W08D3 – NLP I

Instructor: Brian Lynch

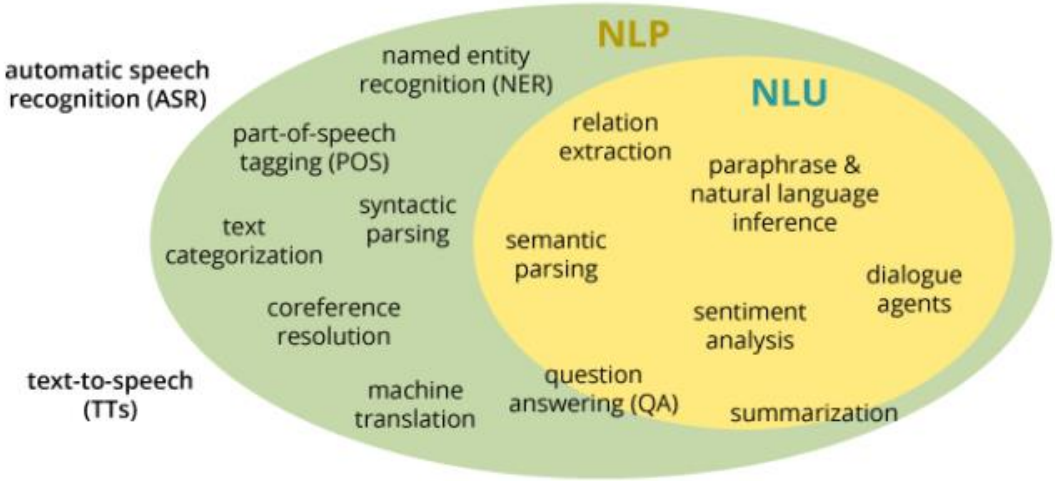
Credit: Zain Hasan, Jeremy Eng

Outline

- Introduction to NLP
- Data Prep in NLP
  - Simplifying text
  - Punctuation removal, tokenization, stop words, stemming, lemmatization
- Representing Text with Numbers
  - Bag of Words (BoW), term frequency-inverse document frequency (TF-IDF)
  - Word2Vec
- Demo

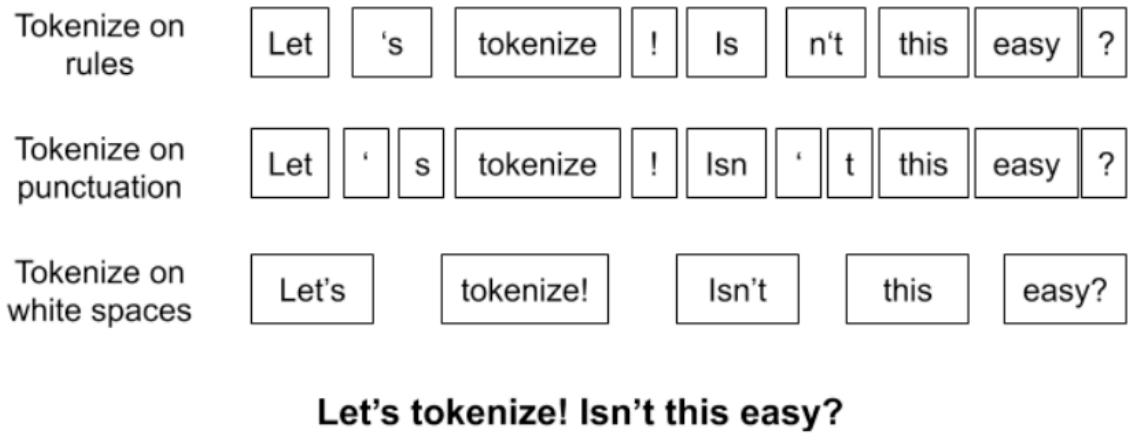
Introduction to Natural Language Processing (NLP)

- NLP involves processing and analyzing large amounts of natural language speech/text.
  - Not solved; very hard problem.
- NLU = Natural language understanding
- Some well-known applications:
  - Virtual assistants (Siri, Alexa, Cortana)
  - Chat bots
  - Predictive text
  - Email filters
  - Sentiment analysis
  - Topic modeling
- NLP has been around for a while, we just now have better tools.
  - Deep learning, more data, stronger computing power, lower costs.



