Data Mining
(EECS 4412)

Text Classification

Parke Godfrey

EECS

Lassonde School of Engineering

York University

Thanks to

Professor Aijun An

for creation & use of these slides.

Outline

- Introduction and applications
- Text Representation (traditional)
- Text Preprocessing Steps
- Advanced techniques for text representation (word embedding)

Text Mining

- It refers to data mining using text documents as data.
  - A text document could be an article, a web page, a product review, an xml file, an email message, a blog and so on.
- Tasks of text mining
  - Text classification
  - Text clustering
  - Text summarization
  - Topic detection
  - ...

Text Classification

- Learn a classification model from a set of pre-classified documents
- Classify new text documents using the learned model
- Applications
  - Classify articles into categories
  - Classify web pages into different categories
  - Classify emails into different categories
  - Spam email filtering
  - ...

Example Applications

- News topic classification (e.g., Google News)

  C={politics, sports, business, health, tech, …}

- "SafeSearch" filtering

  C={pornography, not pornography}

- Language classification

  C={English, Spanish, Chinese,…}

- Sentiment classification

  C={positive review, negative review, neutral review}

- Email sorting

  C={spam, meeting reminders, invitations, …} – user-defined!

Text Representation

- Most classification learning programs require the examples to be represented as a tuple, which is a vector of attribute values.
- How to represent a document using a vector of attribute values?
- Typical method (which has become traditional):
  - Attributes
    - "Bag of words" method: Use a set of words as attributes
  - Attribute values
    - Method 1: use 0 or 1 as attribute value to indicate whether the word appears in the document.
    - Method 2: use the absolute or relative frequency of each word in the document as the attribute value.
    - Method 3: assign a weight to a word in a document using TF-IDF and use the weight as the attribute value

Text Representation (*Cont'd*)

Training data sets:

- Method 1:

	word$_1$	**word$_2$**	**...**	**word$_m$**	**Class**
document$_1$	0	1	…	1	C1
document$_2$	1	0	…	1	C2
...	…	…	…	…	…
document$_n$	1	0		0	C2

- Method 2 with absoluate term frequency:

	word$_1$	**word$_2$**	**...**	**word$_m$**	**Class**
document$_1$	0	3	…	1	C1
document$_2$	2	0	…	3	C2
...	…	…	…	…	…
document$_n$	5	0		0	C2

Method 3: TF-IDF Term Weighting

- TF: term frequency
  - Definition: TF = t_{ij}
    - frequency of term i in document j
  - Purpose: makes the frequent words *for the document* more important
- IDF: inverted document frequency
  - Definition: IDF = $\log(N/n_i)$
    - n_i : number of documents containing term i
    - N : total number of documents
  - Purpose: makes rare words *across documents* more important
- TF-IDF value of a term i in document j
  - Definition: TF×IDF = $t_{ij} \times \log(N/n_i)$

Example: TF-IDF Weighted Vectors

Assume there are three documents in the training set:

Document D1: “yes we got no bananas”

Document D2: “what you got”

Document D3: “yes I like what you got”

	yes	we	got	no	bananas	what	you	I	like
D1:	.18	0.48	0	0.48	0.48	0	0	0	0
D2:	0	0	0	0	0	0.18	0.18	0	0
D3:	0.18	0	0	0	0	0.18	0.18	0.48	.48

Text Processing for Selecting the Bag of Words

- Word (token) extraction
  - Extract all the words in a document
  - Convert them into lower cases
- Stop words removal
- Stemming
- Selecting words

Stop Words

- Many of the most frequently used words in English are worthless in text mining – these words are called *stop words*.
- Examples of stop words

  the, of, and, to, a, …
- Typically about 400 to 500 such words
- For an application, there may be additional domain-specific stop words
- These stop words are usually removed from the set of words for representing a document.

Stemming

- A technique used to find the root/stem of a word.
- For example:
  - discussed
  - discusses
  - discussing
  - discuss

  Stem: discuss
- Usefulness
  - Reduce the number of words
  - Improve effectiveness of text classification

Example Stemming Rules

- Remove ending
  - If a word ends with *s*, preceded by a consonant other than an *s*, then delete the *s*.
  - If a word ends with *ed*, preceded by a consonant, delete the *ed* unless this leaves only a single letter.
- Transform words
  - If a word ends with “ies” but not “eies” or “aies”, then “ies” is replaced with “y”.

Stemming Algorithms

- Porter stemming algorithm
  - The most widely used stemming algorithm
  - Developed by Martin Porter at the University of Cambridge in 1980
  - http://www.tartarus.org/~martin/PorterStemmer/ contains source codes in a few languages
- Other stemming algorithms
  - http://www.comp.lancs.ac.uk/computing/research/stemming/general/

Text Processing for Selecting the Bag of Words

- Word (token) extraction

  - Extract all the words in a document

  - Convert them into lower cases

- Stop words removal
- Stemming
- *Selecting words*

Feature Selection

- Selecting the "bag of words" to represent documents
- Why do we need to select?
  - The number of unique words in a set of documents can be too many
  - Leaning program may not be able to handle all possible features
  - Good features can result in higher accuracy

What are Good and Bad Features?

- Good features: (should be kept)
  - Co-occur with a particular category
  - Do not co-occur with other categories
- Bad features: (best to remove)
  - Uniform across all categories
  - Very infrequent (appear 1 or 2 times in the whole training set of documents)
    - unlikely to be met again
    - can be noise
    - co-occurrence with a class can be due to chance

Feature Selection Methods

- Class independent methods (Unsupervised)
  - Document Frequency (DF)
  - Term Strength (TS)
- Class-dependent methods (Supervised)
  - Information Gain (IG)
  - Mutual Information (MI)
  - χ^2 statistic (CHI)

Document Frequency (DF)

- Document frequency of a word *w*:

$$DF(w) = \text{number of documents containing } w$$

- Rank the words according to their document frequency

- Select the first *m* words with high DF values

Document Frequency (*Cont'd*)

- Advantages
  - Easy to compute
  - Can remove rare words (hence noise)
- Disadvantages
  - Class independent.
    - If the word appears frequently in many classes, it cannot distinguish the classes well
  - Some infrequent terms can be good discriminators, which cannot be selected by this method.

