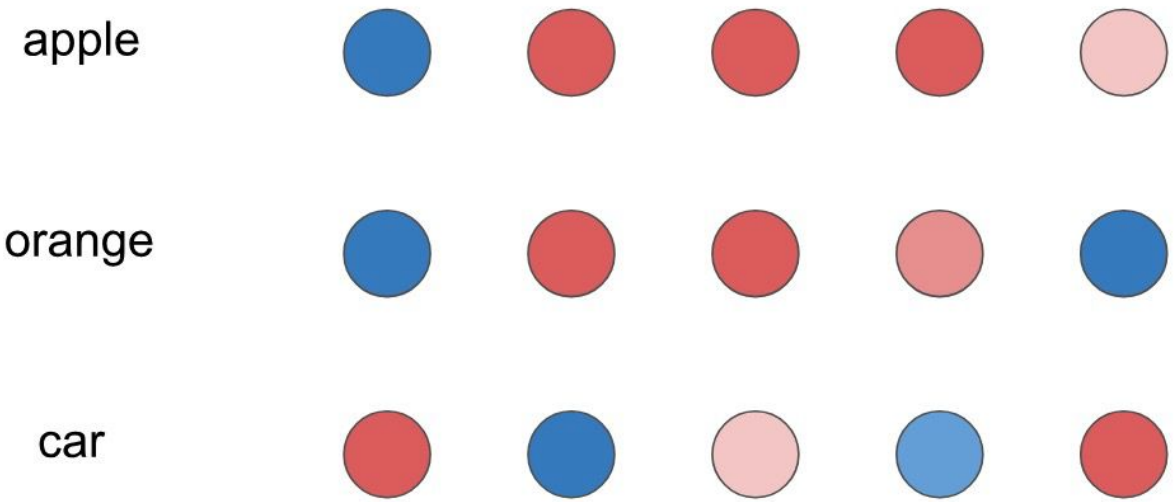
orange [0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 ... 0 0 0 0 0 0]

car [0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

One-hot Vector

Distributed Representation

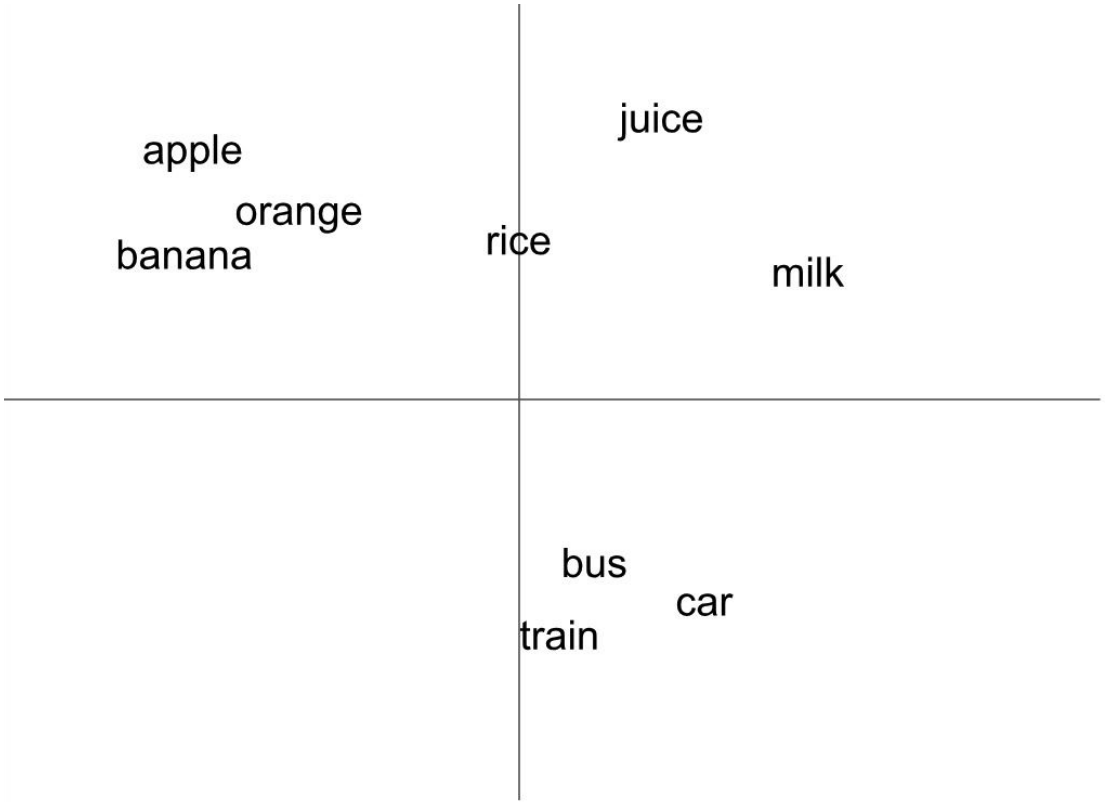
- Words should be encoded into a **low-dimensional** and **dense** vector



Word Vectors

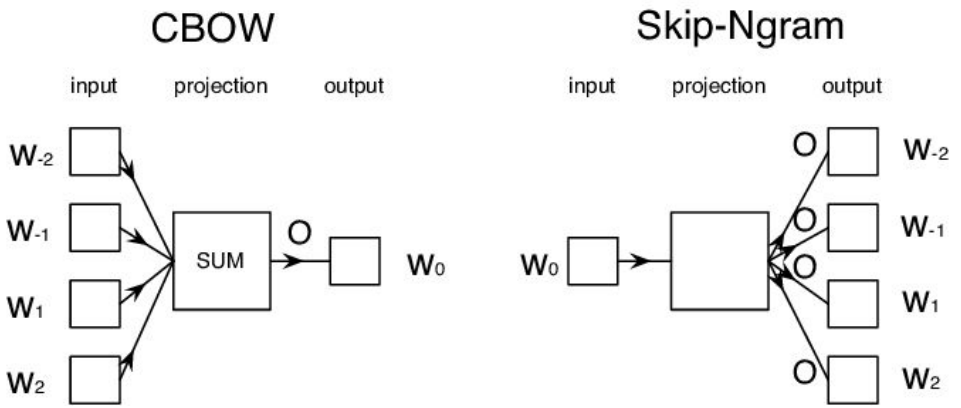
Project word vectors in a two-dimensional space. And visualize them!

Similar words are close to each other.

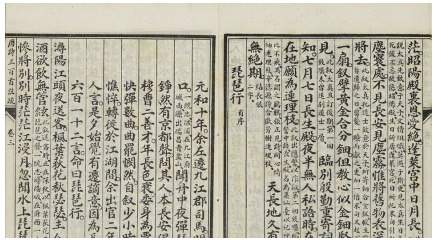


Word2Vec

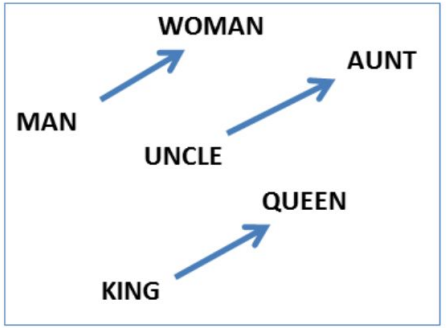
- A method of computing vector representation of words developed by Google.
- Open-source version of Word2Vec hosted by Google (in C)
- Train a simple neural network with a single hidden layer to perform word prediction tasks
- Two structures proposed **Continuous Bag of Words (cbow)** vs **skip-gram**:



Word2Vec as BlackBox



input, output



Corpus **Word2Vec Tool** **Word Embeddings**

A Good Visualization for Word2Vec

https://ronxin.github.io/wevi/

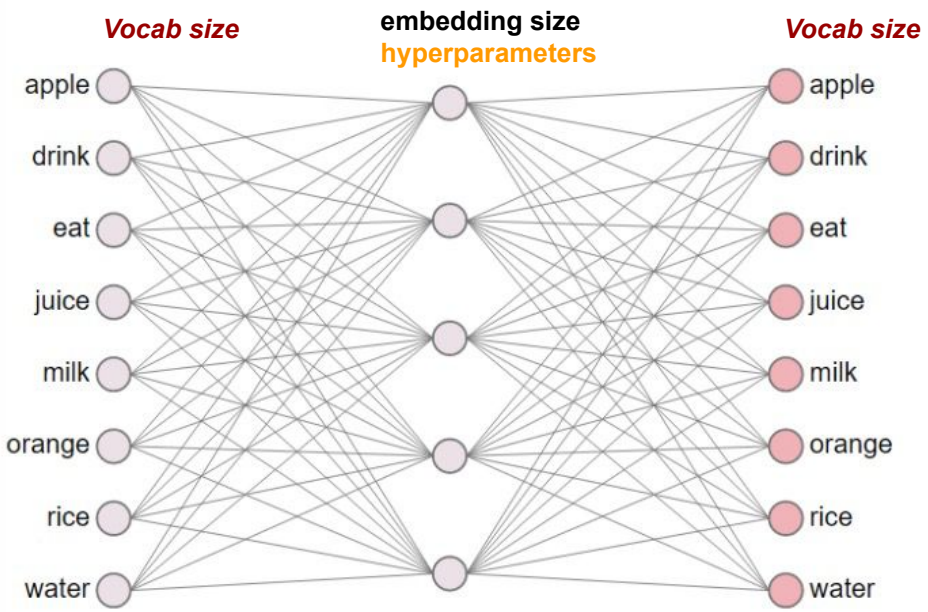
Target

- Given a training corpus, we prepare a list of N (input_word, output_word).
- Objective Function: Maximize probability of all the output words given the corresponding input words.