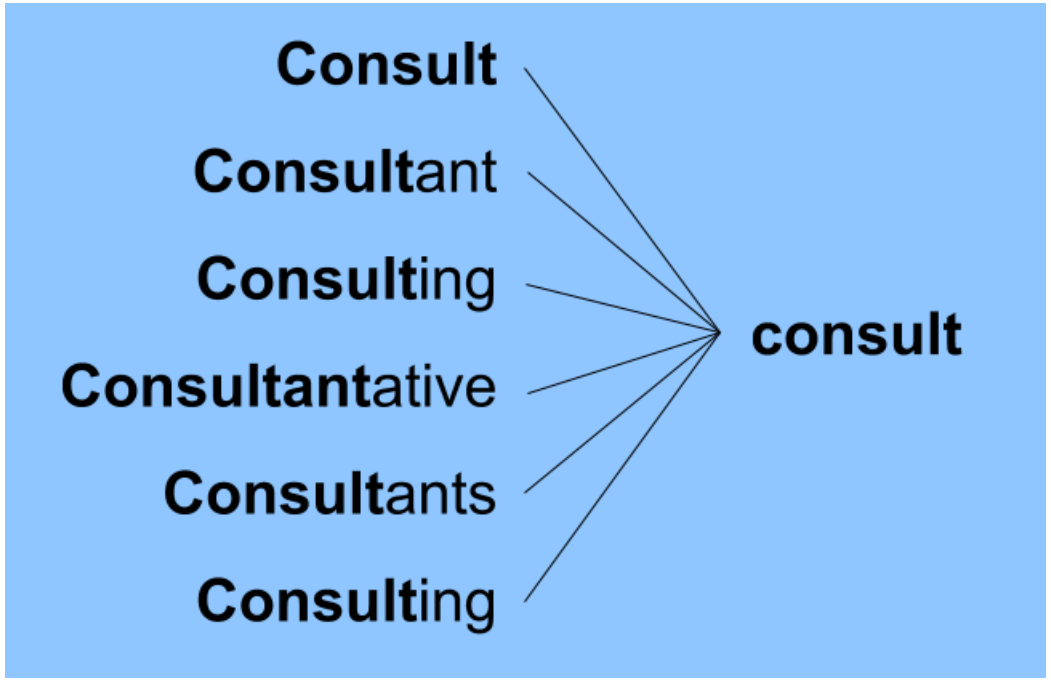
Simplifying Text

- Today will just focus on pre-processing text.
  - Simplifying text
  - Converting from text into numerical values
- Text simplification examples:
  - Remove punctuation
  - Remove common words (the, a, and)
    - Aka “stop words” or “glue words”
  - Remove capitalization
  - Tokenize
  - Stemming/lemmatization
- NLTK package (and others) in Python