Information Gain

- A measure of importance of the feature for predicting the classes of documents
- Defined as:
  - The number of "bits of information" gained by knowing the word is present or absent

$$Gain(w) = -\sum_{i=1}^{k} P(C_i)\log P(C_i)$$
$$+ P(w)\sum_{i=1}^{k} P(C_i \mid w)\log P(C_i \mid w) + P(\overline{w})\sum_{i=1}^{k} P(C_i \mid \overline{w})\log P(C_i \mid \overline{w})$$

where w is a word and $C_1, C_2, \ldots, C_k$ are classes.

- Rank the words according to their information gain value
- Select the first m words with high gain values

Information Gain (*Cont'd*)

- Advantage:
  - Consider the classes
- Disadvantage:
  - Computationally expensive (compared to using DF)
  - Noisy words occurring only once in the document collection have high IG
- Solution
  - Remove rare words (appears 1 or 2 times) first. This can
    - reduce the amount of computation, and
    - remove noisy words that have by-chance correlations with the classes.

What Do People Do In Practice?

- Rare term removal
  - rare across the whole collection (i.e. DF is very low)
  - met in a single document
- Most frequent term removal (i.e. removing stop words) (*often*)
- Stemming. (*often*)
- Use a class-dependent method (e.g., the information gain method) to select features.

Outline

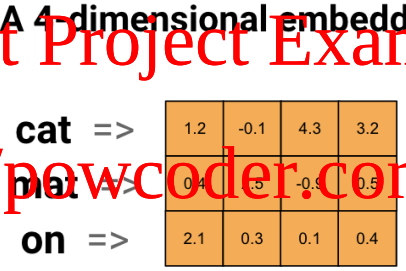
- Introduction and applications
- Text Representation (traditional)
- Text Preprocessing Steps
- *Advanced techniques for text representation (word embedding)*

Beyond Bag of Words

- Bag of words representation
  - does not consider the position or order of words in a document
  - does not consider the context a word is in.
  - does not consider semantic relationships between words
- It would be great to include multi-word features like "New York", rather than just "New" and "York"
- Bigram document representation (or n-gram in general)
  - a pair of consecutive words in the document
  - But: including all pairs of words, or all consecutive pairs of words, as features creates WAY too many features to deal with, and many are very sparse.
- Document representation using *word embeddings*

Word Embedding

- A type of word representation using vectors
  - Embed words into a vector space
  - The vector of a word is called its embedding



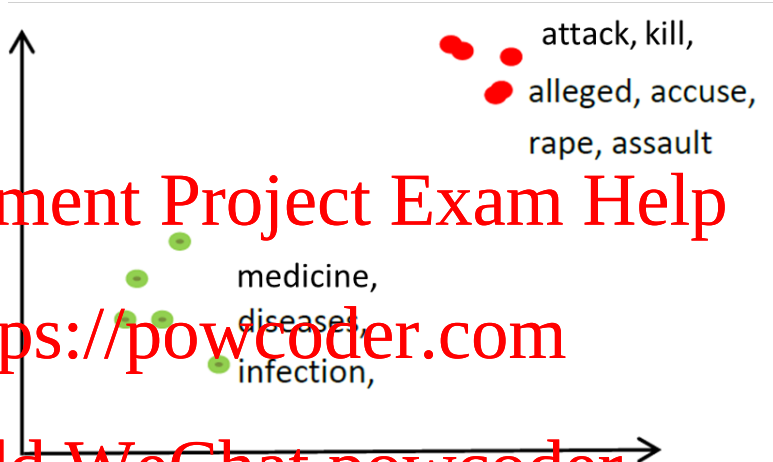
- Given a vocabulary of words
  - The embedding of the i^{th} word in the vocabulary is a *d-dimensional vector*:

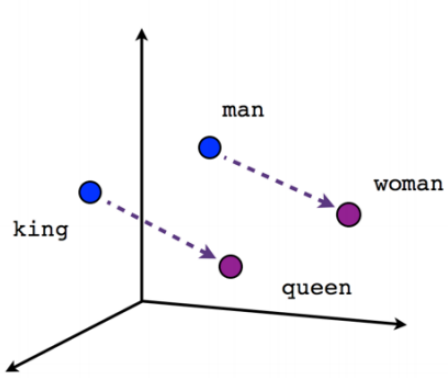
$$w_i \in \Re^d$$

where *d* is typically in the range 50 to 1000.

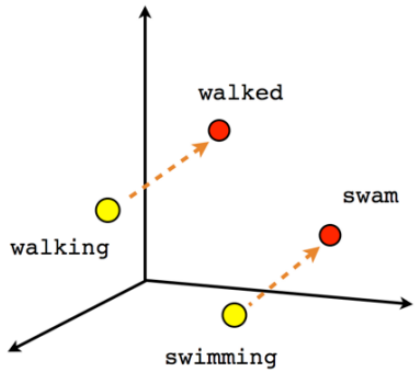
Important Feature of Word Embedding

- In an embedding space, semantically similar words are close to each other:

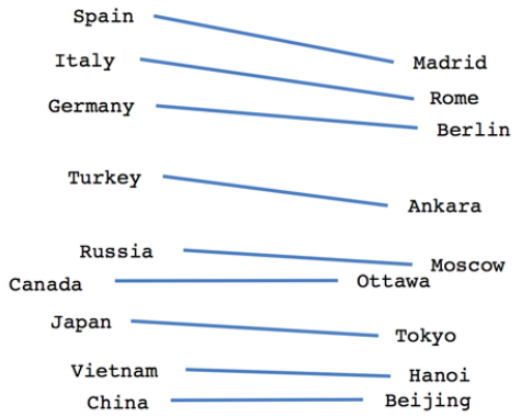




Male-Female



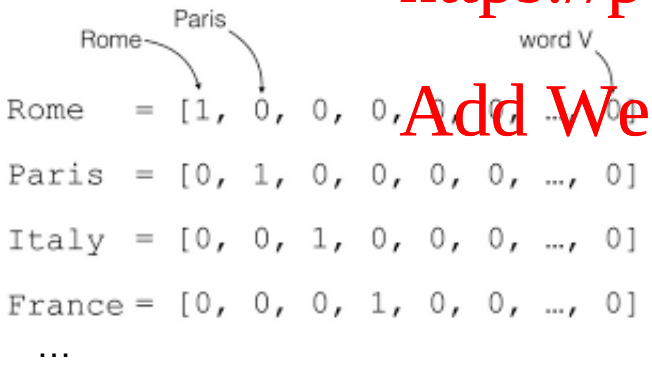
Verb tense



Country-Capital

Benefits of Word Embedding

- Word embeddings is better than one-hot encoding
- One-hot encoding:
  - Large dimension: size of vocabulary (could be tens of thousands)
  - Sparse
  - Words are considered independent (no semantics is encoded)