$$\mathbf{J}(\theta) = \prod_{i=1}^{N} p(w^{i}_{output} | w^{i}_{input}, \theta)$$

Neural network parameters that will be optimized

Model Architecture



Structure Highlights:

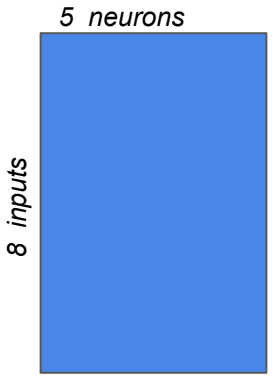
- input layer
  - one-hot vector
- hidden layer
  - linear (identity)
- output layer
  - softmax

From Xin Rong 2016

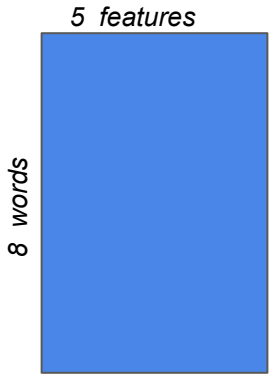
Hidden Layer

- **linear-activation** function here
- **5** neurons are the word vec. dimensions
- This layer is operating as a ‘lookup’ table
- Input word matrix denoted as **IVec**

Hidden Layer Weights Matrix



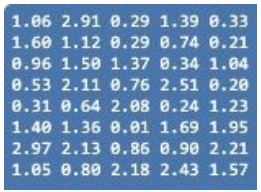
Word Vector Look Up Table



One-hot vector

[0,0,1,0,0,0,0, 0]

Index of eat



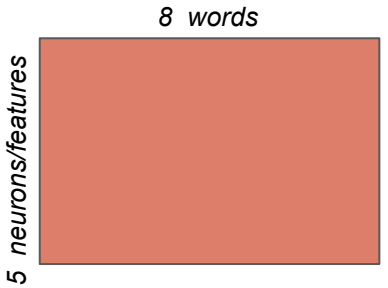
Word vector for “eat”

0.96, 1.5, 1.37, 0.34, 1.04

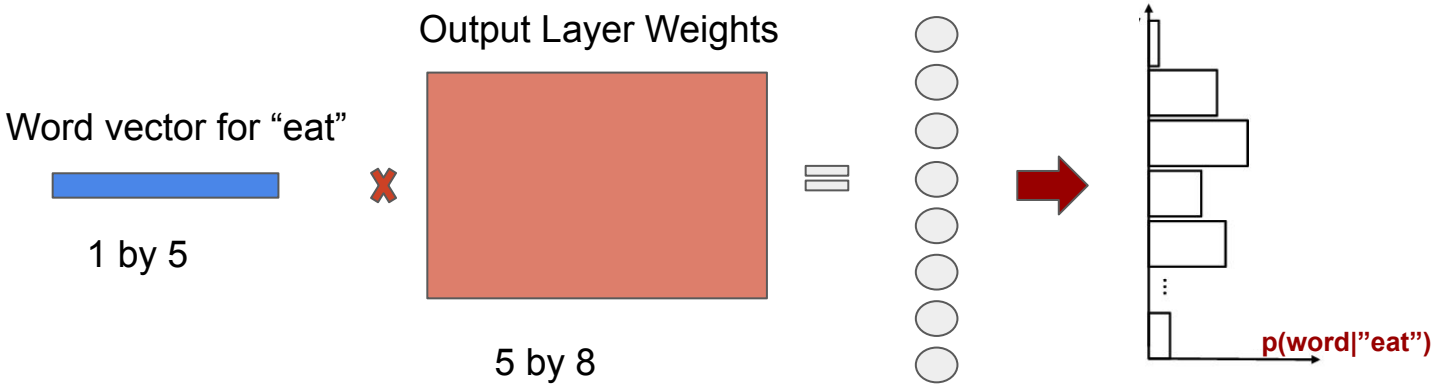
This is a **projection/look up** process: given the index of the word, we take the ith row in the word vector matrix out

Output Layer

Output Layer Weights Matrix
A.K.A Output word vectors



- Softmax classifier
- Output word matrix denoted as **OVec**

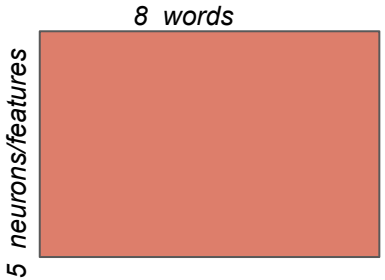


Scores over 8 words

Output Layer

- Softmax classifier
- Output word matrix denoted as **OVec**

Output Layer Weights Matrix
A.K.A Output word vectors



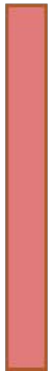
Output weights for “apple”

e^(IVec[eat]*OVec[apple])

Word vector for “eat”

5 features

×



5 features

softmax

$$\frac{e^x}{\sum e^x}$$

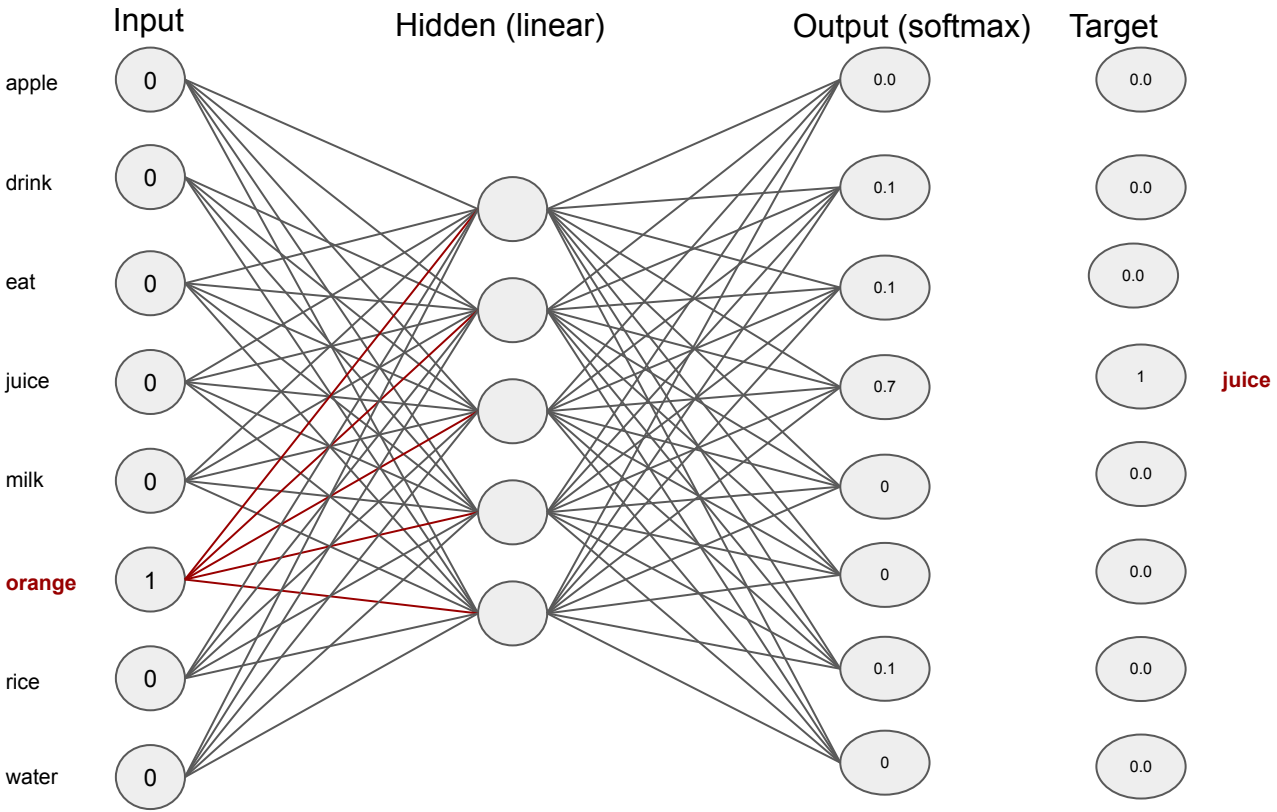
=

Probability that if you randomly pick a word nearby “eat”, that it is “apple”

p(apple|eat)

