INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Natalie Parde
parde@uic.edu

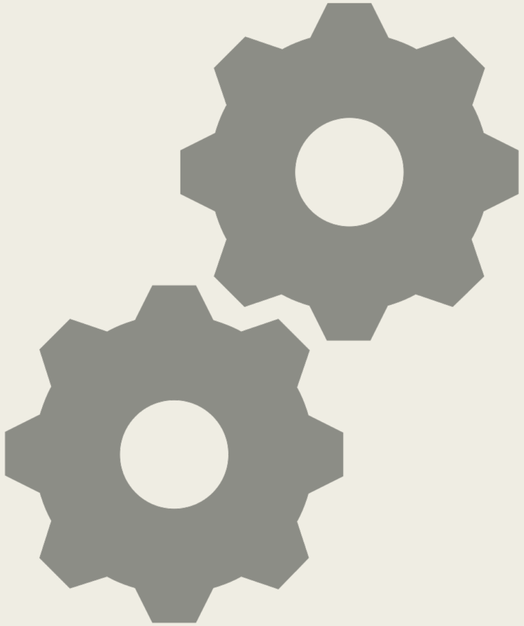
CS 594: Language and Vision
Spring 2019

What is natural language processing?

The study of language and linguistic interactions from a computational perspective, enabling the development of algorithms and models capable of (a) **natural language understanding (NLU)** and (b) **natural language generation (NLG).**

Core Terminology

- N-gram
- Corpus
- Text Normalization
- POS Tagging
- Dependency Parsing
- Stemming
- Lemmatization



N-gram

- A unit of text, of length *n*
- Most common n-grams:
  - *Unigram*
  - *Bigram*
  - *Trigram*
  - *4-gram*
  - *5-gram*
- Text unit can be an individual character (useful for language identification) or an individual word (useful for text classification)

Corpus

- Plural: Corpora
- Synonym: Dataset
- A generally large (although this is not a requirement) collection of text or speech data, used to train machine learning models for natural language processing tasks.
- Example Corpora:
  - *Google Books N-gram Corpus:* *http://storage.googleapis.com/books/ngrams/books/datasetsv2.html*
  - *British National Corpus:* *http://www.natcorp.ox.ac.uk/*
  - *Metaphor Novelty Dataset:* *http://hilt.cse.unt.edu/resources.html#metaphor_novelty_dataset*

Text Normalization

- A sequence of actions applied to unstructured text to convert it into a more useful form for further text processing.
- Often includes:
  - *Sentence segmentation*
  - *Tokenization*
- Depending on task:
  - *Punctuation removal*
  - *Contraction handling*
  - *URL removal*
  - *Case adjustment*

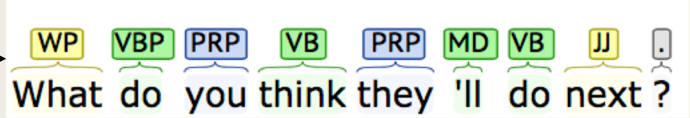
What do you think they'll do next?

what, do, you, think, they, will, do, next

POS Tagging

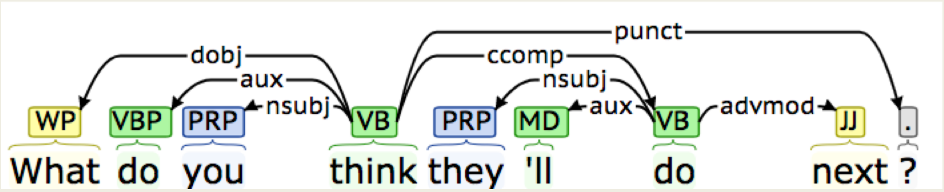
- Full term: Part-of-Speech Tagging
- Automatically tagging a token of text with its syntactic part of speech
  - *Part of speech: a category of words with similar grammatical properties (e.g., nouns)*
- Common POS tag sets:
  - *Penn Treebank: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html*
  - *Universal POS Tags: https://universaldependencies.org/u/pos/*

What do you think they'll do next?



Dependency Parsing

- Automatically tagging pairs of syntactically related words with their grammatical relation type
- Common dependency type sets:
  - *Universal Dependencies: https://universaldependencies.org/*
  - *Stanford Dependencies: https://nlp.stanford.edu/software/dependencies_manual.pdf*
- Differs from a **constituency parser**:
  - *Dependency parser: Word pairs are connected based on syntactic relationship*
  - *Constituency parser: Text is broken into a hierarchical tree of subphrases, with terminal nodes corresponding to words in the source text and non-terminal nodes corresponding to phrase types*



Stemming

- Removing identified prefixes and suffixes from words
  - *Flies → fli*
  - *Mules → mule*
  - *Agreed → agre*
  - *Owned → own*
  - *Traditional → tradit*
- Can be done heuristically without a dictionary

Lemmatization

- Reduces words to their base form
  - *Flies* → *fly*
  - *Mules* → *mule*
  - *Agreed* → *agree*
  - *Owned* → *own*
  - *Traditional* → *tradition*
- Requires a dictionary
- Lemmatizers are more accurate and useful than stemmers, but also much more difficult to implement!

Popular Tools for Core NLP Tasks

- **Stanford CoreNLP**
  - https://stanfordnlp.github.io/CoreNLP/
- **NLTK**
  - https://www.nltk.org/book/
- **spaCy**
  - https://spacy.io/

Two Main Approaches to NLP

RULE-BASED

STATISTICAL

Rule-based Natural Language Processing

- Extracting information from text patterns
- Generating language by filling slots
- Pros:
  - *Quicker to implement*
  - *Produces "normal-sounding" text in target scenarios*
  - *No processing overhead*
- Cons:
  - *Heavily constrained*
  - *Only works in target scenarios*
  - *Difficult to generalize across domains*

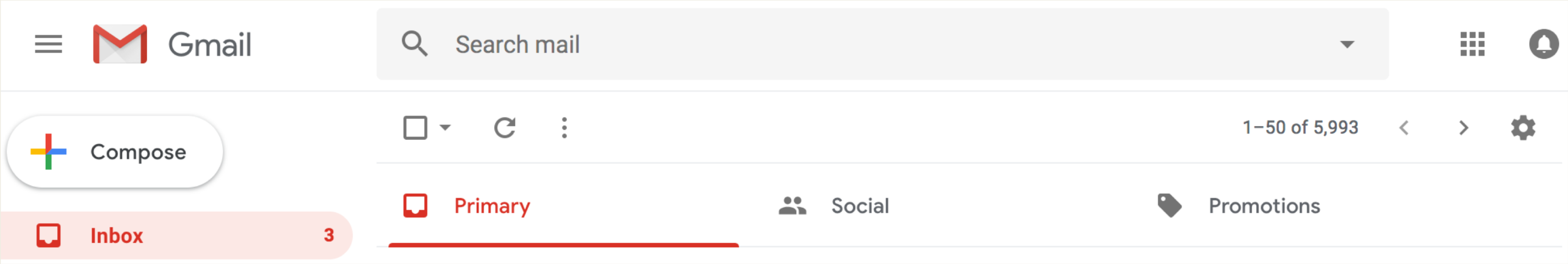
Rule-based Natural Language Processing

- Often used with chatbots
- Classic use case: Eliza
  - https://www.masswerk.at/elizabot/
  - https://youtu.be/RMK9AphfLco