- Word embedding
  - Low dimensionality: typically 50 – 1000
  - Dense and distributed
  - Semantically related words are close to each other.

Rome = [1.73 0.73 -0.90 -0.62 0.12 -1.35 ...]
Paris = [0.77 1.18 -1.12 -0.75 -0.60 -1.05 …]
Italy = [1.77 -0.78 -0.95 0.33 0.04 -0.09 …]
France = [0.67 0.30 -1.05 -0.70 -0.78 0.00 …]

Main Methods for Word Embedding

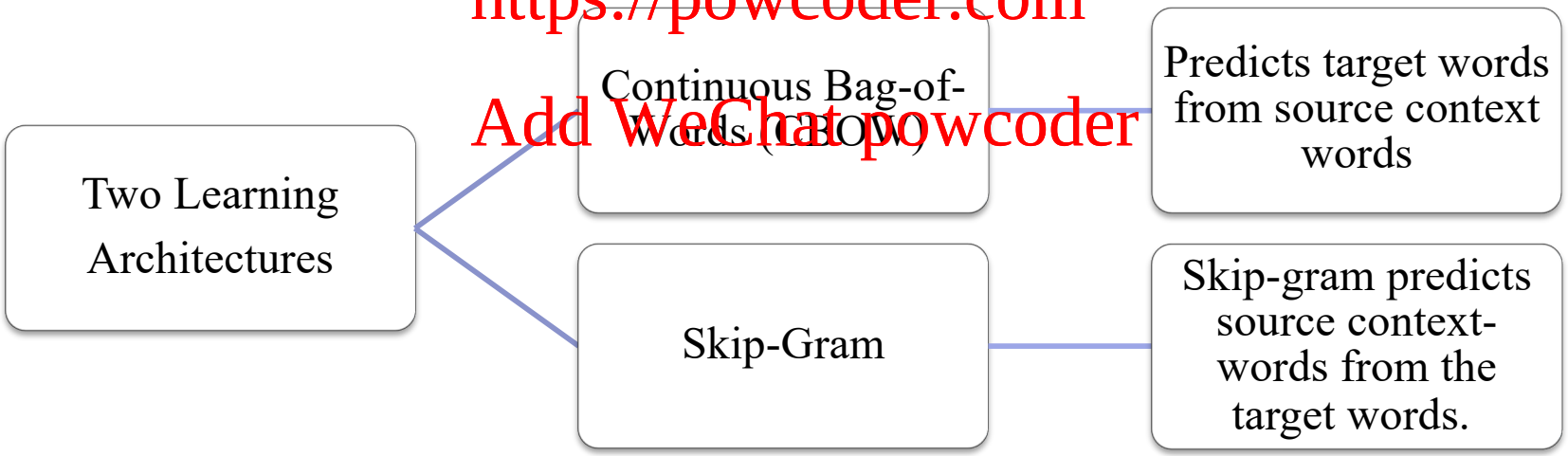
- Latent Semantic Indexing (Deerwester et al., ’88).
- Neural Net Language Models (NN-LMs) (Bengio et al., ’06)
- Convolutional Nets for tagging (SENNA) (Collobert & Weston, ’08).
- Supervised Semantic Indexing (Bai et al., ’09).
- Wsabie (Weston et al., ’10).
- Recurrent NN-LMs (Mikolov et al., ’10).
- Recursive NNs (Socher et al., ’11).
- **Word2Vec** (Mikolov et al., ’13).
- GloVe (Pennington et al., ’14)
- BERT (Devlin et al., ’18)

Characteristics of Word2Vec

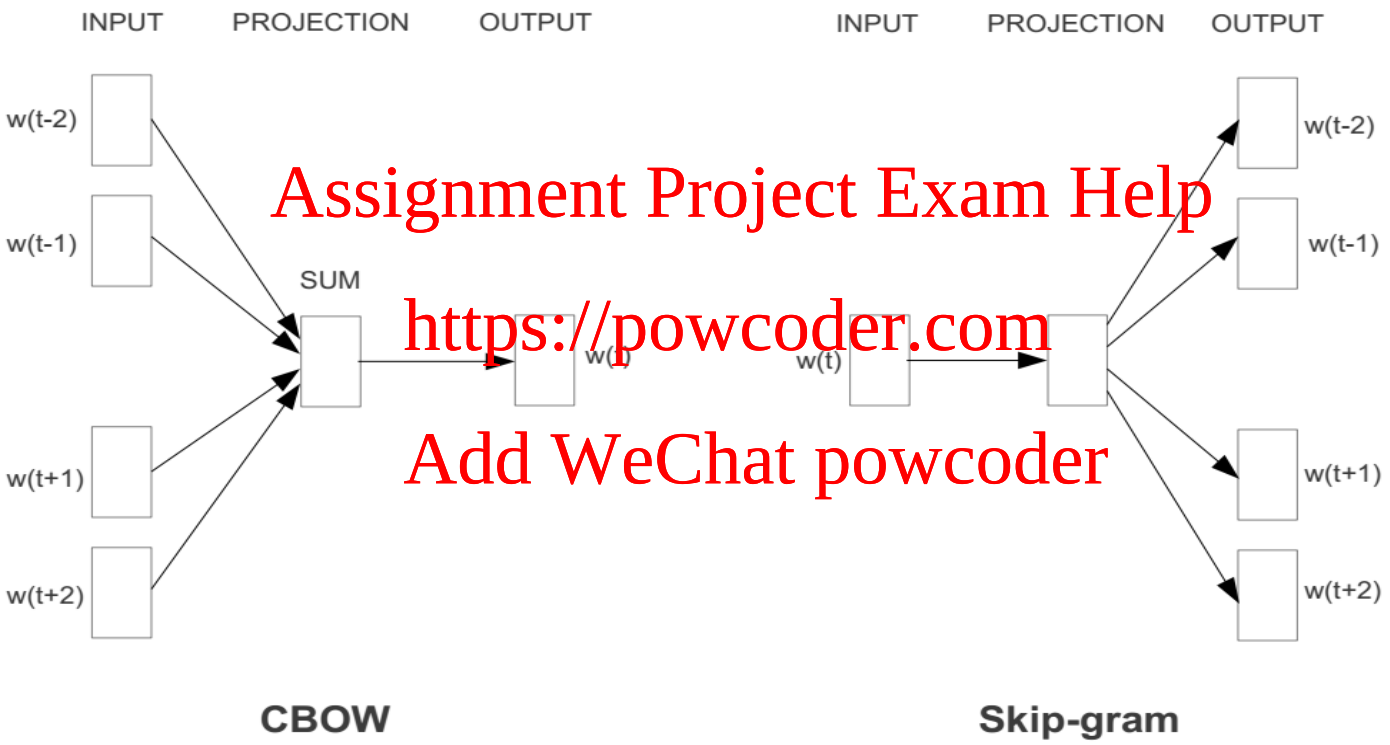
- Word2Vec is a model for learning word embeddings from a text corpus.
- Use a simple feedforward neural network with a single hidden layer.
- Word2Vec is a successful example of "shallow" learning.