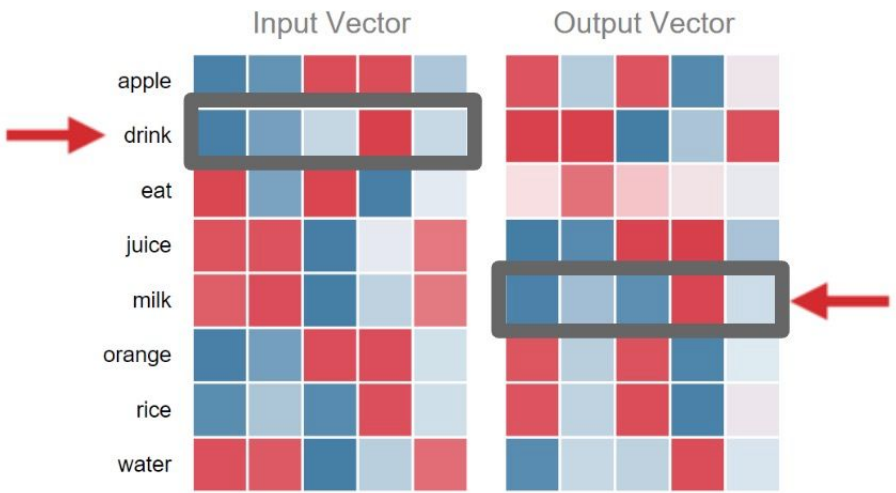
e^(IVec[eat]*OVec[apple]) + e^(IVec[eat]*OVec[juice]) + e^(IVec[eat]*OVec[drink])+e^(IVec[eat]*OVec[other vocab words)

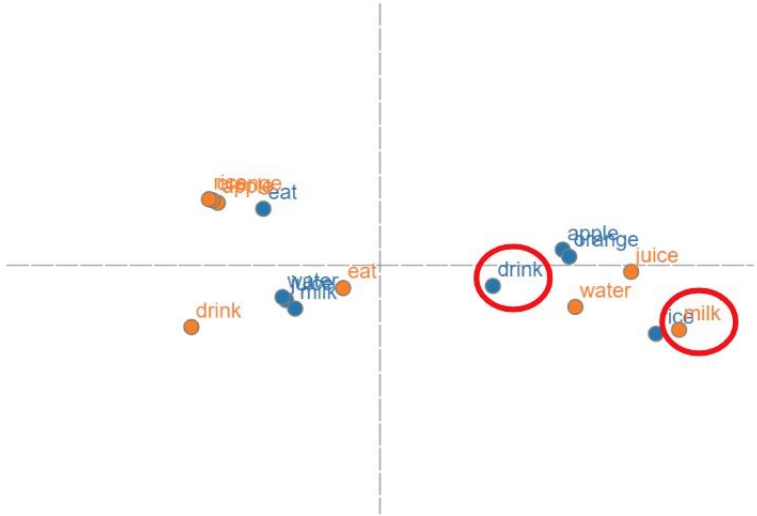
Word2Vec Network



Then, we can compute the loss and call gradient descent to update model parameters.

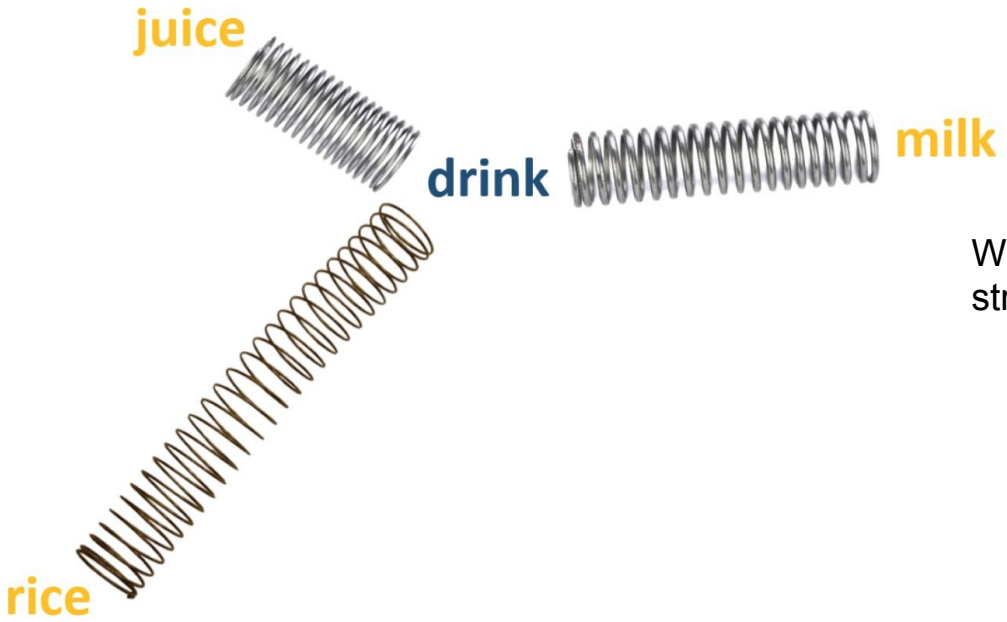
Updating Word Vectors





From Xin Rong 2016

A force-directed graph



What decides the strength of the string?

Idea behind Word2Vec

- Feature vector assigned to a word will be adjusted if it can not be used for accurate prediction of that word's context.

- Each word's context in the corpus is the teacher sending error signals back to modify the feature vector.

- It means that words with similar **context** will be assigned similar **vectors**!

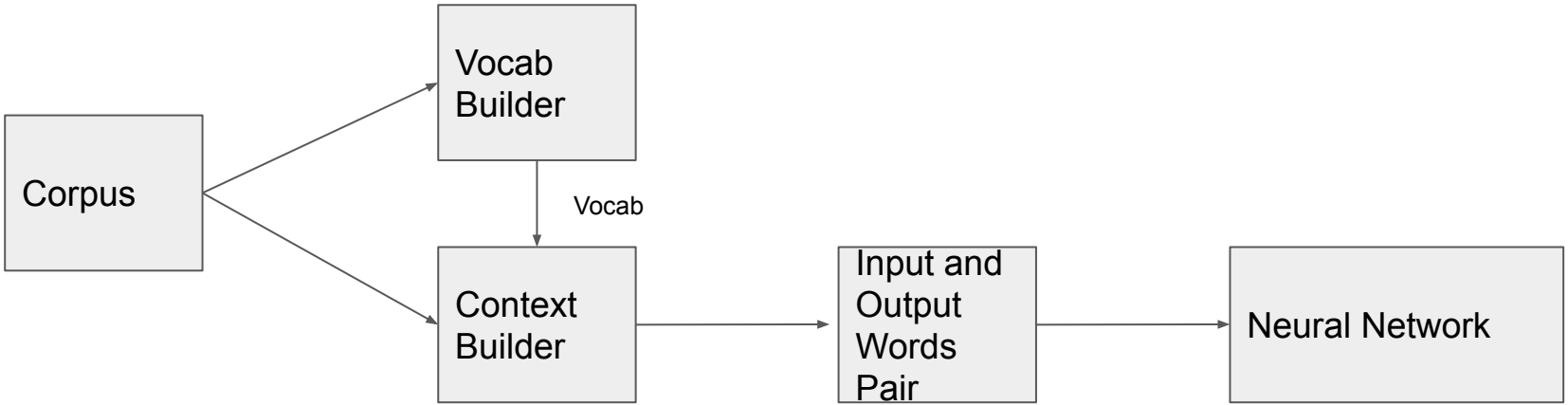
Distributional Semantics

"**You shall know a word by the company it keeps**" - by Firth (1957)

Input and Output Words

- How to select them from corpus

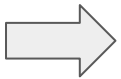
- Skip-gram and CBoW differ here.