ELIZA: Is something troubling you ?
YOU: Yes.
ELIZA: You seem to be quite positive.
YOU: I'm really not!
ELIZA: Do you believe it is normal to be really not ?
YOU: No, not really.
ELIZA: Are you saying no just to be negative?
YOU: I'm saying no because I think most people are positive.
ELIZA: Do you know anyone else who is saying no because you think most people are positive ?
YOU: No.
ELIZA: You are being a bit negative.
YOU: That's what I was trying to tell you!
ELIZA: Why do you tell me you were trying to tell I now ?

Statistical Natural Language Processing

- Much more common for most NLP tasks
- Uses machine learning to train models from text corpora and applies those models to make predictions about new text
- Pros:
  - *Scalable*
  - *Higher coverage*
  - *Easier to generalize across domains*
- Cons:
  - *More development overhead*
    - Feature selection, model selection, parameter tuning
  - *Greater CPU, RAM, and storage needs*
  - *Requires training corpus*



STATISTICAL NATURAL LANGUAGE PROCESSING

Countless applications, many of which are pervasive in day-to-day life

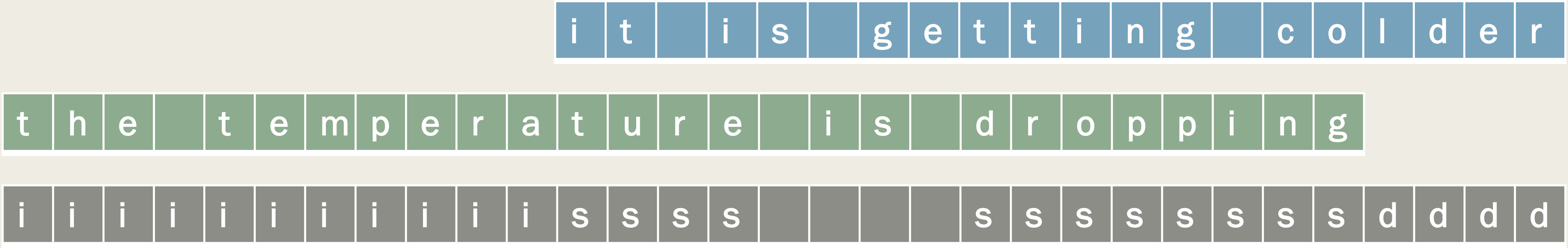
Think, Pair, Share

- Write three possible applications of rule-based NLP and three possible applications of statistical NLP on your notecard
- Share those ideas with a partner
- Choose one example of each to share with the class

- Timer: https://www.google.com/search?q=timer

Text Similarity

- Simplest approach: edit distance
- How many transformations (insertions, deletions, or substitutions) are necessary to change one string into another?



Text Similarity

- Common approach: cosine similarity
- Assuming each word in the vocabulary is represented as a point in space, how similar are the vectors representing two sentences?

	it	is	getting	colder	the	temperature	dropping
S1: It is getting colder.	1	1	1	1	0	0	0
S2: The temperature is dropping.	0	1	0	0	1	1	1

$$\mathrm{sim}(S1, S2) = \frac{\mathrm{S1} \cdot \mathrm{S2}}{\|\mathrm{S1}\|\|\mathrm{S2}\|} = \frac{\sum_{i=1}^{n} \mathrm{S1}_i \mathrm{S2}_i}{\sqrt{\sum_{i=1}^{n} \mathrm{S1}_i^2}\sqrt{\sum_{i=1}^{n} \mathrm{S2}_i^2}}$$

Text Similarity

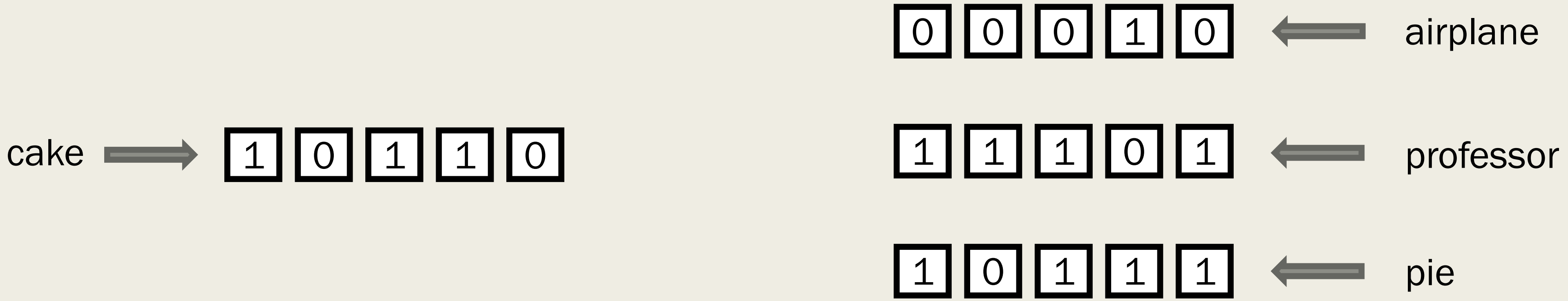
- What are these approaches missing?
  - *Synonyms*
  - *Paraphrases*
  - *These approaches compute lexical similarity, but are ignoring* ***semantic similarity****!*

It is getting colder. ≈ The temperature is dropping.