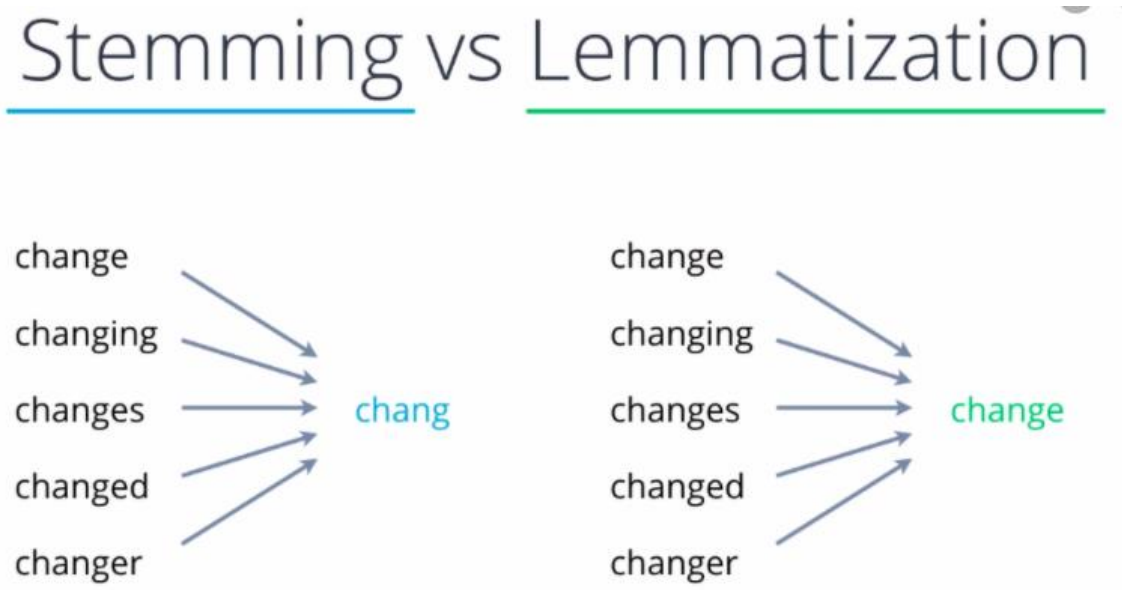
Simplifying Text

- Today will just focus on pre-processing text.
  - Simplifying text
  - Converting from text into numerical values
- Text simplification examples:
  - Remove punctuation
  - Remove common (“glue”) words (the, a, and)
    - Aka “stop words”
  - Remove capitalization
  - Tokenize
  - Stemming/lemmatization
- NLTK package (and others) in Python



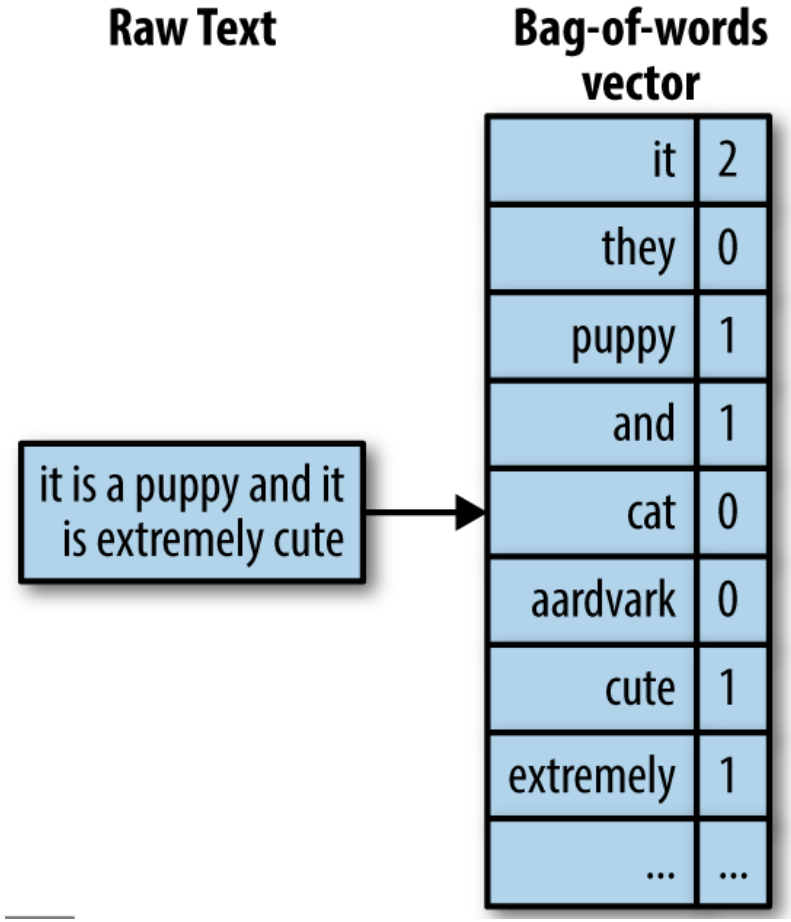
Simplifying Text

- Today will just focus on pre-processing text.
  - Simplifying text
  - Converting from text into numerical values
- Text simplification examples:
  - Remove punctuation
  - Remove common (“glue”) words (the, a, and)
    - Aka “stop words”
  - Remove capitalization
  - Tokenize
  - Stemming/lemmatization
- NLTK package (and others) in Python



Vectorizing Text

- After simplifying our text, we can represent the text using vectors.
- Bag of Words
  - Simply count how often the word appears (problems?).
- Text Frequency–Inverse Document Frequency (TF-IDF)
  - Like BoW, but weighted by how rare the word is across all documents.
- Word2Vec: uses a neural network to learn word associations.
  - Each word is represented by a VECTOR
  - Similar words are close together. Direction between words can have meaning.