Skip-Gram

- Task Definition: given a specific word, predict its nearby word (probability output)
- Model input: source word, Model output: nearby word
- Input is one word, output is one word
- The output can be interpreted as prob. scores, which are regarded as how likely it is that each vocabulary word can be nearby your input word.

give a talk at the

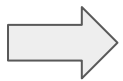


Input x	Target y
talk	give
talk	a
talk	at
talk	the

CBoW

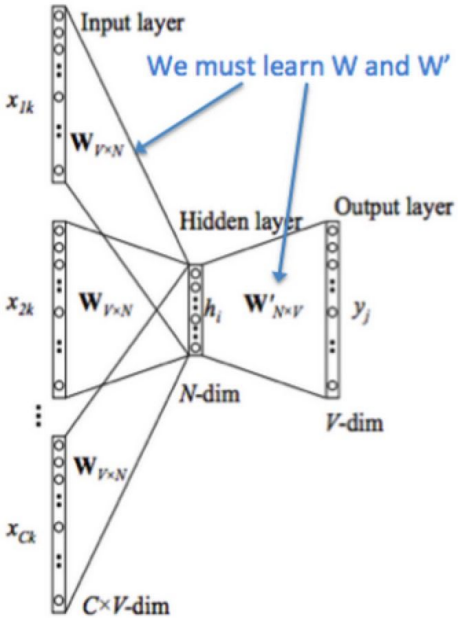
- Task Definition: given context, predict its target word
- Model input: context (several words), Model output: center word
- Input is several words, output is one word
- Core Trick: **average** these context vectors for prob score computing

give a **talk** at the



Input x
(give,a,at,the)

Target y
talk



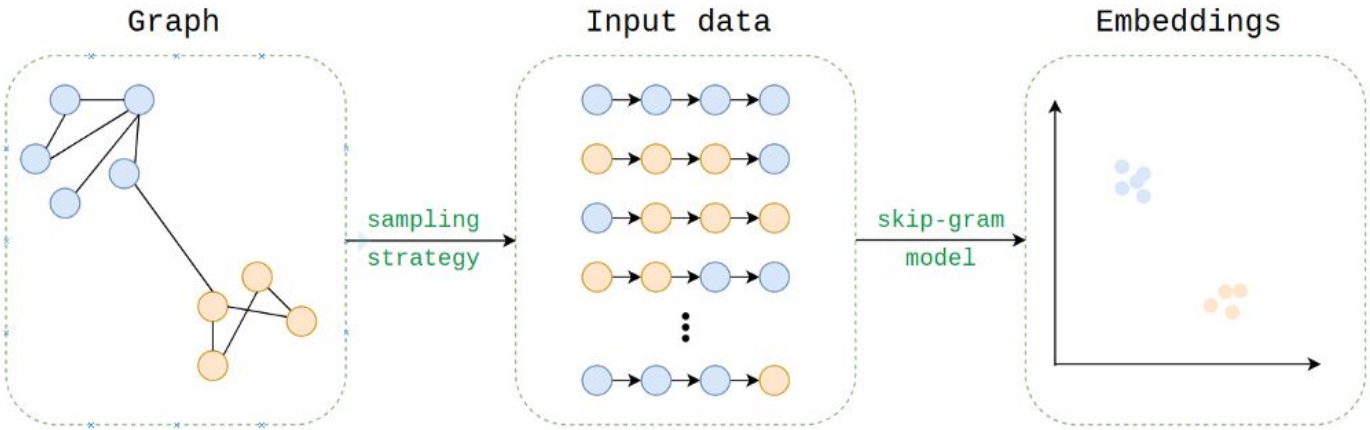
Skip-Gram Vs CBoW

- **CBoW: learning to predict the word by the context**
- **Skip-gram: learning to predict the context by the center word**

- ？: several times faster to train the ？
- ？: works well with small amount of the training data, represents well even rare words or phrases.

Embedding for Graph Data

- Embeddings can be extended beyond NLP domain
- Embeddings can be learned for any nodes in a graph



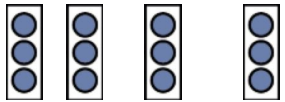
- Nodes can be items, web pages and so on in user clicked stream data
- Embeddings can be learned for any group of discrete and co-occurring states.

Neural Networks for NLP

Sequence of Words

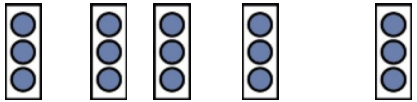
- Each sentence or document can be regarded as a sequence of vectors.

I hate this movie



4 by d

This is my favorite movie.



5 by d

- The shape of matrix depends on the length of sequence. However, the majority of ML systems need fixed-length feature vectors.
- One simple solution: average the sequence of vectors, just like bag-of-words (abandon order information).

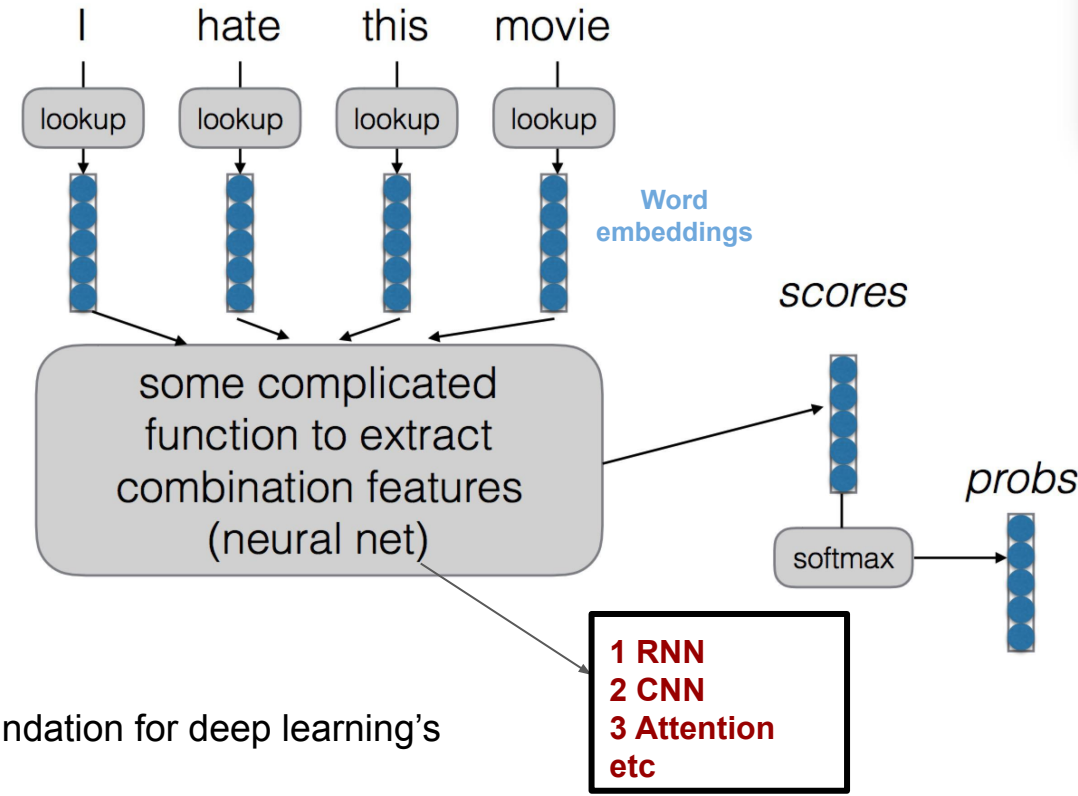
Complex Semantic

1. **Input Text:** a sequence of words;

2. **Through Word Embedding Look-up:** a sequence of word vectors;

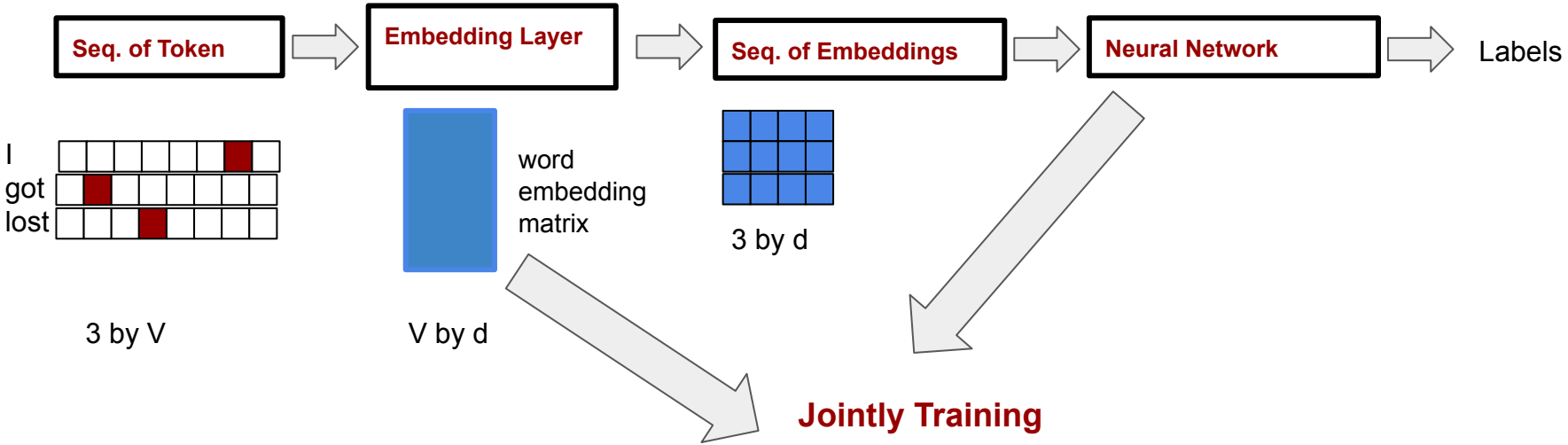
3. Neural networks is applied upon the vector sequences to learn semantic **composition** for final prediction;

➔ Human understand the word meaning firstly, then get the whole sentence meaning by composing these words' meaning together.