WORD EMBEDDINGS

Word Embeddings

- Vectors that refer to a word's point in a multidimensional semantic space
- Can be of any size
  - *Most common size: 100 or 300 dimensions*
- Learned automatically from massive text corpora



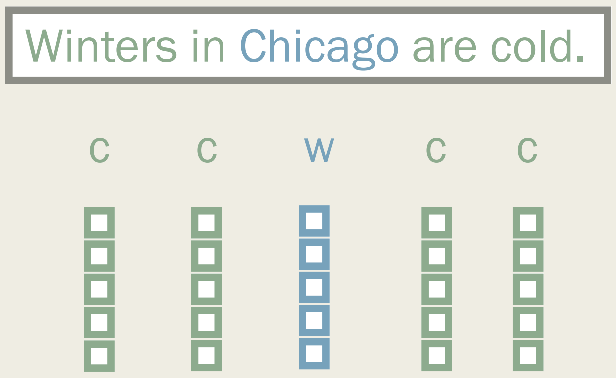
Word Embeddings

- Lots of different varieties
- Some of the most popular:
  - *Word2Vec*
  - *GloVe*
  - *ELMo*

Photo Credit: tiffany terry, https://flic.kr/p/612Hvh

Word2Vec

- Model learns *n*-dimensional embeddings for two types of vectors:
  - *Words*
  - *Contexts*
- Weights for all vectors are initialized to random small numbers
- Weights are updated over time as learning progresses
- When Word2Vec finishes, the weights associated with the word vectors are returned as the **embeddings** and the weights associated with the context vectors are discarded



Word2Vec

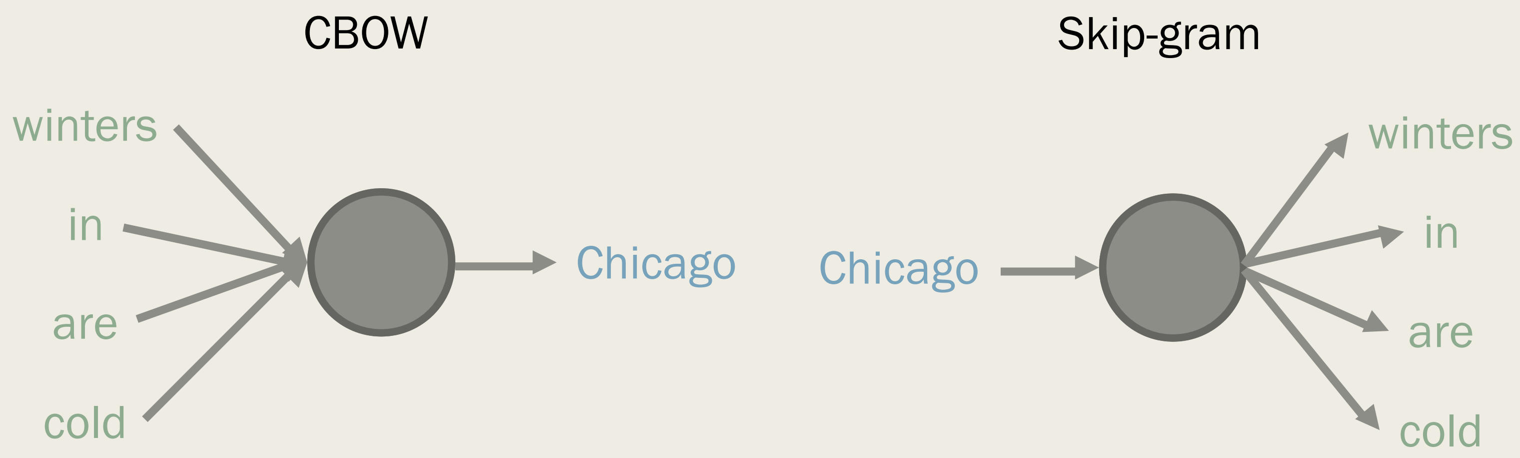
- Word2Vec essentially creates a neural network that we don't really need ...all we're interested in are the learned weights!
- Assuming that σ is an activation function in the output layer of the neural network, and assuming $c_i \in C$ is a context word vector associated with a target word vector w, Word2Vec computes the following:
  - $\sum_{c_i \in C} \sigma(w \cdot c_i)$
- The goal is to learn weights for w and all $c_i \in C$ such that:
  - *The value resulting when w is the target word is high*
  - *The value resulting when w is not the target word is low*

Winters in Chicago are cold.

Winters in Phoenix are cold.

Word2Vec

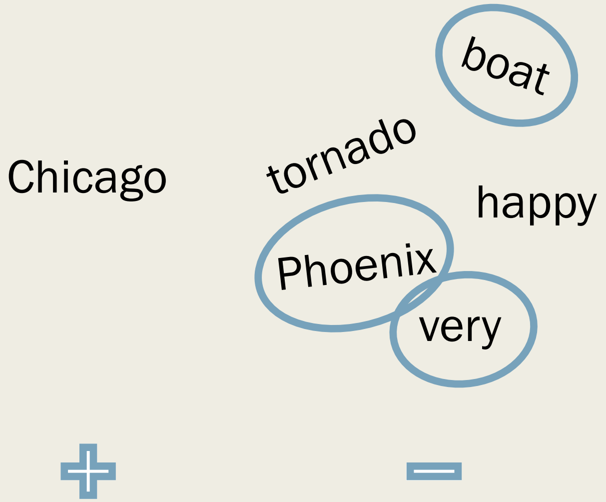
- Embeddings can be learned in one of two ways:
  - *Continuous Bag of Words (CBOW): Predict a word given a context*
  - *Skip-gram: Predict a context given a word*



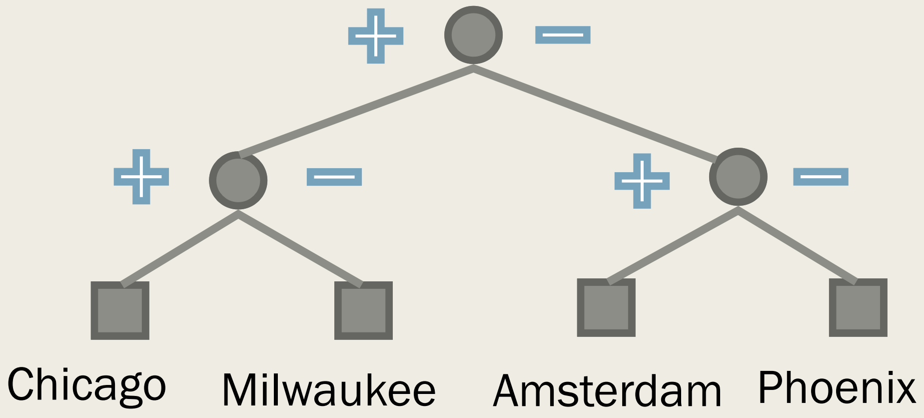
Word2Vec

- The weights can also be updated using a couple different strategies:
  - *Negative Sampling: Randomly sample negative target words rather than computing values for all possible target words*
  - *Hierarchical Softmax: Iterate through a binary tree in which nodes are weight vectors and leaves are target words—learn weights that are close to those on the correct path to the target word*

Negative Sampling



Hierarchical Softmax



Count-based Embedding Models

It is freezing cold. Winters in Chicago are cold. Winters in Phoenix are warm.