Two Learning Architectures of Word2Vec

- Word2Vec learns word embeddings
  - from a large text corpus (i.e., a large collection of text documents, e.g., Wikipedia documents)
  - using one of the two architectures:



Word2Vec: Two Learning Architectures



Snapshot of the training text corpus (for context window size = 2):

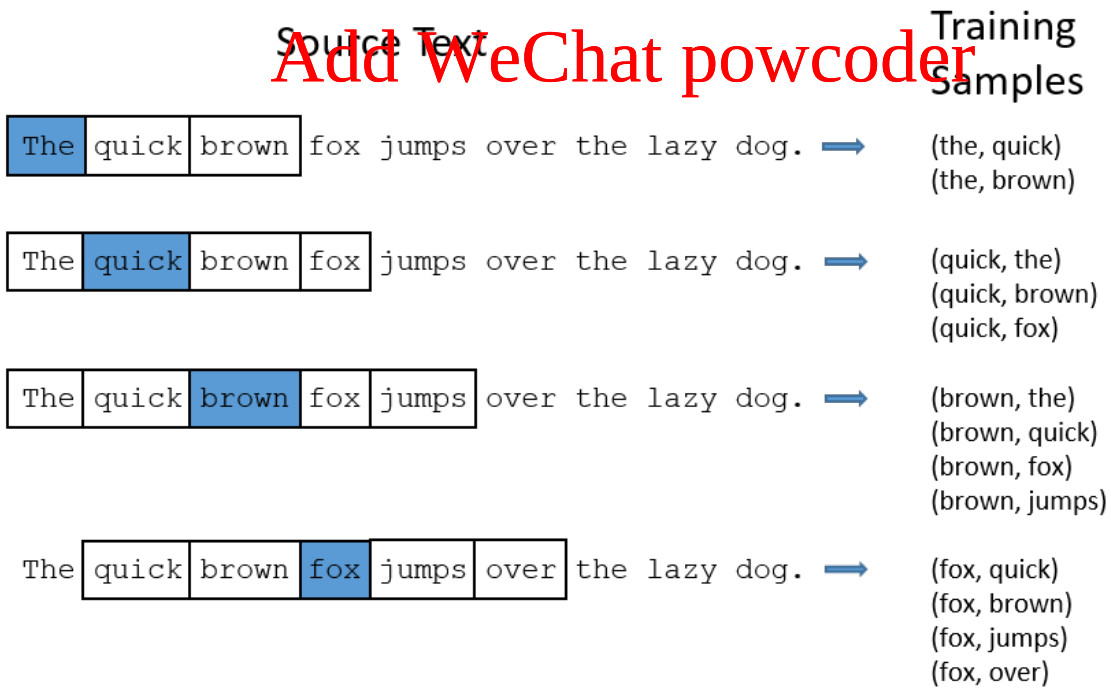
… The cute cat jumps over the lazy dog …

w(t-2) w(t-1) w(t) w(t+1) w(t+2)