Vectorizing Text

- After simplifying our text, we can represent the text using vectors.
- Bag of Words
  - Simply count how often the word appears (problems?).
- Text Frequency–Inverse Document Frequency (TF-IDF)
  - Like BoW, but weighted by how rare the word is across all documents.
- Word2Vec: uses a neural network to learn word associations.
  - Each word is represented by a VECTOR
  - Similar words are close together. Direction between words can have meaning.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Vectorizing Text

- After simplifying our text, we can represent the text using vectors.
- Bag of Words
  - Simply count how often the word appears (problems?).
- Text Frequency–Inverse Document Frequency (TF-IDF)
  - Like BoW, but weighted by how rare the word is across all documents.
- Word2Vec: uses a neural network to learn word associations.
  - Each word is represented by a VECTOR
  - Similar words are close together. Direction between words can have meaning.

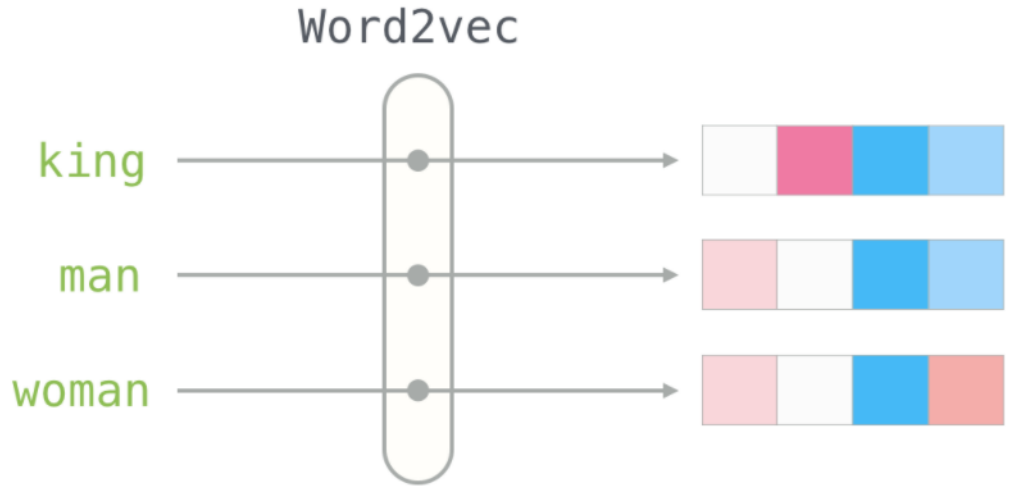
Sentence A : The car is driven on the road.

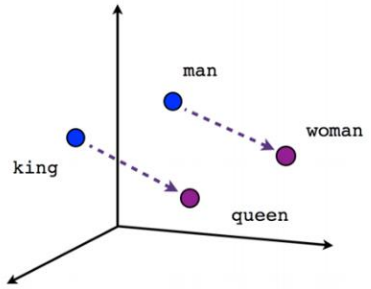
Sentence B: The truck is driven on the highway.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

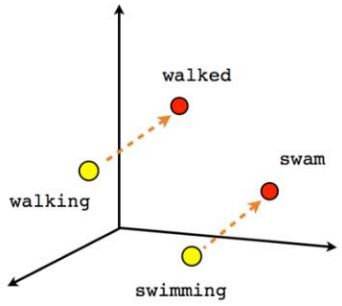
Vectorizing Text

- After simplifying our text, we can represent the text using vectors.
- Bag of Words
  - Simply count how often the word appears (problems?).
- Text Frequency–Inverse Document Frequency (TF-IDF)
  - Like BoW, but weighted by how rare the word is across all documents.
- Word2Vec: uses a neural network to learn word associations.
  - Each word is represented by a VECTOR
  - Similar words are close together. Direction between words can have meaning (add/subtract).