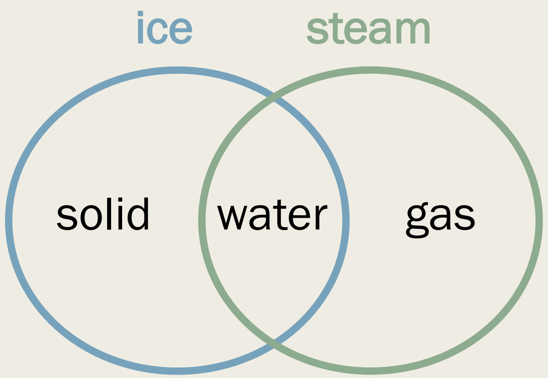
	it	is	freezing	cold	winters	in	chicago	are	phoenix	warm
it	0	1	1	1	0	0	0	0	0	0
is	1	0	1	1	0	0	0	0	0	0
freezing	1	1	0	1	0	0	0	0	0	0
cold	1	1	1	0	1	1	1	1	0	0
winters	0	0	0	1	0	2	1	2	1	1
in	0	0	0	1	2	0	1	2	1	1
chicago	0	0	0	1	1	1	0	1	0	0
are	0	0	0	1	2	2	1	0	1	1
phoenix	0	0	0	0	1	1	0	1	0	1
warm	0	0	0	0	1	1	0	1	1	0

GloVe

- Co-occurrence matrices quickly grow extremely large
- Intuitive solution to increase scalability → dimensionality reduction
  - *However, typical dimensionality reduction strategies may result in too much computational overhead*
- GloVe combines aspects of predictive models (e.g., Word2Vec) and count-based models
- Learns to predict weights that correspond to the co-occurrence probabilities between words
  - *Specifically: The dot product between two words' vectors should equal the logarithm of their probability of co-occurrence*

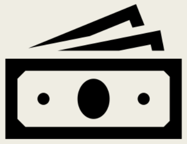
GloVe

- Why is this useful?
  - *Predictive models → black box*
    - They work, but why?
  - *GloVe models are easier to interpret*
- GloVe models also encode the ratios of co-occurrence probabilities between different words …this makes these vectors useful for word analogy tasks



ELMo

- Full Term: Embeddings from Language Models
- Accepts character inputs instead of words, which enables the model to predict embeddings for out-of-vocabulary words
- Concatenates information from multiple layers of a bidirectional **language model**
  - *A model that predicts the next word in a sequence of words, given the words that precede it*
- This allows ELMo to store multiple representations of the same word!
- Predicts an embedding for a target word given its context



bank ← 0.2 | 0.3 | 0.1 | 0.2 | 0.5

0.1 | 0.1 | 0.6 | 0.2 | 0.4 → bank

Which embeddings are best?

- It depends on your data!
- In general, Word2Vec and GloVe produce similar embeddings
- Word2Vec → slower to train but less memory intensive
- GloVe → faster to train but more memory intensive
- Word2Vec and Glove both produce context-independent embeddings
- ELMo produces context-dependent embeddings
- ELMo can predict embeddings for new words

Think, Pair, Share

- Write three possible use cases for word embeddings on your notecard. For each one, indicate what type of word embeddings you think would work best for the task:
  - *Word2Vec, GloVe, or ELMo? (Some other type of embedding entirely?)*
  - *What dataset would they be trained on? (Google Books? Wikipedia? A corpus of news articles? Something else?)*
  - *If GloVe, any preference between (CBOW X Skip-gram) X (Negative Sampling X Hierarchical Softmax)?*
- Discuss these use cases with a partner. Did your partner propose different word embeddings for a similar task?
- Share some clear agreements or differences of opinion with the class.
- Timer: https://www.google.com/search?q=timer

You can experiment with all of these embedding models yourself!

Word2Vec:

Code:
[*https://github.com/tmikolov/word2vec*]

Pretrained Embeddings:
[*https://code.google.com/archive/p/word2vec/*]

GloVe:

Code:
[*https://github.com/stanfordnlp/GloVe*]

Pretrained Embeddings: (same website)

ELMo:

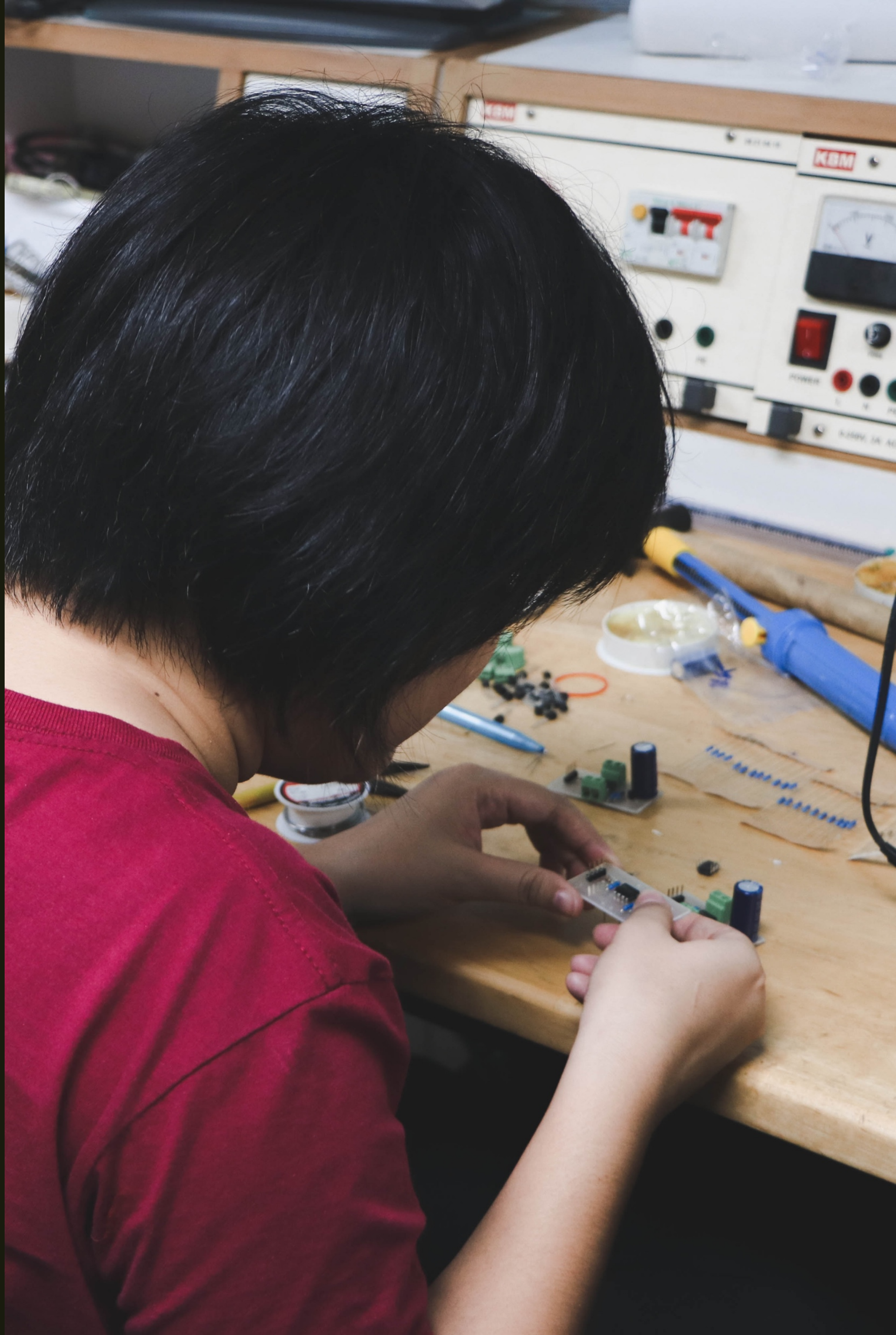
Code (AllenNLP version):
[*https://github.com/allenai/allennlp/blob/master/tutorials/how_to/elmo.md*]