Word Embeddings is the foundation for deep learning's applications on NLP

Neural Networks for NLP



Three different approaches to build word embedding layer

1. Learn from Scratch

   Random initialize the word embedding matrix and update the matrix and neural network parameters in the specific task

2. Pre-train

   Got pre-trained word embeddings as the embedding layer and only update neural network parameters in the specific task

3. **Pre-train then fine tune**

Pre-train then Fine-tune

Text without annotation

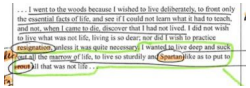
A model that can “read” text

Pre-train

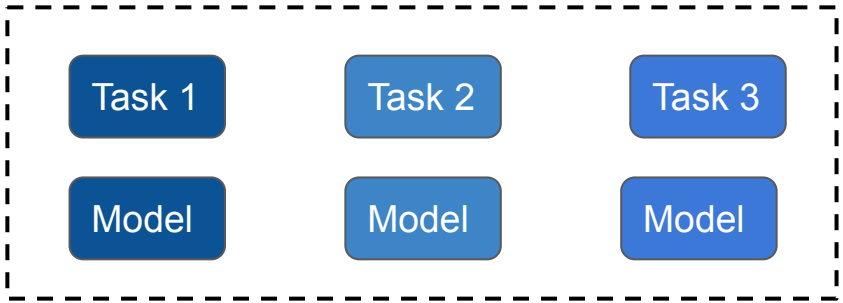
Model

In this lecture, the model is only referred to word embedding matrix. However, in state-of-arts NLP techniques, the model will also include the following neural network.

Fine-tune

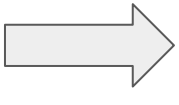


Task-specific data with labels

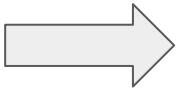


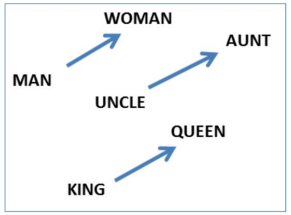
Is word2vec good enough?

Corpus



Word2Vec Tool



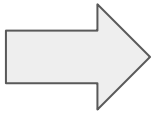


Word Embeddings used for downstream tasks

- Can not capture different senses of words (context independent)
  - Solution: Take the word order into account

- Can not address Out-of-Vocabulary words
  - Solution: Use characters or subwords

Contextualized Word Embeddings: BERT



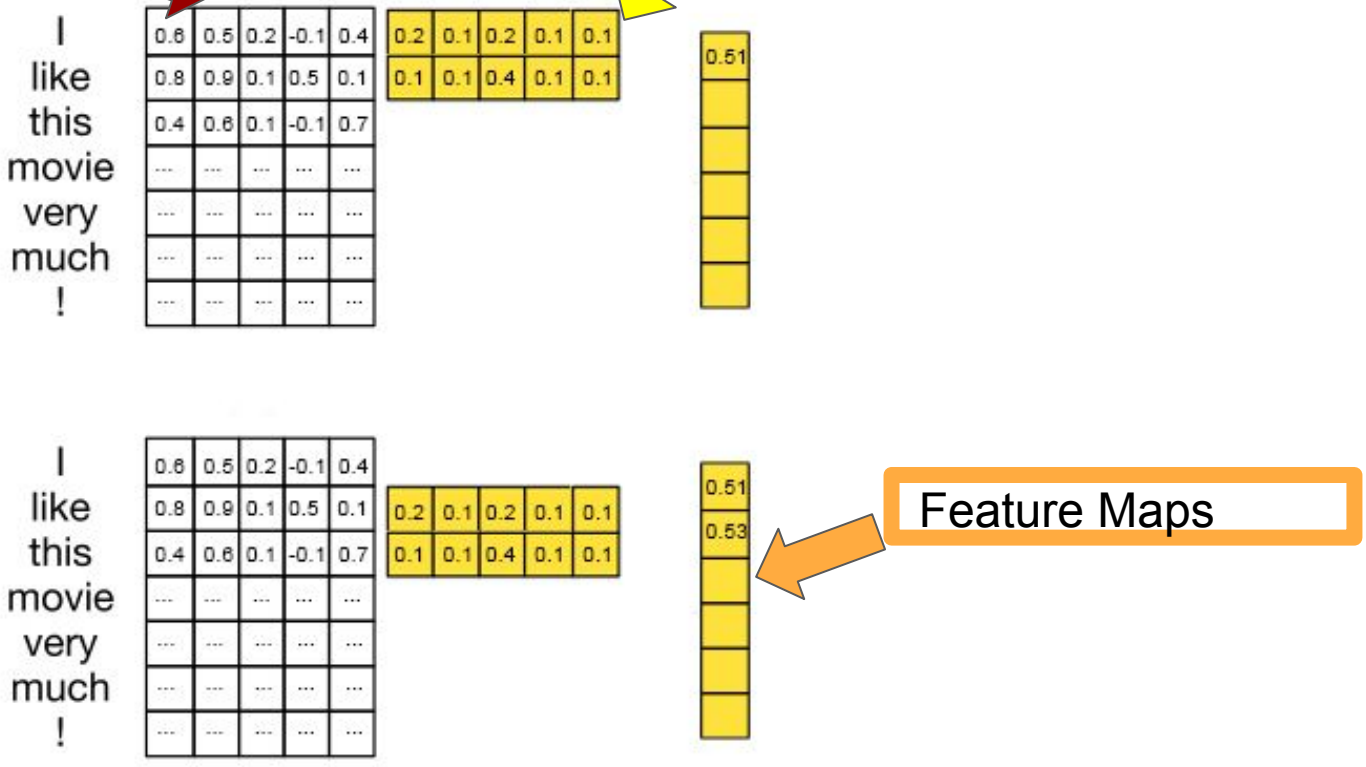
1 use transformer to capture word order
2 input is subwords and output is subwords embeddings