context words target word context words

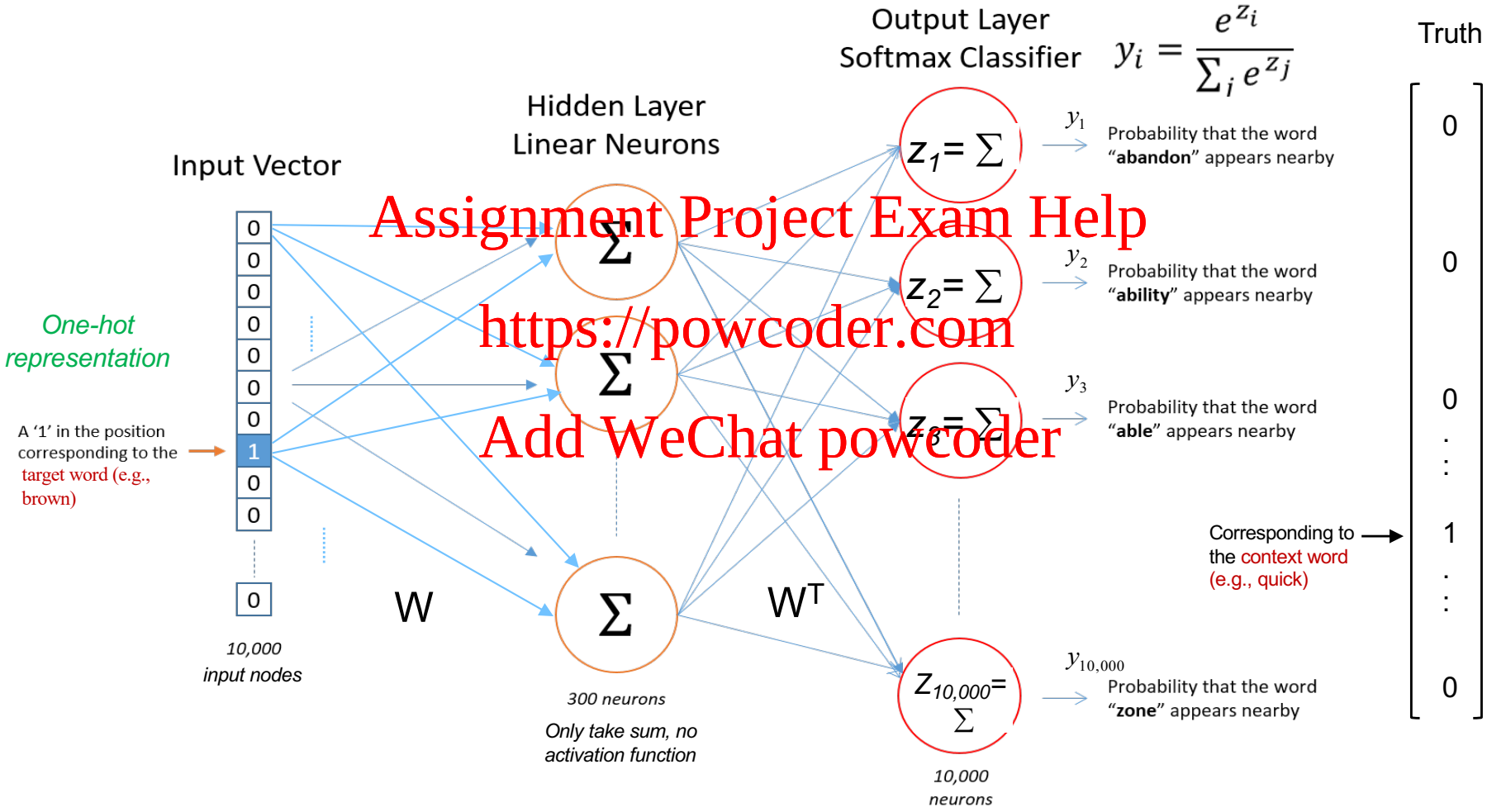
Skip-gram

- Learn to predict the context words from a target word based on a training corpus
  - Target word moves from the beginning to the end of the corpus
  - Context words are the k words before or after the target word (k is the context window size)
  - Training data contains (target word, context word) pairs:



Skip-gram Model

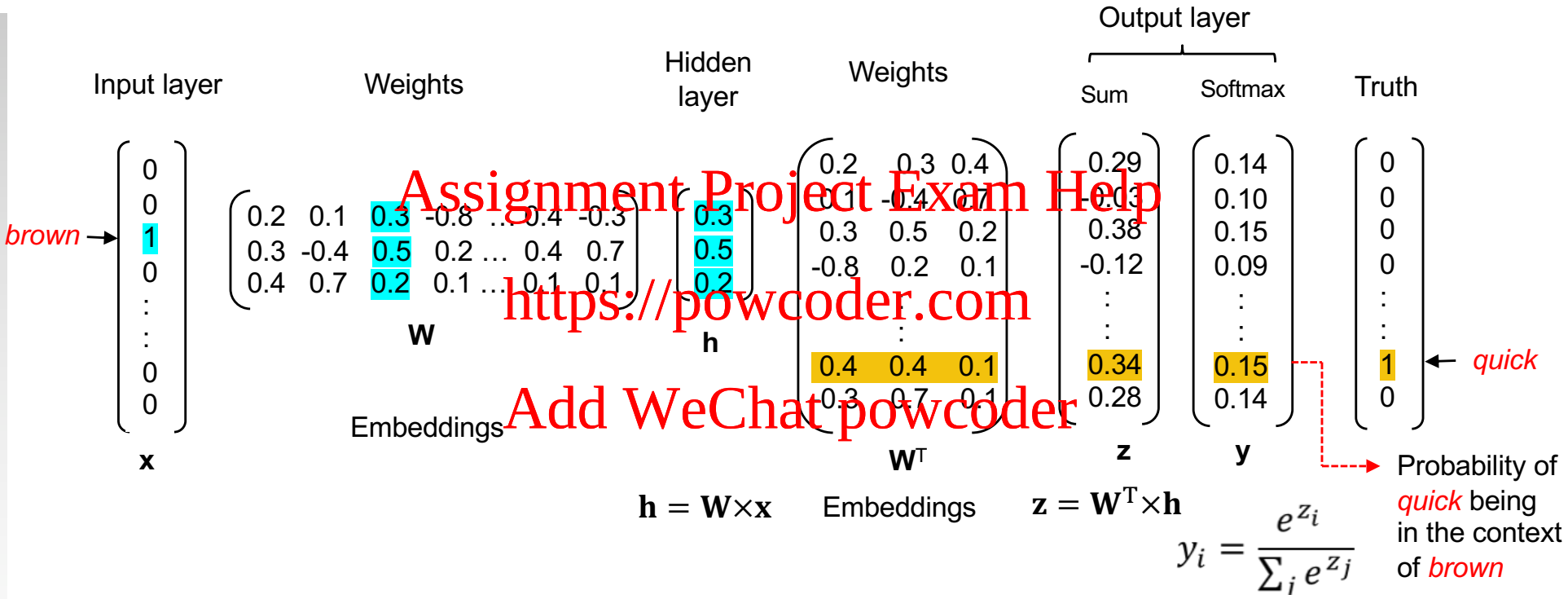
Assume there are 10,000 words in the vocabulary:



- After training, the weights are the word embeddings
- Dimension of the word vector is the number of neurons in the hidden layer

Skip-gram Model

Example: assume training example is (*brown*, *quick*)