Code (TensorFlow version):
[*https://github.com/allenai/bilm-tf*]

Pretrained Embeddings:
[*https://allennlp.org/elmo*]

NLP Features

- Two types:
  - *Implicitly learned*
  - *Engineered*



Implicitly Learned Features

Word Embeddings

Topic Models

- Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation

- Generative probabilistic model that considers two units:
  - *Documents*
  - *Words*
- How it works:
  - *Randomly assign a topic to each word*
  - *For each word, assign a new topic based on the likelihood perceived from the current topic/word distribution*
  - *Repeat until convergence (or until an iteration threshold is met)*

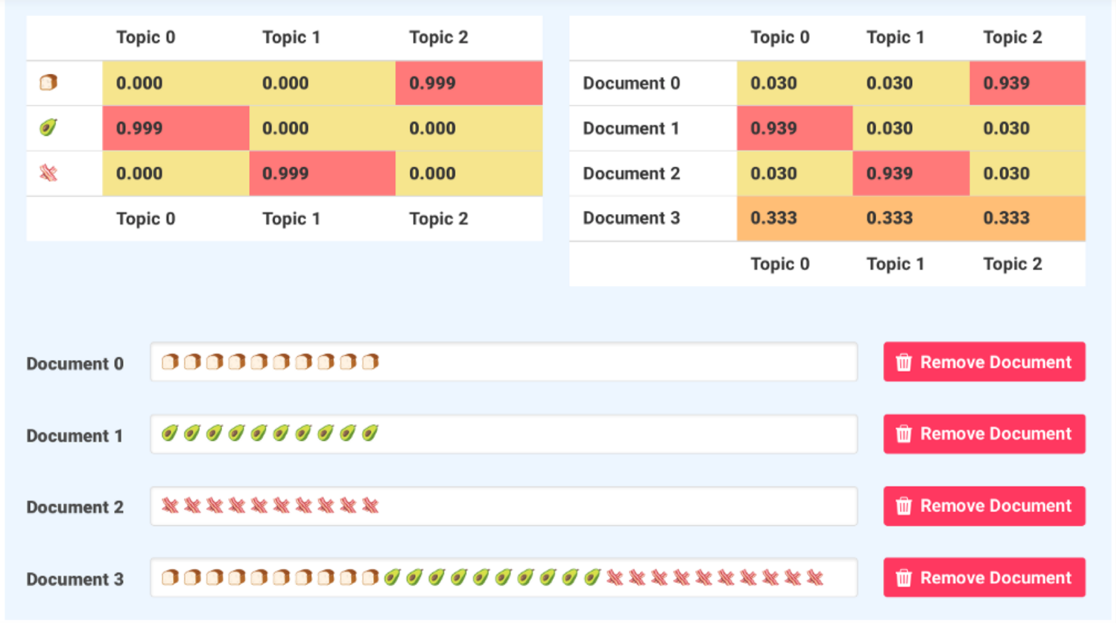
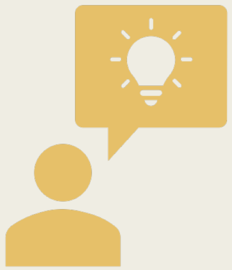


Photo Credit: Lettier, https://medium.com/@lettier/how-does-lda-work-ill-explain-using-emoji-108abf40fa7d

Engineered Features

- Psycholinguistic
  - *Concreteness/Imageability*
  - *Sentiment*
- Count-based
  - *TFIDF*
  - *Pointwise Mutual Information*
- Syntactic
- Lexical

Psycholinguistic Features



Concreteness/Imageability

How easily "imageable" is the target word?

- "mug" ⟶ high imageability
- "idea" ⟶ low imageability

Sentiment

Is this word positive or negative?

- "friendly" ⟶ positive sentiment
- "cruel" ⟶ negative sentiment

Psycholinguistic Resources

- Brysbaert Concreteness Ratings: http://crr.ugent.be/archives/1330
- MRC Psycholinguistic Database: http://websites.psychology.uwa.edu.au/school/MRCDatabase/uwa_mrc.htm
- SentiWordNet: https://sentiwordnet.isti.cnr.it/

(Non-Embedding) Count-based Features

- TFIDF: Term Frequency * Inverse Document Frequency
  - *Computes the ratio between the word's frequency in a specific document and its frequency in a corpus as a whole*
- PMI: Pointwise Mutual Information
  - *Computes the strength of the association between two words*

$$\mathrm{TF}(x) = \frac{\#\text{ times x occurs in document } d}{\#\text{ words in document } d}$$

$$\mathrm{DF}(x) = \frac{\#\text{ documents containing } x}{\text{total } \#\text{ documents}}$$

$$\mathrm{TFIDF}(x) = \mathrm{TF}(x) \times \log \frac{1}{\mathrm{DF}(x)}$$

$$\mathrm{p}(x) = \frac{\#\text{ documents containing } x}{\text{total } \#\text{ documents}}$$

$$\mathrm{p}(x, y) = \frac{\#\text{ documents containing } x \text{ and } y}{\text{total } \#\text{ documents}}$$