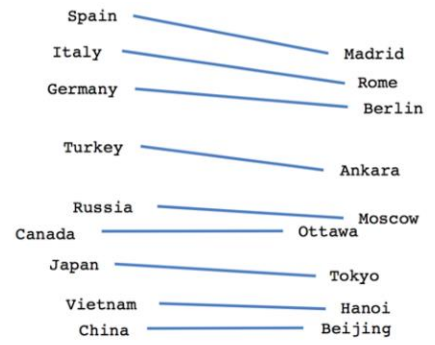




Male-Female



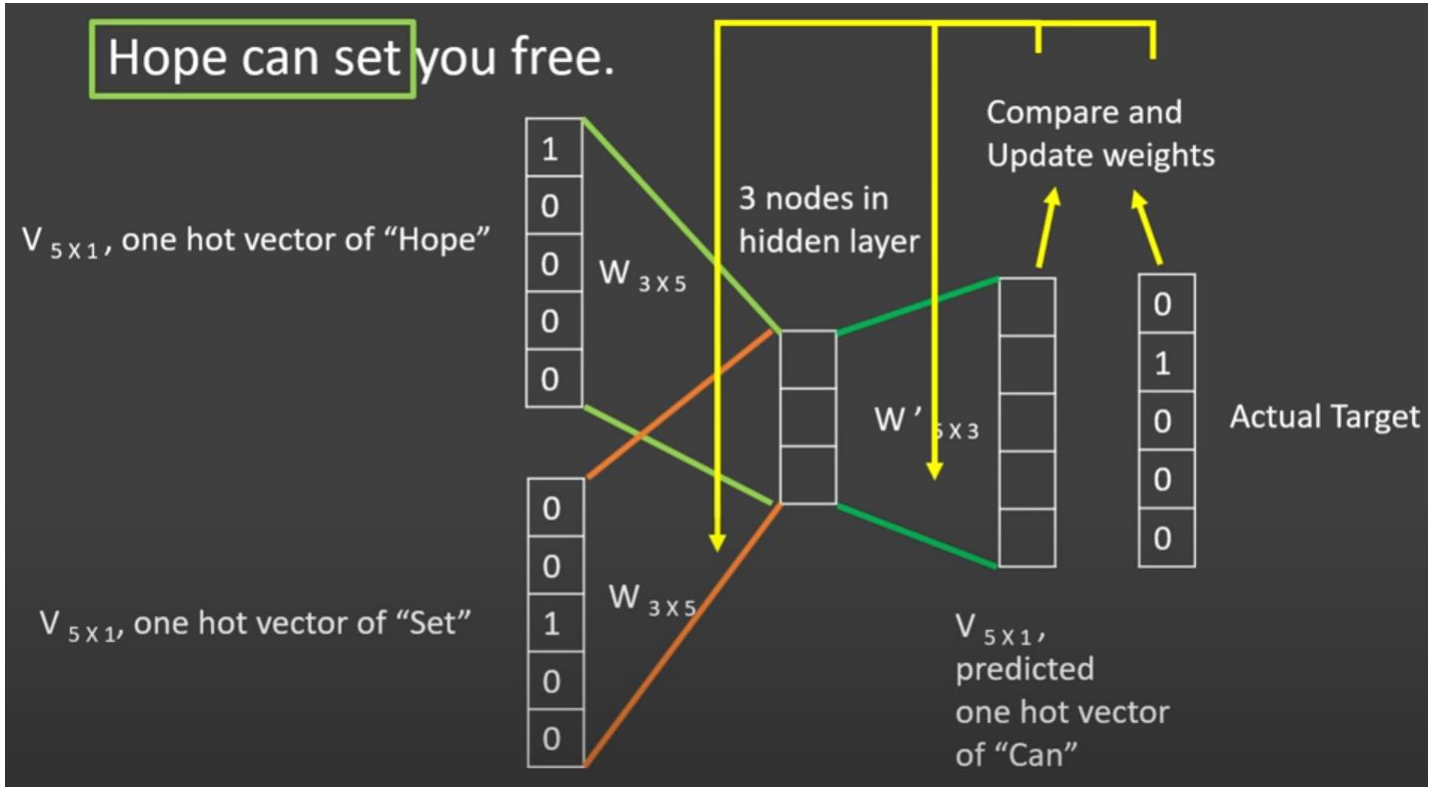
Verb tense



Country-Capital

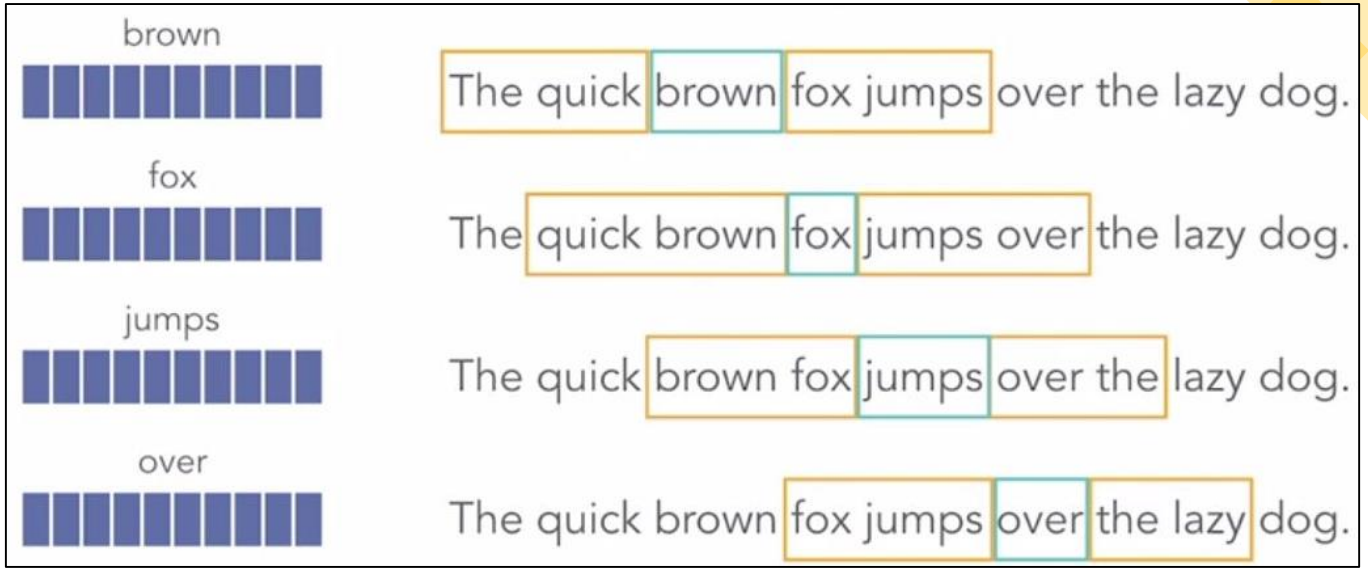
