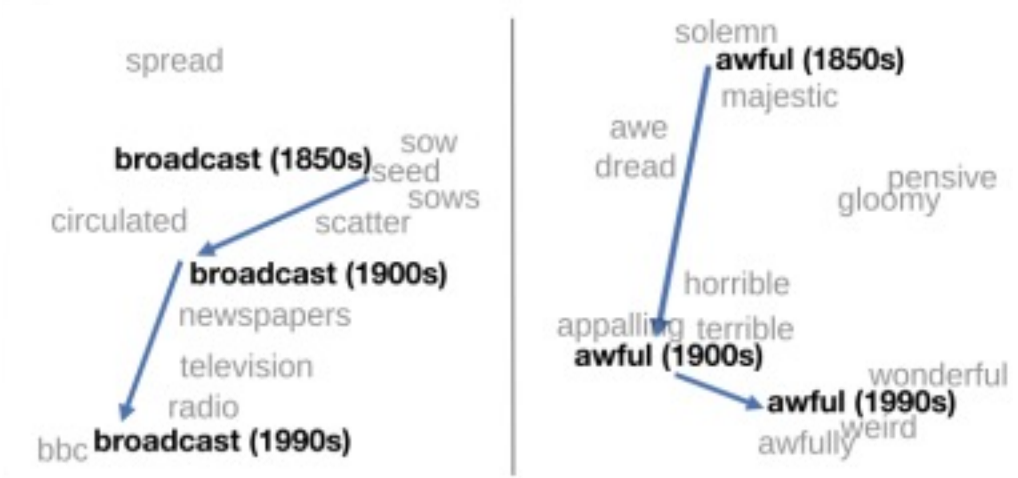
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Word Embeddings

1/31/23

Motivation

- Core question in understanding cultural and language evolution: how do words change meaning over time?



How can we represent meaning of a word?

Hamilton, William L., Jure Leskovec, and Dan Jurafsky. "Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016.

Motivation

- Can we use language analysis to identify and measure stereotypes?

- Example from last week:
  - Using PMI scores, Wikipedia articles about women tend to talk personal life more
  - Might we expect words like “family”, and “marriage” to be women-associated?

How can we measure “associations” between words?

Wagner, Claudia, et al. "It's a man's Wikipedia? Assessing gender inequality in an online encyclopedia." *Proceedings of the international AAAI conference on web and social media*. Vol. 9. No. 1. 2015.

How might we represent words?

"Lexical Semantics"

- Dictionary definition
- Lemma and word forms
- Senses

How might we represent words?

"Lexical Semantics"

- Dictionary definition
- Lemma and word forms
- Senses

pepper [**pep**-er] SHOW IPA

See synonyms for: pepper / peppered / peppering on Thesaurus.com

noun

1. a pungent condiment obtained from various plants of the genus *Piper,* especially from the dried berries, used whole or ground, of the tropical climbing shrub *P. nigrum*.
2. any plant of the genus *Piper.*: Compare pepper family.

A sense or "concept" is the meaning component of a word.

How might we represent words?

"Lexical Semantics"

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness

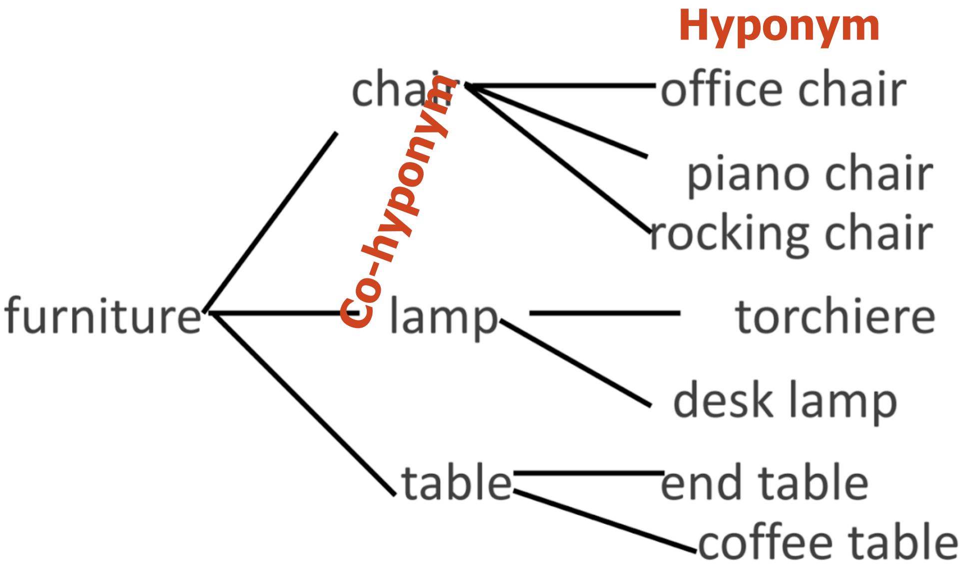
Relations between words

- **Synonyms** have the same meanings in some or all contexts
  - Couch / sofa, car / automobile
  - [Note that there are no examples of perfect synonymy]