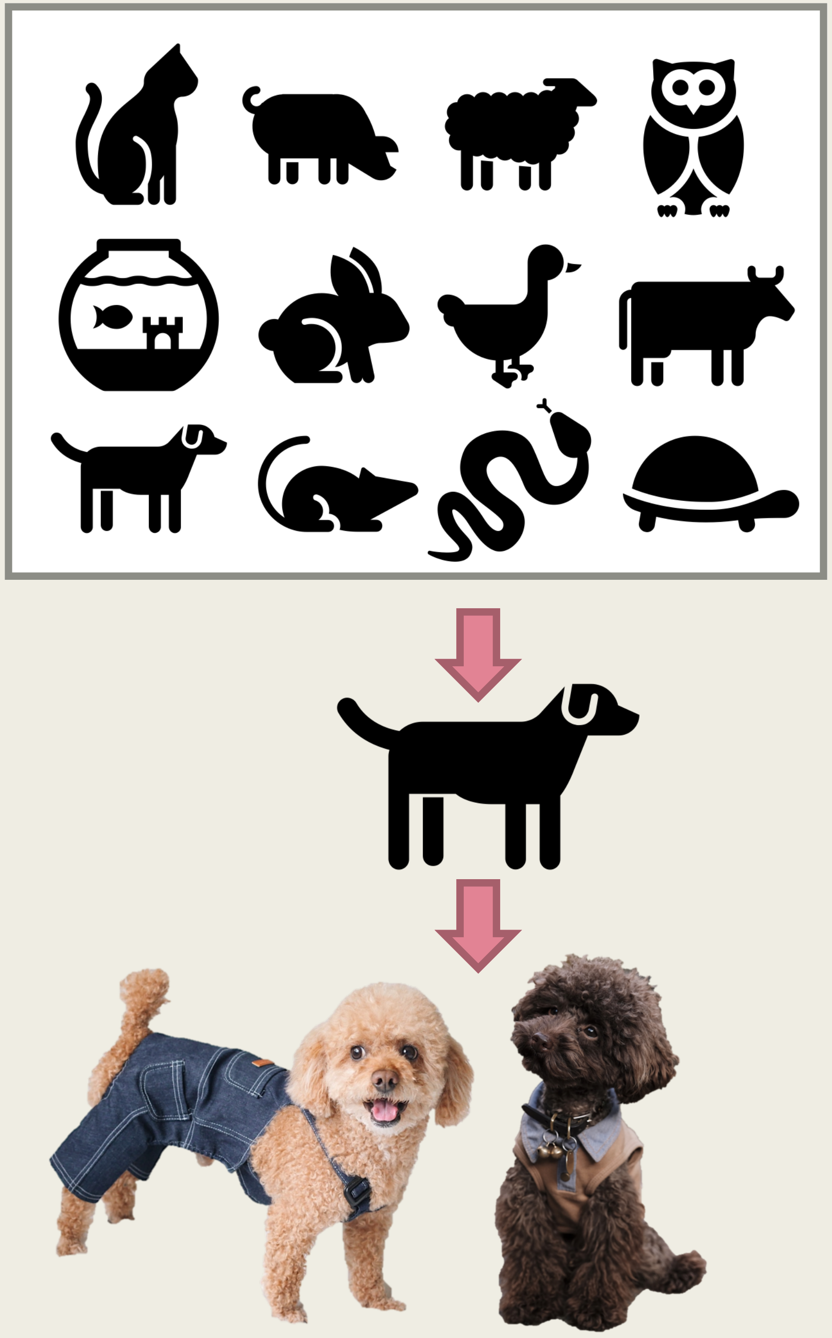
$$\mathrm{PMI}(x, y) = \log \frac{\mathrm{p}(x, y)}{\mathrm{p}(x)\mathrm{p}(y)}$$

Syntactic Features

- POS tags
- Dependency parse tags
- Word order
- Word distance (positional)
- Capitalization
- Punctuation
- Character repetition

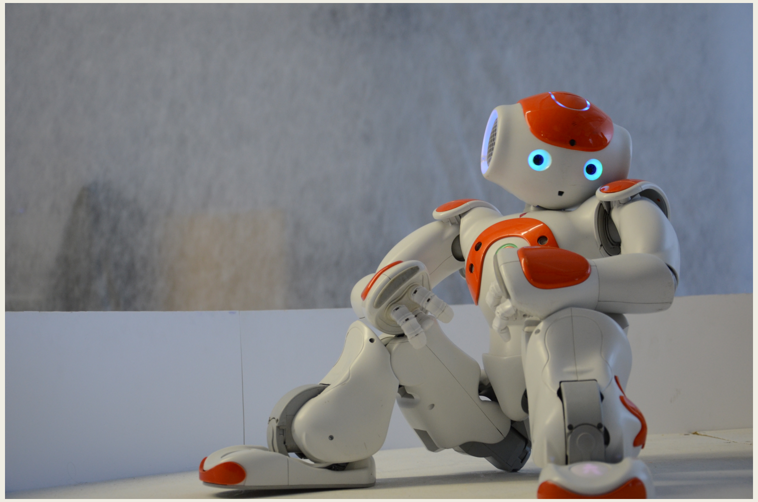
Lexical Features

- Information from machine-readable dictionaries
  - *WordNet: http://wordnetweb.princeton.edu/perl/webwn*
- Word distance (path from one word to another in a dictionary)
- Hypernym
  - *More general category (dog → animal)*
- Hyponym
  - *More specific category (dog → poodle)*



The list of features you can use to solve NLP problems is endless!

- Advantages of implicitly-learned features:
  - *No need to handcraft anything*
  - *Can identify patterns that may not be obvious to humans*
- Advantages of engineered features:
  - *Provides clearer insight into why an approach works/doesn't work*
  - *Can directly encode ideas from other research fields (e.g., social science)*
- Most researchers try out a wide variety of features on a held-out validation set while developing their models
- Many researchers end up combining implicitly-learned and engineered features

NLP APPLICATIONS

Dialogue Systems/Chatbots

- Two types:
  - *Conversational*
  - *Task-based*
- Increasingly pervasive!
  - *Siri, Alexa, Google Assistant….*
- Typically include components capable of completing the following tasks:
  - *Natural language understanding*
  - *Dialogue management*
  - *Natural language generation*
- Spoken dialogue systems also need to perform automated speech recognition (ASR) and text-to-speech synthesis