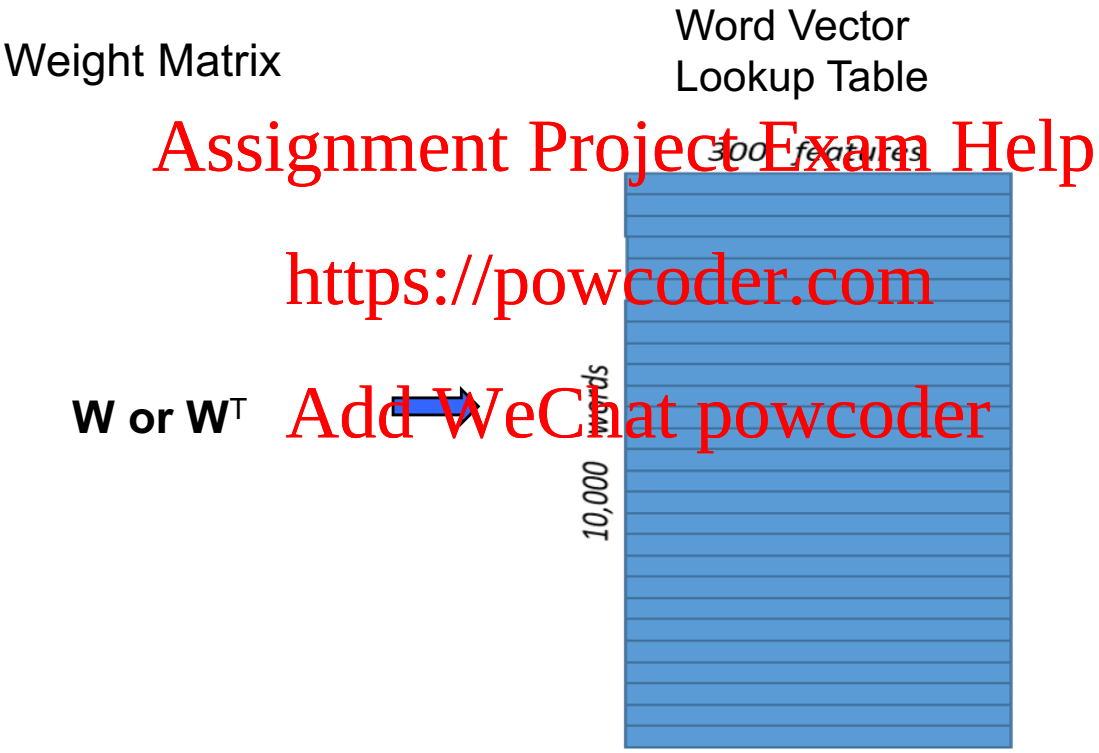


$$y_i = \frac{e^{z_i}}{\sum_i e^{z_j}}$$

- z_i is the dot product of the target and context word embeddings, representing the similarity between the two words
- Such training tries to make words close to each other in the text corpus have similar embeddings

Word Vectors: weights

Assume there are 10,000 words in the vocabulary and 300 hidden neurons:

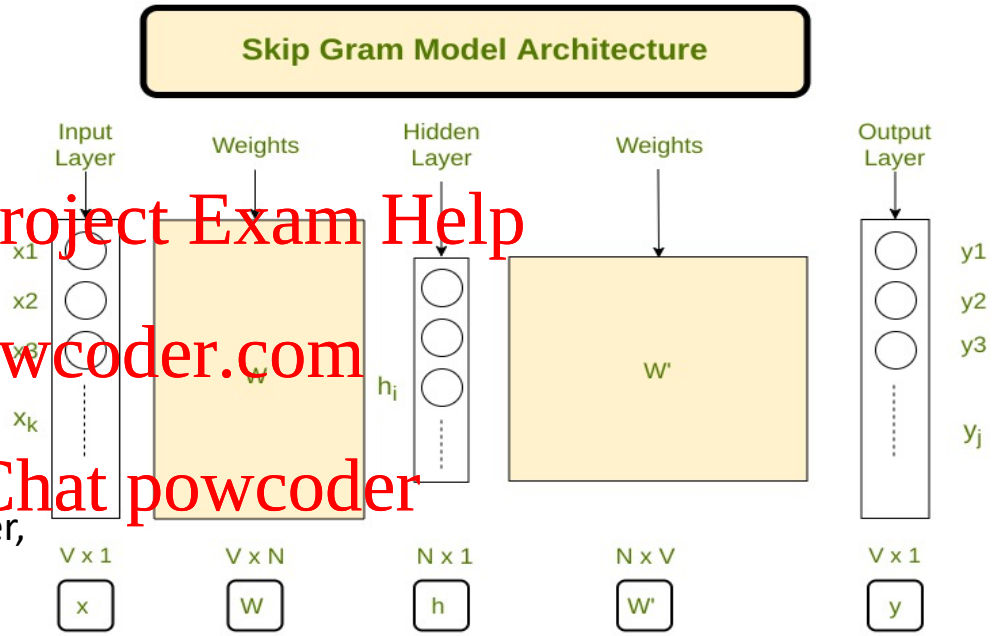


Dimension of the word vector is the number of neurons in the hidden layer

General Picture of Skip-gram

Notations:

- **V** → # of unique words in corpus
- **x** → Input layer
- **N** → # of neurons in hidden layer of NN
- **W** → Weights btw input layer and hidden layer
- **W'** → Weights btw hidden layer and output layer, transpose of **W**
- **y** → Softmax output



$$h' = x'W$$

It assumes words close to each other are semantically related by making the embedding of a word similar to the vectors of its context words,

Usefulness of Word Embeddings

- Can be used to compute similarity between words

$$\text{cosine\_similarity}(w_i, w_j) = \frac{\vec{w}_i \bullet \vec{w}_j}{\|\vec{w}_i\|\|\vec{w}_j\|}$$

where w_i and w_j are words and $\vec{w_i}$ and $\vec{w_j}$ are word vectors

  - Useful for tasks, such as text matching, information retrieval, etc.
- Building block for longer text embedding

  - Sentence embedding (Sent2Vec)
  - Paragraph/Document Embedding (doc2vec)
- Used in many other NLP tasks, such as
  - Text classification
  - Machine translation
  - Summarization)
  - …