Word2Vec Architectures

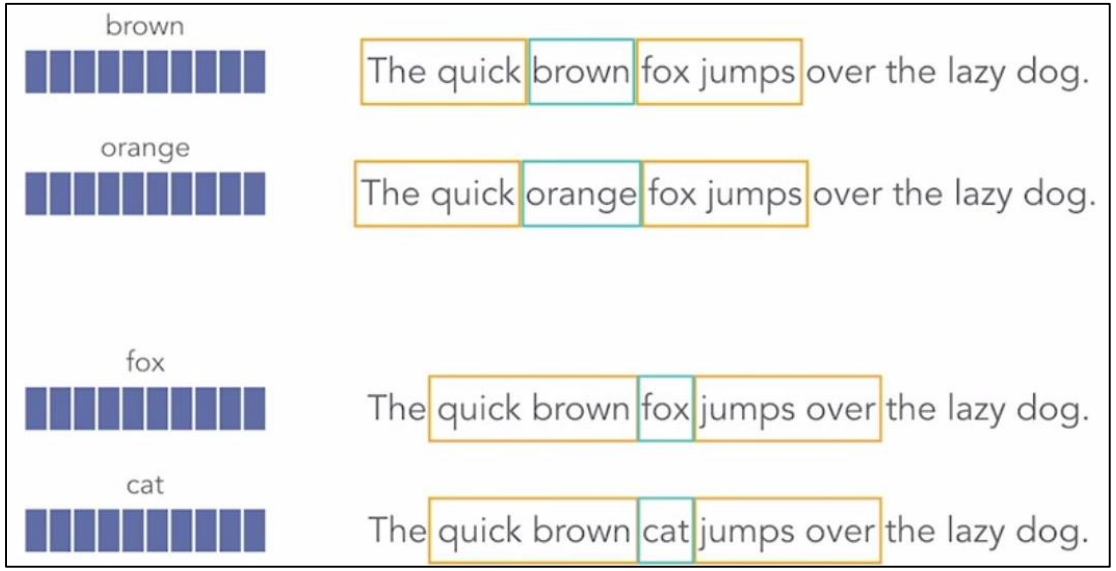
- Continuous Bag of Words (CBoW)
  - Tries to predict the **target word** (center) based on the **surrounding words** (context).