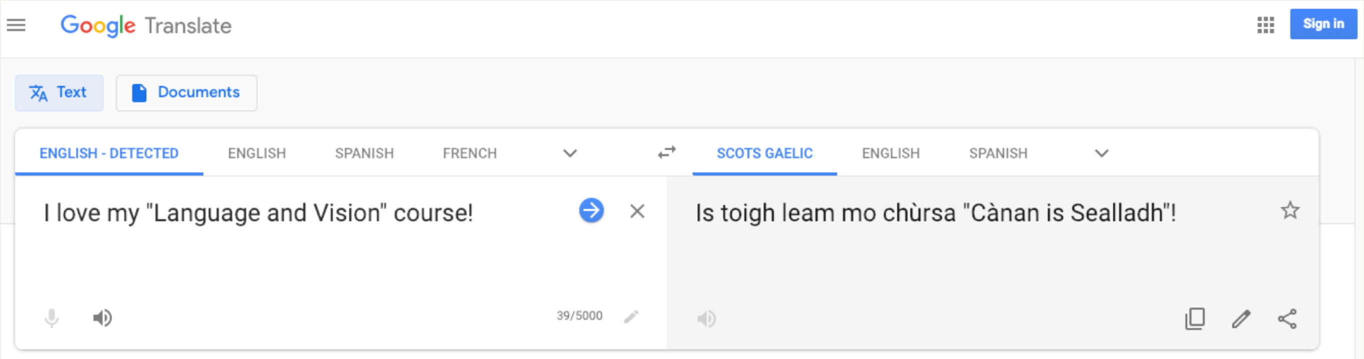
- Dialogue system frameworks:
  - *Dialogflow*
    - https://dialogflow.com/
  - *Wit.ai*
    - https://wit.ai/
  - *Microsot Bot Framework*
    - https://dev.botframework.com/
  - *IBM Watson*
    - https://www.ibm.com/watson/
  - *ChatScript*
    - https://github.com/ChatScript/ChatScript

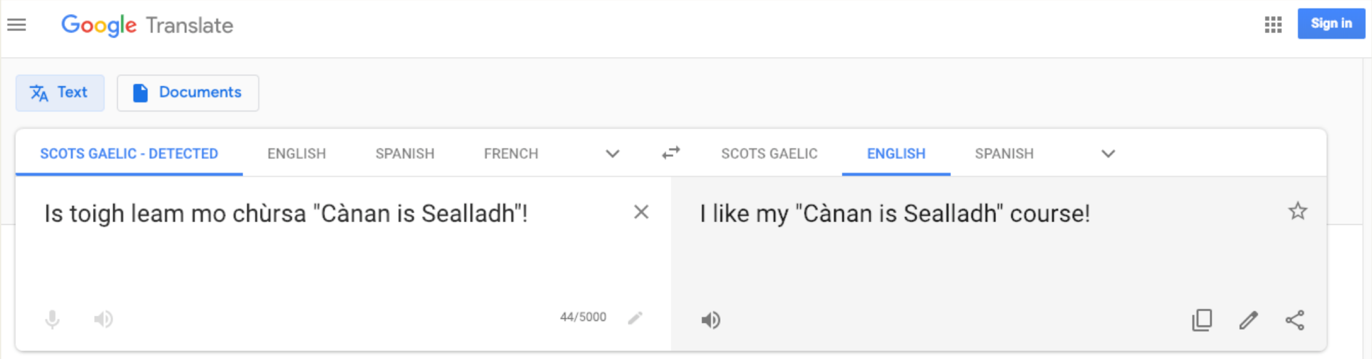
Cognitive Modeling and Psycholinguistics

- Attempting to understand the human mind by simulating cognitive processes using computational models
- “How do people comprehend language?”
- Often incorporates neuroimaging techniques:
  - *Electroencephalogram (EEG)*
  - *Functional magnetic resonance imaging (fMRI)*
- For more background reading on these topics, search for resources on cognitive science:
  - *http://cognet.mit.edu/*
  - *https://www.amazon.com/Cognitive-Science-Introduction-Mind/dp/1107653355*

Machine Translation

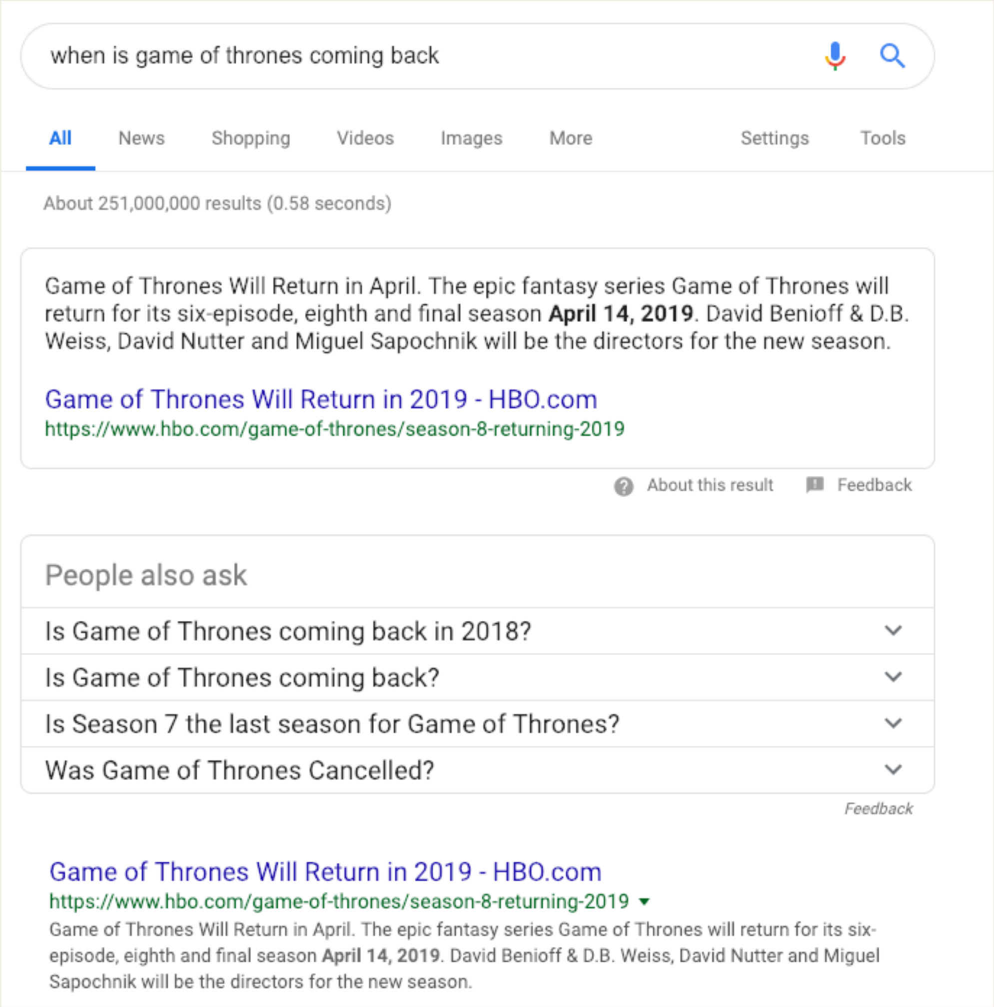
- Automatically translating text from one language to another
- Can be rule-based or statistical
- Statistical machine translation models require large corpora of aligned phrases from two languages
- They learn to predict scores for possible translations using the probabilities of different text alignments