Text Classification with Word Embedding

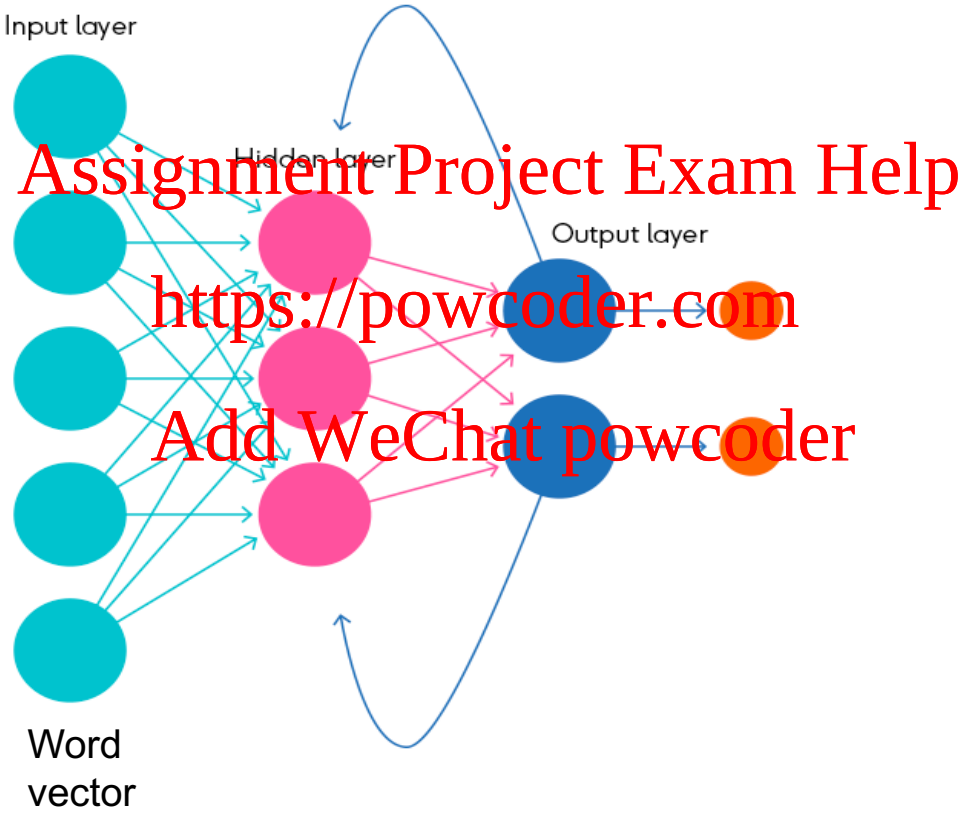
- Objective is to learn to classify a document
- Key is to represent a document with word embeddings
- There are a few ways to do so, e.g.,
  1) Compute a *document vector* with word embeddings
  2) Directly using *the sequence of word embeddings* to train and classify a document with *recurrent neural networks*.
  3) Train *document embeddings* directly (e.g., doc2vec, Doc2VecC, GPT, BERT, etc)
- We will only briefly describe the first two

Methods for Computing Document Vector with Word Embeddings

- Simple averaging on word embeddings
  - Average the embeddings of the words occurring in the document
- TF-IDF weighted averaging on word embeddings
  - Using the TF-IDF value of a word in the document as the weight for the word when averaging word embeddings
- Word embeddings
  - Can be pre-trained with a large text corpus
  - Fine-tuned with the training data
  - Freshly trained with the training data

Recurrent Neural Network (RNN) with Word Embeddings

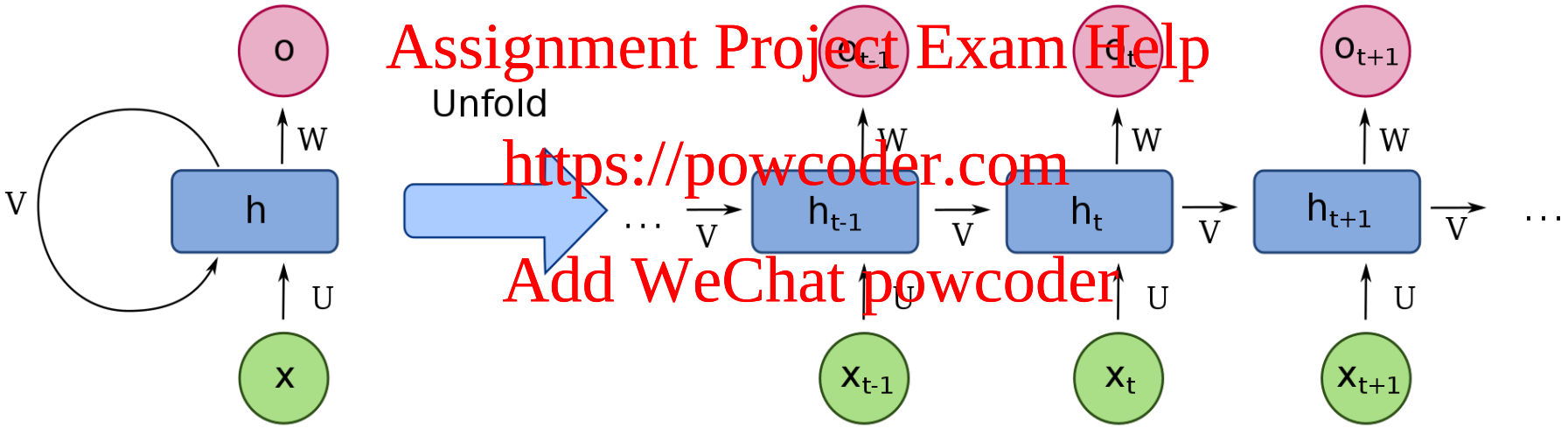
- Recurrent neural network:



- Words in a document are inputted to RNN sequentially.
- Each word is represented by its embedding
- Output indicates the class of the document.