CNN for Text

Convolution Operation

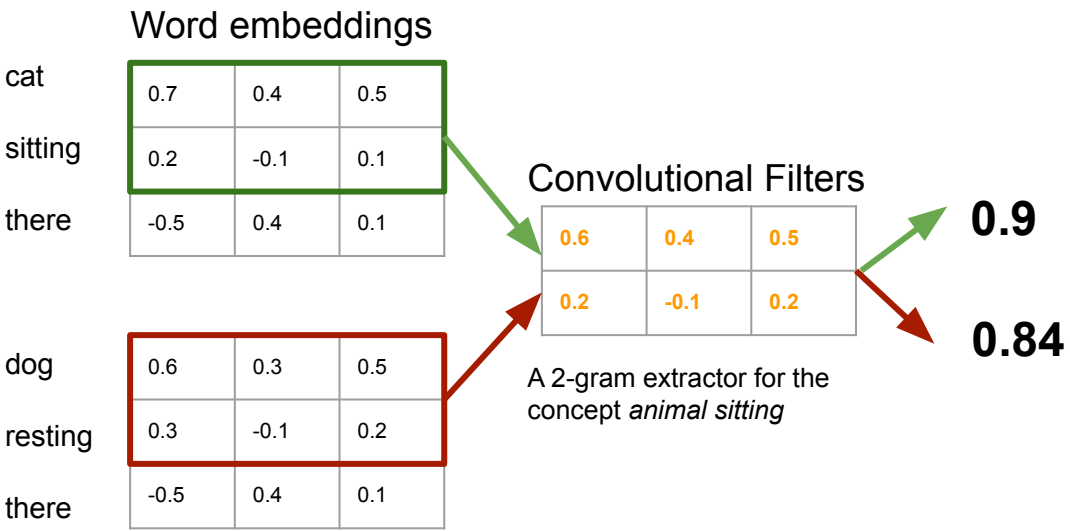
Word Vectors

Filters updated during training



Feature Maps

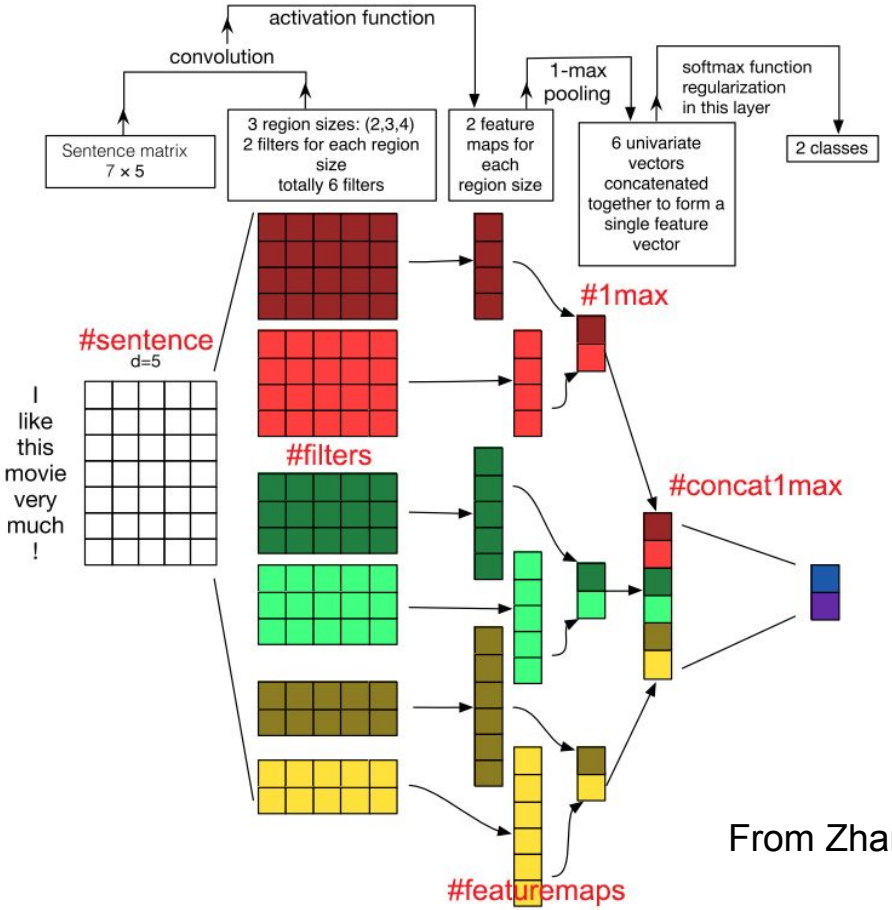
Toy Example



- This convolution provides high activations for 2-grams with certain meaning
- Can be extended to 3-grams, 4-grams, etc.
- Can have various filters, need to track many n-grams.
- They are called 1D since we only slice the windows only in one direction

Why is it better than BoW?

CNN Framework



From Zhang 2015

Multiple Channels

- Like image, CNN is applied on R-G-B channels
- For NLP, different word embeddings can be regarded as different channels

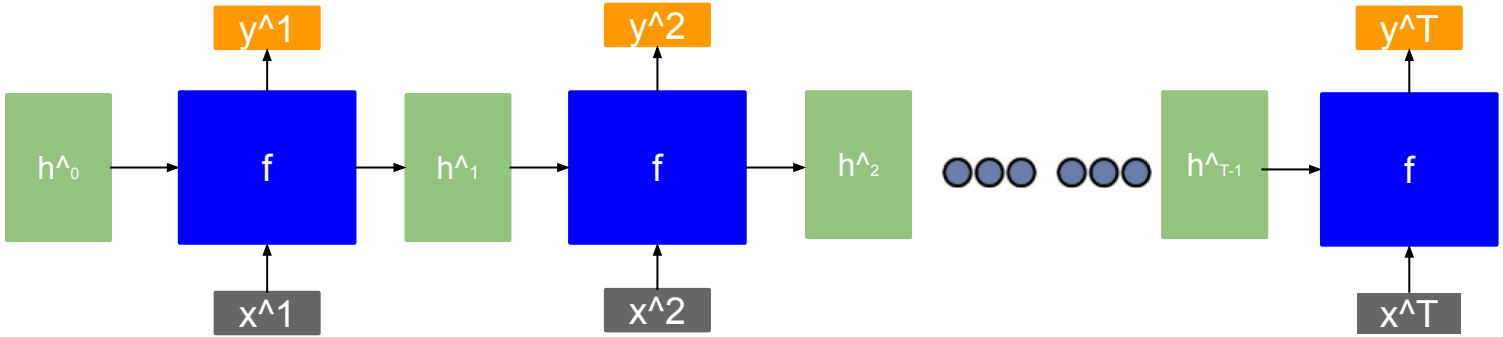
CNN for NLP

1. n-grams features are important (window size)
2. Location of key n-grams are trivial (pooling)
3. Stack of Convolutional layer or large window size can also capture long-range information

RNN for Text

Recurrent Neural Network

Given a sequence of T steps: x^1, x^2, …, x^T



For each step, the same function f is applied as:

Output Vectors

$$\mathbf{h}', \mathbf{y} = f(\mathbf{h}, \mathbf{x})$$

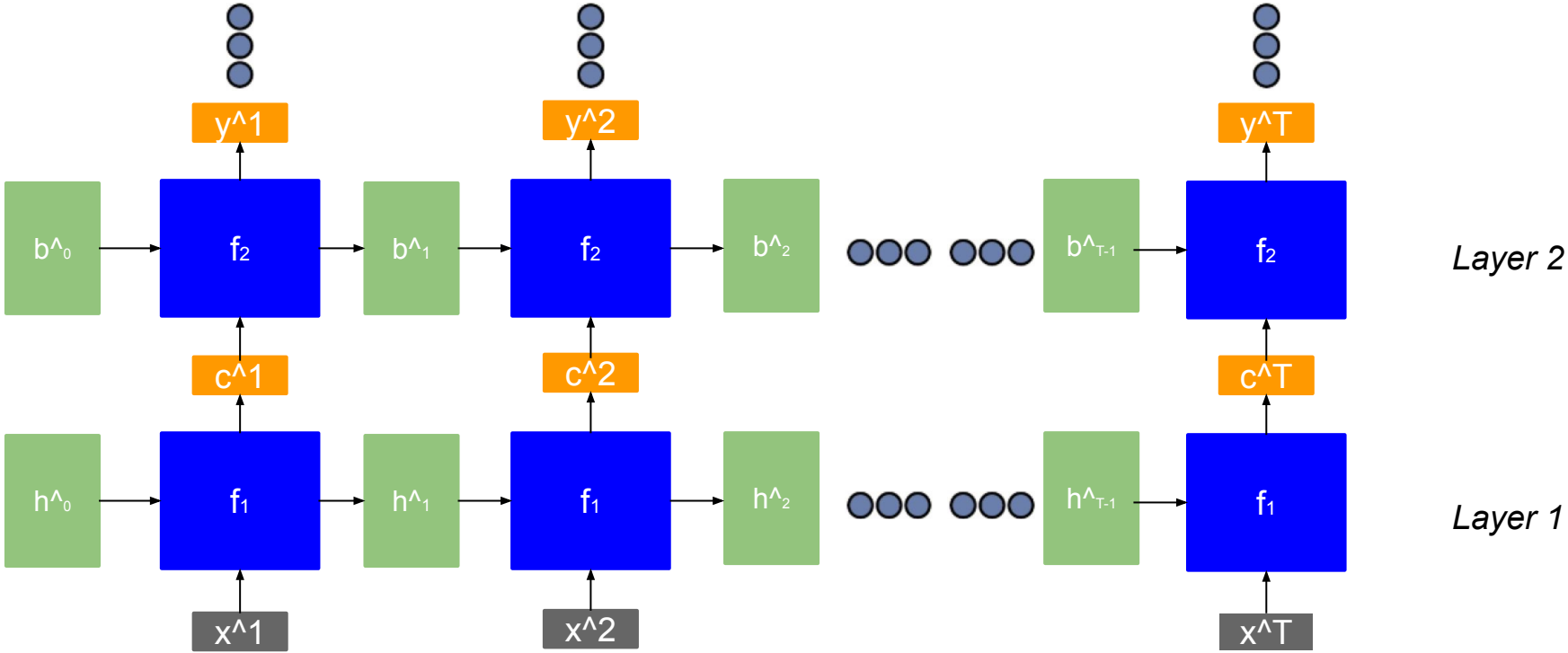
Input Vectors

Hidden vectors with the same dimension

Deep RNN

$$\mathbf{h}', \mathbf{c} = f_1(\mathbf{h}, \mathbf{x}) \quad \mathbf{b}', \mathbf{y} = f_2(\mathbf{b}, \mathbf{c})$$