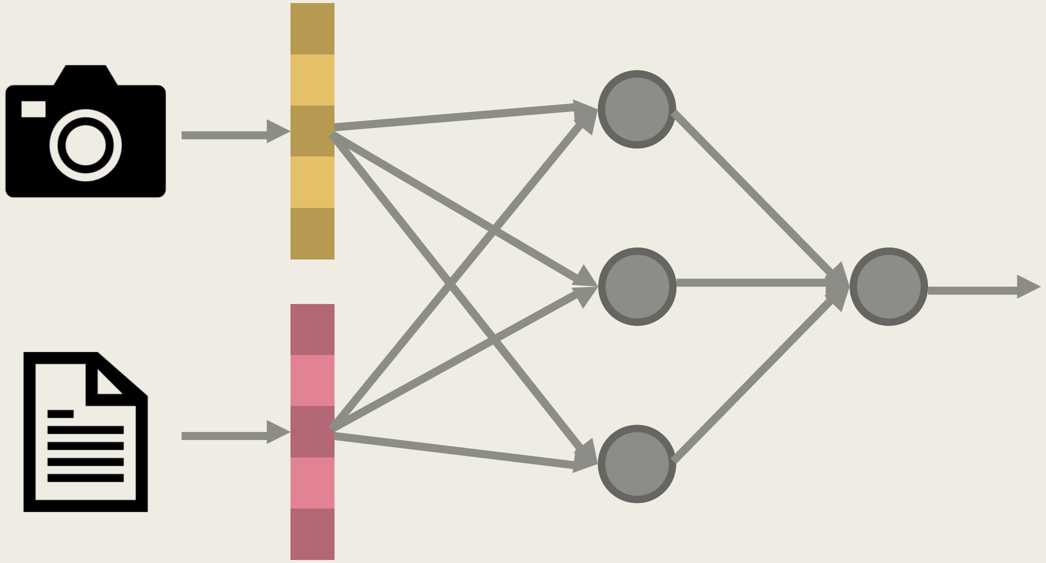
Question Answering

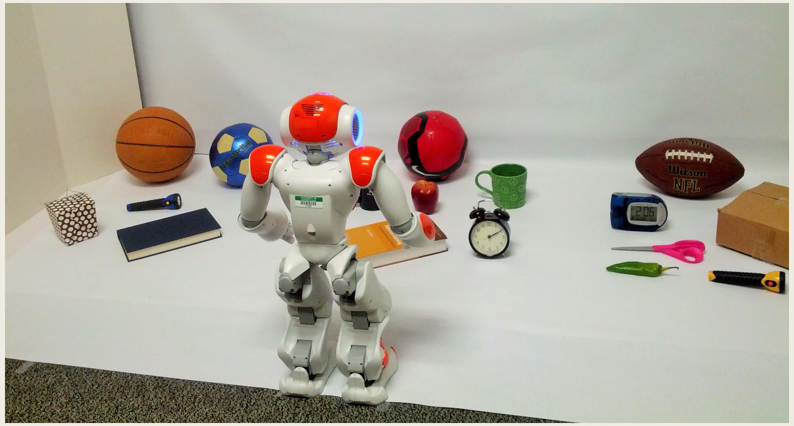
- Automatically interpreting the user's question and retrieving the correct information to provide in response
- In general, QA problems can be broken down such that there are three things associated with a question-answer pair:
  - *Context*
  - *Question*
  - *Text*
- Most QA models today work by matching a context (such as an article) with the question, and then identifying the start and end points of the actual answer within that context



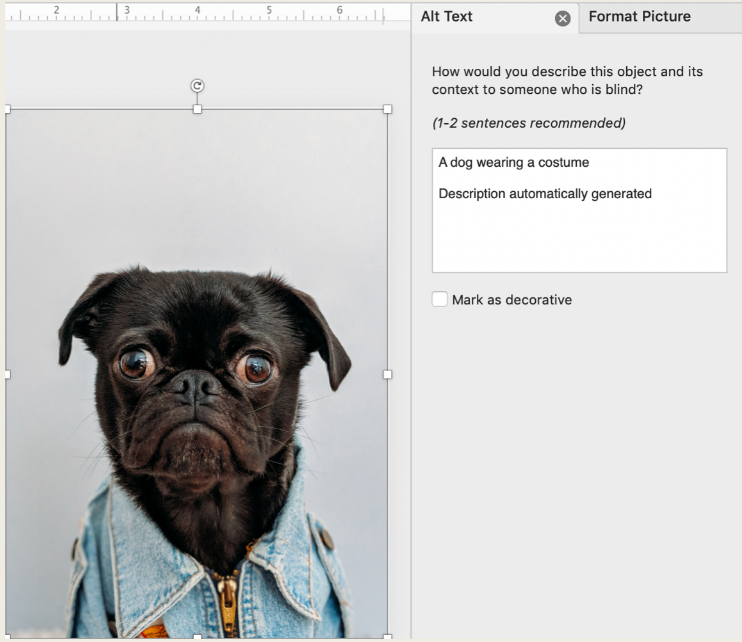
Multimodal NLP

- Learning word representations using multiple modalities
  - *Images, acoustic signals, haptic feedback, etc.*
- Aligning text with non-linguistic data
- Very useful in robotics applications and assistive technologies!





Boarding my ✈️ to go to 🇫🇷!!! 😁😍



Wrapping up….

- Core NLP terminology
  - *N-grams, corpus, text normalization, POS tagging, dependency parsing, stemming, lemmatization*
- Text similarity
  - *Edit distance, cosine similarity*
- Word embeddings
  - *Word2Vec, GloVe, ELMo*
- NLP features
  - *Implicitly learned, engineered*
- NLP applications
  - *Dialogue systems, cognitive modeling, machine translation, question answering, and multimodal NLP*
- For much more information about NLP methods and applications, a good starting point:
  - *https://web.stanford.edu/~jurafsky/slp3/*