Word2Vec Architectures

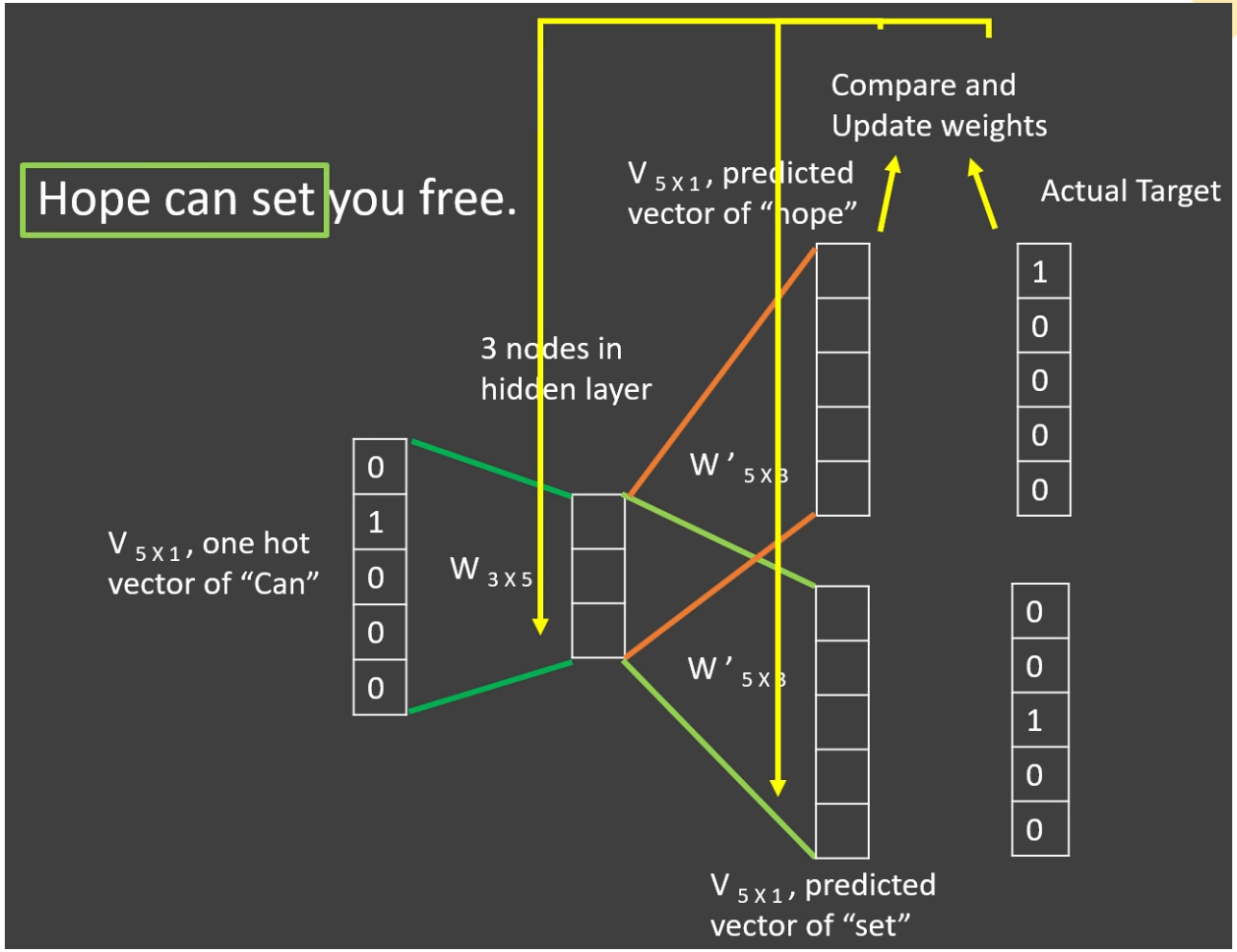
- Continuous Bag of Words (CBoW)
  - Tries to predict the **target word** (center) based on the **surrounding words** (context).