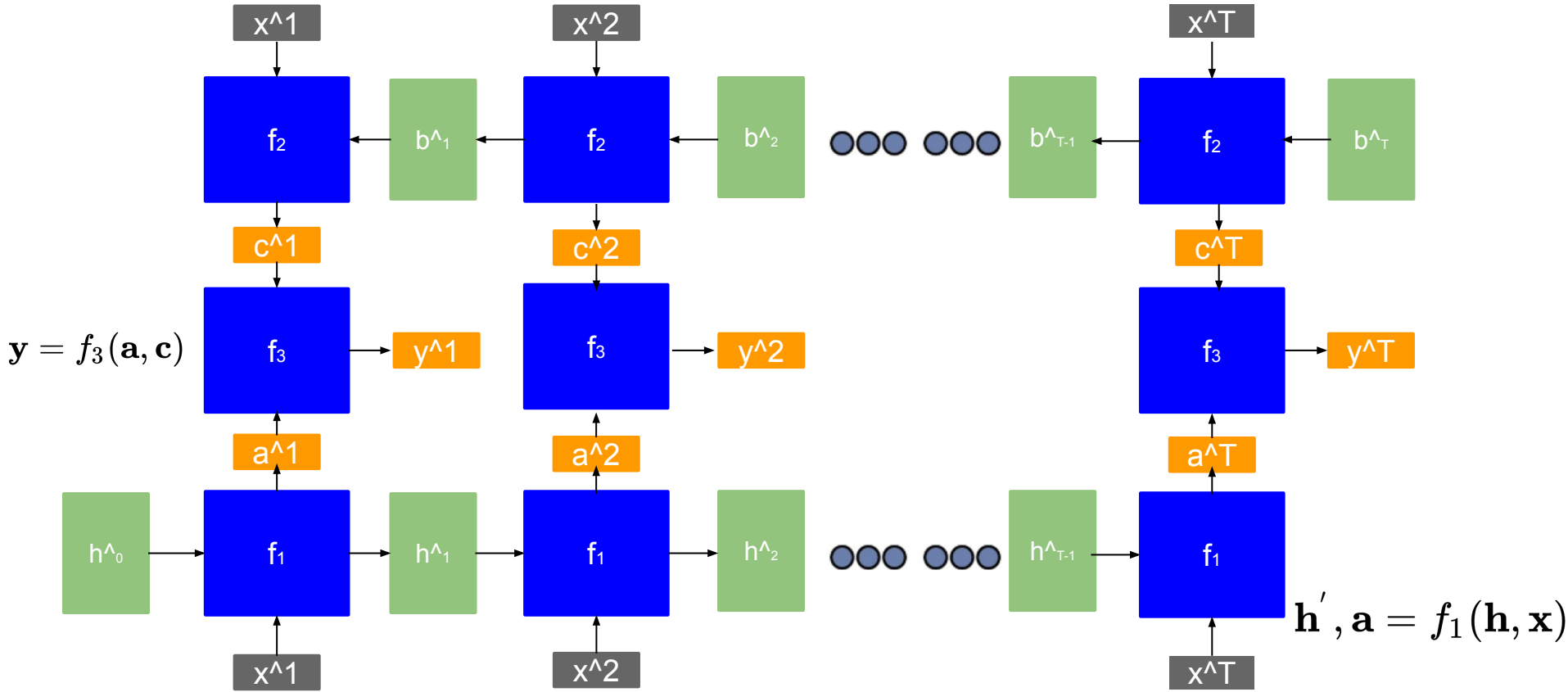
Given a sequence of T steps: x^1, x^2, …, x^T



Bidirectional RNN

Given a sequence of T steps: x^1, x^2, …, x^T

$$\mathbf{b}', \mathbf{c} = f_2(\mathbf{b}, \mathbf{x})$$

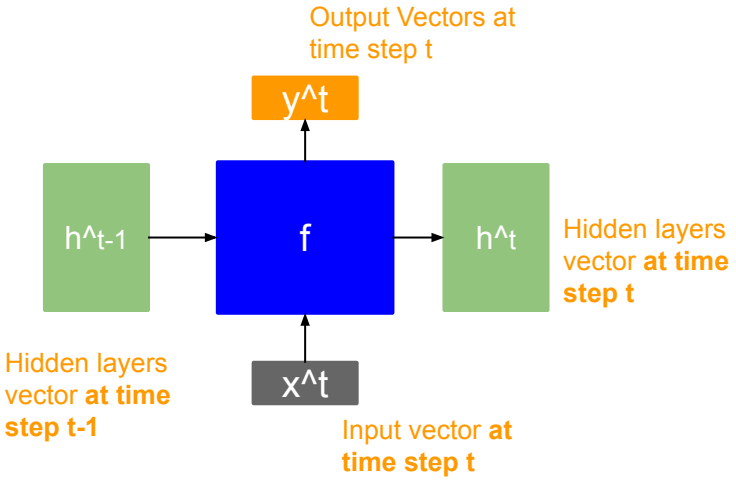


$$\mathbf{y} = f_3(\mathbf{a}, \mathbf{c})$$

$$\mathbf{h}', \mathbf{a} = f_1(\mathbf{h}, \mathbf{x})$$

Naive RNN

Given the function: $\mathbf{h}', y = f(\mathbf{h}, \mathbf{x})$



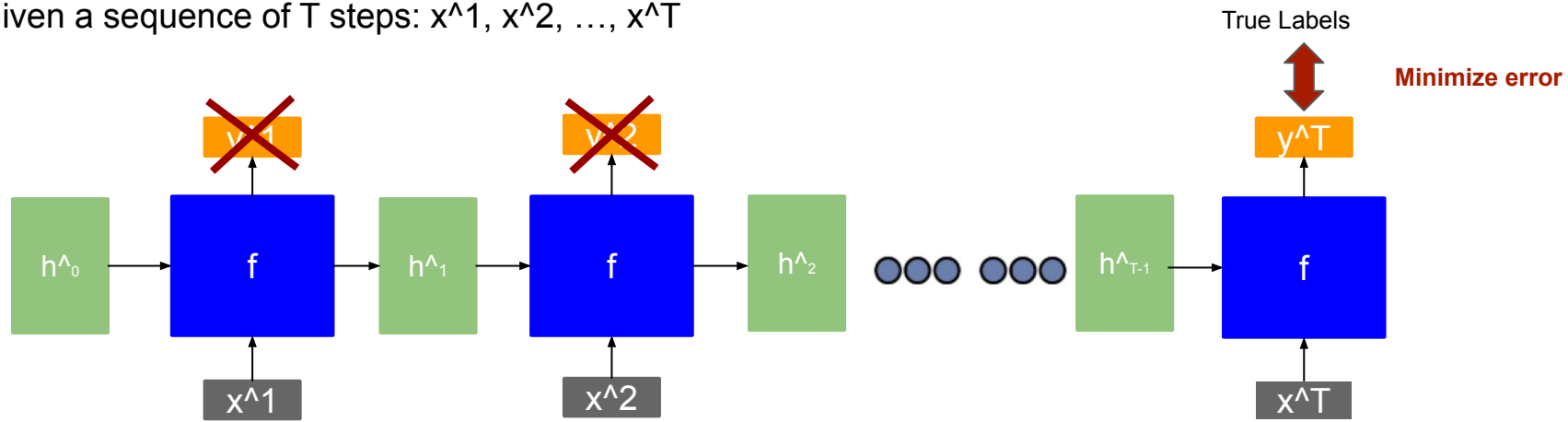
$$\mathbf{h}_t = \sigma(\mathbf{W}^{hh}\mathbf{h}_{t-1} + \mathbf{W}^{i}\mathbf{x}_t)$$

$$\mathbf{y}_t = softmax(\mathbf{W}^{o}\mathbf{h}_t)$$

$\mathbf{W}^{hh}, \mathbf{W}^{i}, \mathbf{W}^{o}$ **Time-Invariant** Model Parameters

RNN for Sequence Classification

Given a sequence of T steps: x^1, x^2, …, x^T



Only apply classification layer on the hidden output of the last time step.