Recurrent Neural Network with Word Embeddings

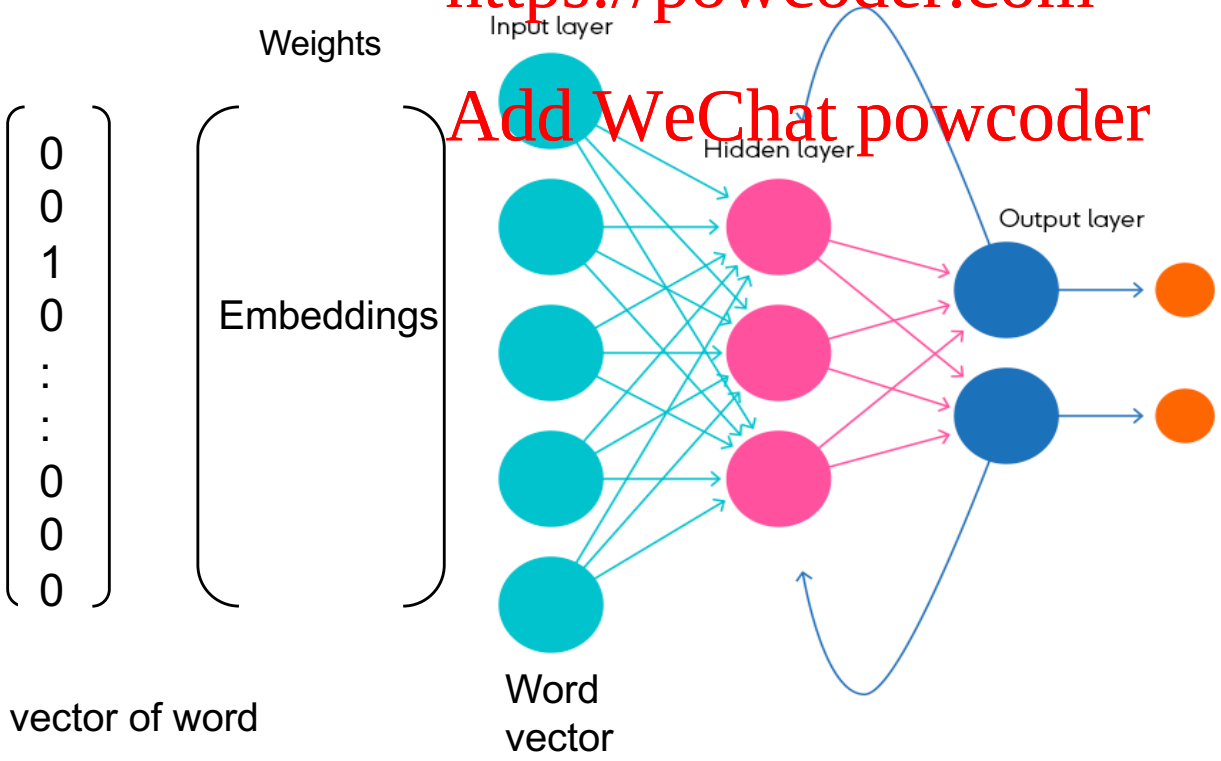
- Unfolded recurrent neural network:



x_i is a word

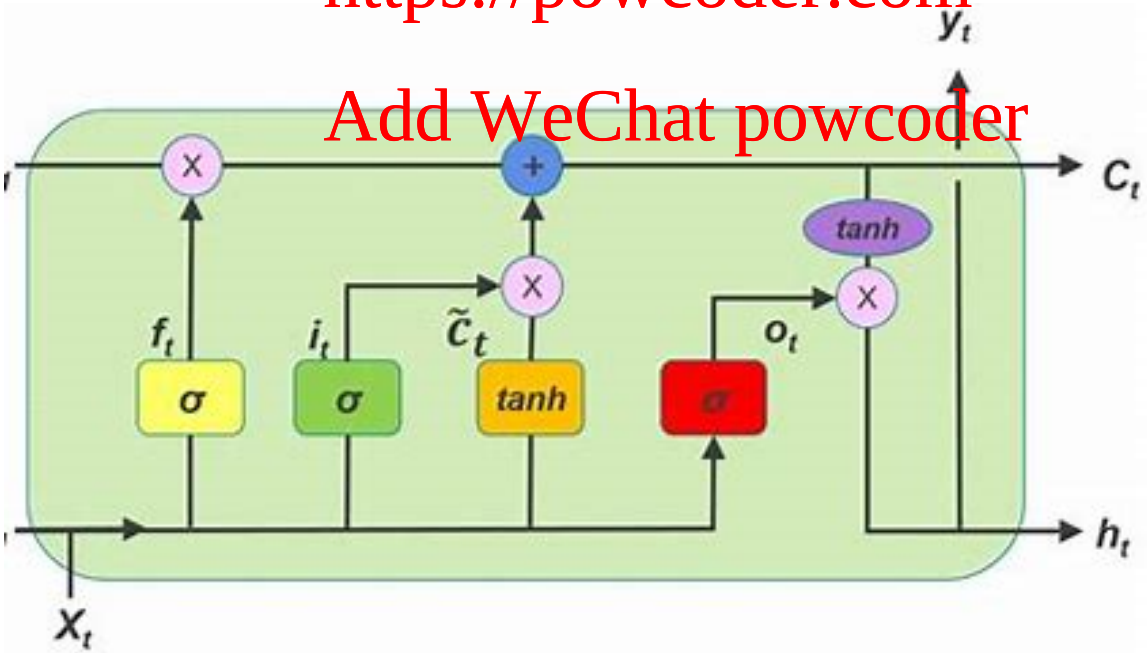
Recurrent Neural Network (RNN) with Word Embeddings

- Word embeddings
  - Can be pre-trained with a large text corpus
  - Fine-tuned with the training data
  - Freshly trained with the training data



Long Short Term Memory networks

- A popular RNN is LSTM
  - Capable of bridging long time lags between inputs
  - Able to remember inputs from up to 1000 time steps



Summary

- Text classification has many applications
- The most important issue is how to represent documents
  - Word extraction
  - Stop word removal
  - Stemming
  - Feature selection
  - Represent document with values of the selected features (e.g., the frequency of the word in the document).
  - Advanced methods for text representation based on word embeddings