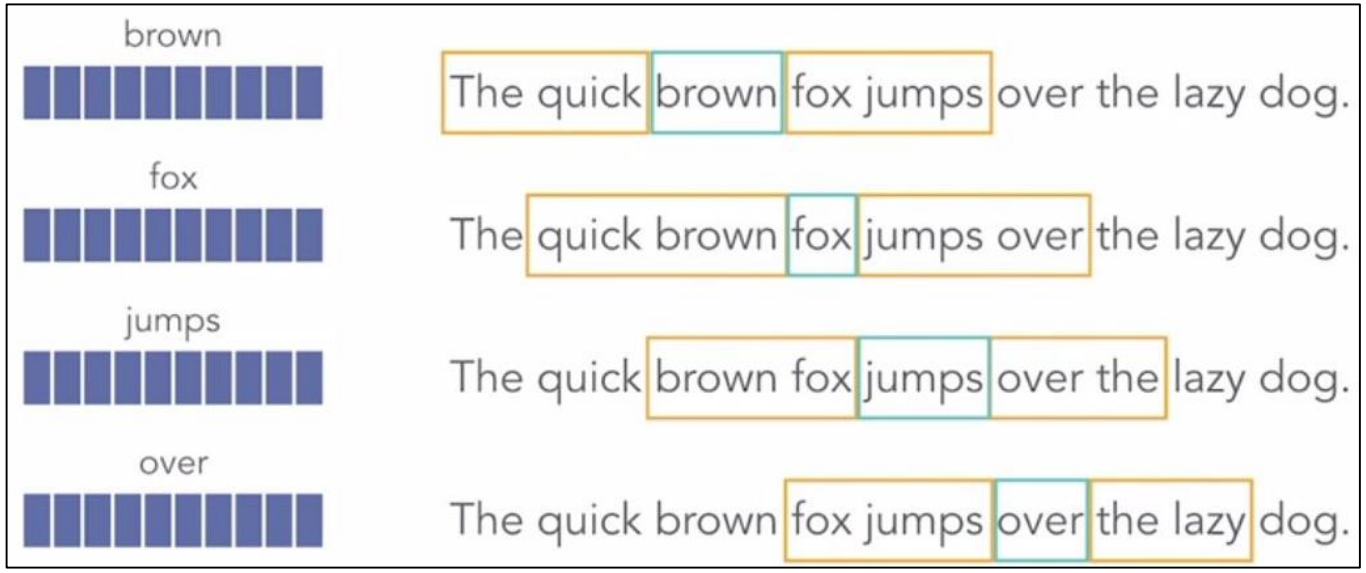
Word2Vec Architectures

- Continuous Bag of Words (CBoW)
  - Tries to predict the **target word** (center) based on the **surrounding words** (context).
- Skip-gram
  - Tries to predict the **surrounding words** (context) based on the **target word**.



Word2Vec Architectures

- Continuous Bag of Words (CBoW)
  - Tries to predict the **target word** (center) based on the **surrounding words** (context).
- Skip-gram
  - Tries to predict the **surrounding words** (context) based on the **target word**.



NLP Pre-processing Summary

- NLP involves processing and analyzing large amounts of natural language speech/text.
  - Not solved; very hard problem.
- Today we just focused on pre-processing text (simplifying, vectorizing)
  - Tomorrow: NLP applications (sentiment analysis and topic modeling)
- Simplifying:
  - Punctuation removal, tokenization, stop words, stemming, lemmatization
- Vectorizing:
  - BoW, TF-IDF, Word2Vec
- Jupyter notebook demo