RNN’s Bottleneck

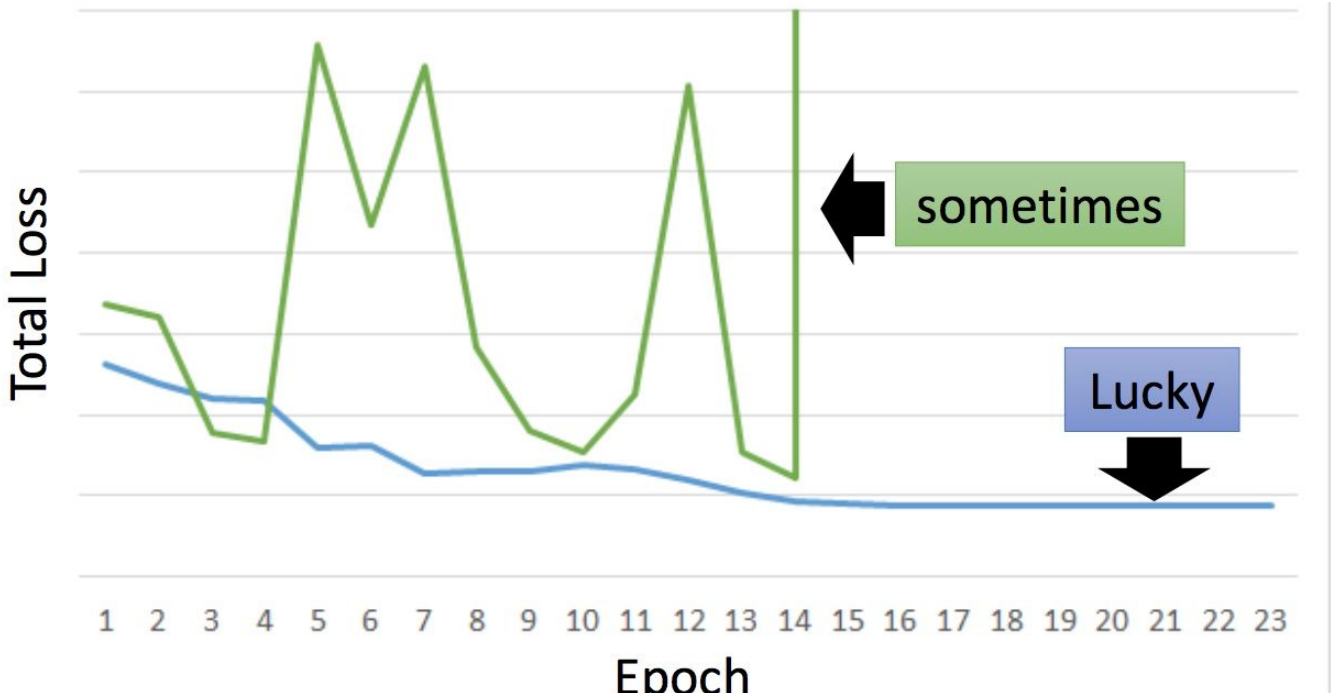
- RNN is not suitable for parallel computation.
- RNN’s training is not easy
  - Gradient Vanishing
  - Gradient Exploding
  - Reminder on gradient descent algorithms:

$$W = W - \alpha \frac{dL}{dW}$$

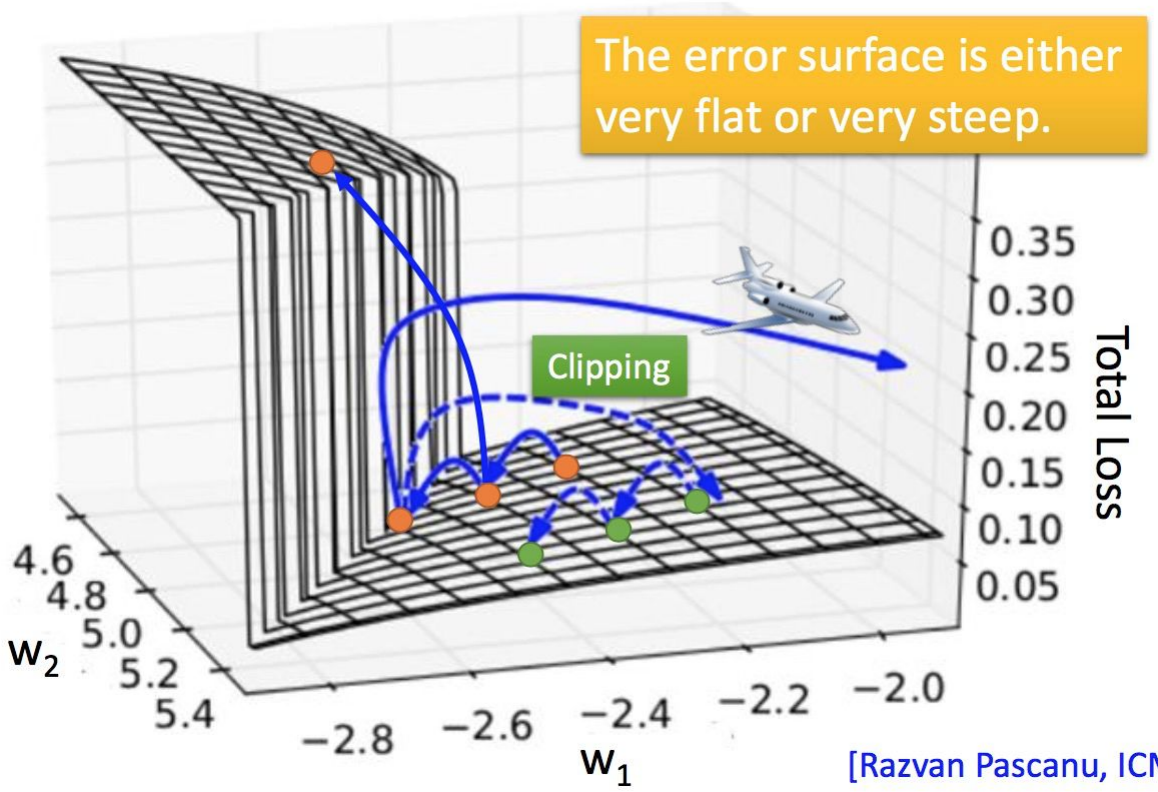
W is the layer weights;
L is the loss function;
Alpha is the learning rate, which represents how aggressive that our model parameters are updated.

RNN Training is Hard

- Real experiments on Language Models



Rough Error Surface of RNN

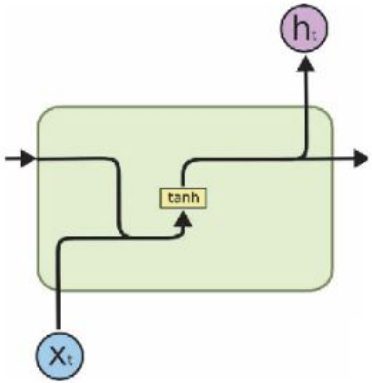


Exploding/Vanishing Gradient

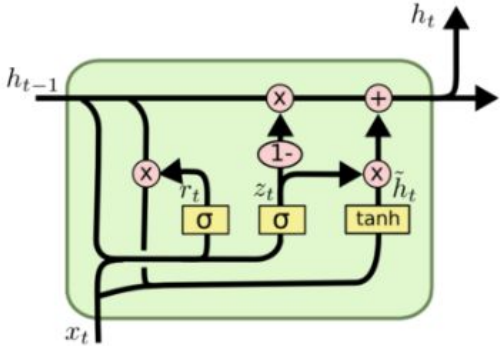
[Razvan Pascanu, ICML'13]

LSTM/GRU

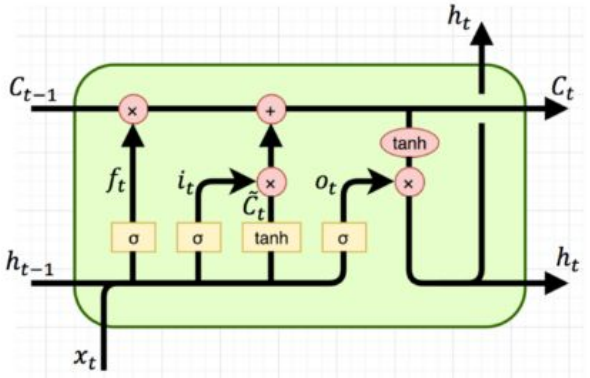
- LSTM/GRU are using gates in cell computation to control information flow



Naive RNN

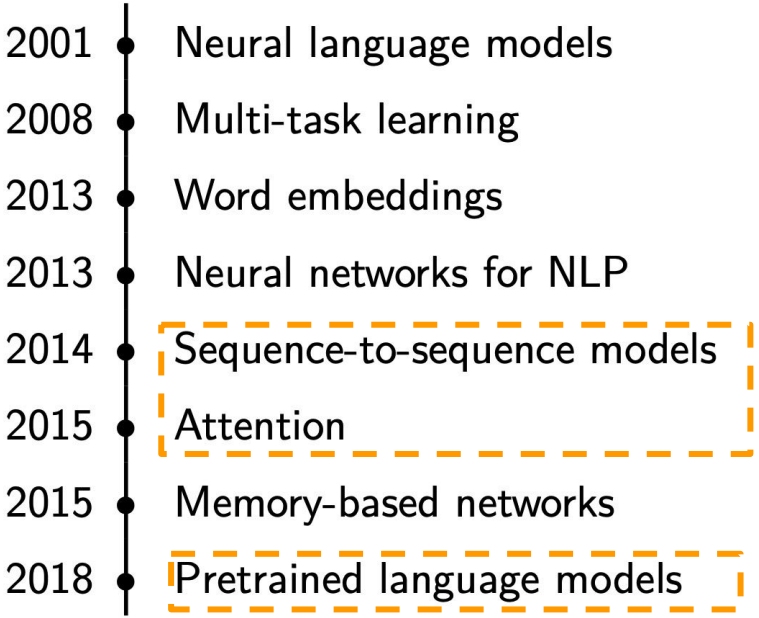


GRU



LSTM

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/


2001 Neural language models
2008 Multi-task learning
2013 Word embeddings
2013 Neural networks for NLP
2014 Sequence-to-sequence models
2015 Attention
2015 Memory-based networks
2018 Pretrained language models