- **Antonyms** senses that are opposite with respect to one feature of meaning
  - Dark / light, short / long, slow / fast
  - [Otherwise they are very similar]
  - [Antonyms can define a binary opposition or be at opposite ends of a scale]

Relations between words

- **Hypernym** / **Hyponym** (superordinate / subordinate)
  - One sense is a hyponym of another if the first sense is more specific, denoting a subclass of the other



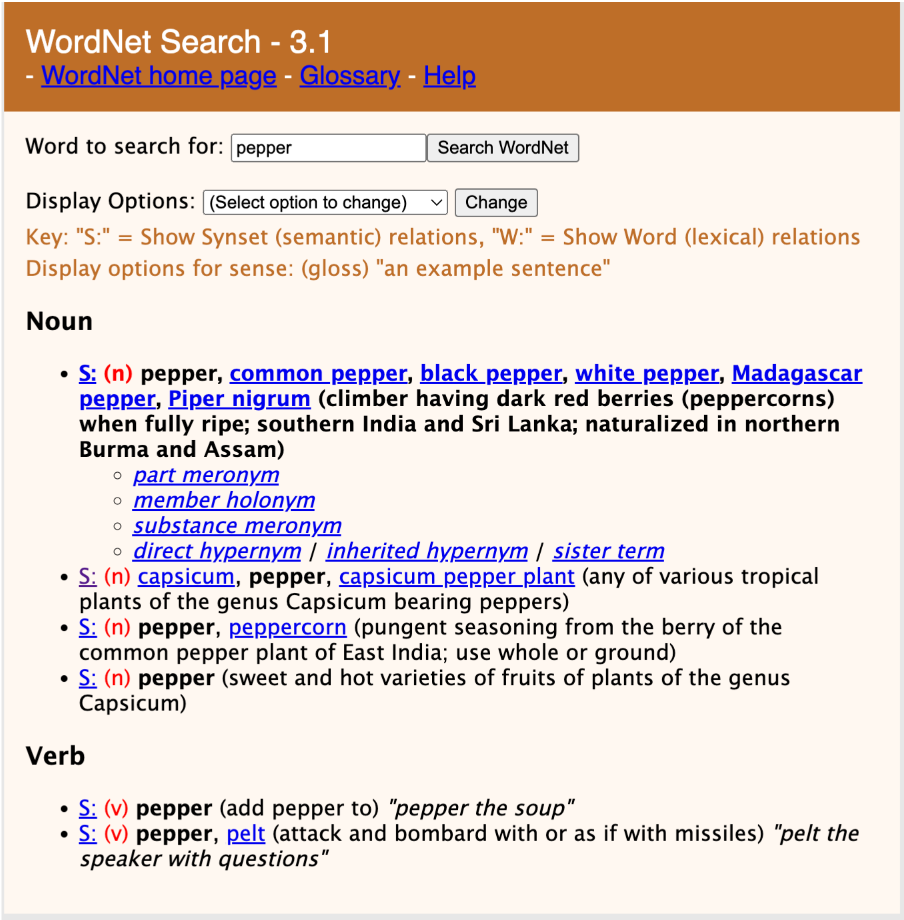
How might we represent words?

"Lexical Semantics"

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness

Annotated Resources for Lexical Semantics

- https://wordnet.princeton.edu/
- (python packages)



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: pepper [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **pepper, common pepper, black pepper, white pepper, Madagascar pepper, Piper nigrum (climber having dark red berries (peppercorns) when fully ripe; southern India and Sri Lanka; naturalized in northern Burma and Assam)**
  - *part meronym*
  - *member holonym*
  - *substance meronym*
  - *direct hypernym* / *inherited hypernym* / *sister term*
- S: (n) capsicum, **pepper**, capsicum pepper plant (any of various tropical plants of the genus Capsicum bearing peppers)
- S: (n) **pepper**, peppercorn (pungent seasoning from the berry of the common pepper plant of East India; use whole or ground)
- S: (n) **pepper** (sweet and hot varieties of fruits of plants of the genus Capsicum)

Verb

- S: (v) **pepper** (add pepper to) *"pepper the soup"*
- S: (v) **pepper**, pelt (attack and bombard with or as if with missiles) *"pelt the speaker with questions"*

How might we represent words?

"Lexical Semantics"

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles
- Connotation and sentiment

How to represent a word

- Until the ~2010s, in NLP words == atomic symbols
- **One-hot** representations in vector space:

w =

the	drinks	were	strong	but	the	tacos	were	bland	
1	0	0	0	0	1	0	0	0	
0	0	0	0	1	0	0	0	0	
0	1	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	0	V
0	0	0	0	0	0	0	0	1	
0	0	1	0	0	0	0	1	0	
…	…	…	…	…	…	…	…	…	
0	0	0	0	0	0	1	0	0	

0	0	...	0	0	0	0	1	tacos
0	0	...	0	1	0	0	0	burritos
0	0	...	0	0	1	0	0	drinks
0	0	...	0	0	0	0	1	leader
0	1	...	0	0	0	0	0	president
0	0	...	1	0	0	0	0	bland

How to represent a word

- Until the ~2010s, in NLP words == atomic symbols
- **One-hot** representations in vector space:

1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	0
…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	1	0	0
the	drinks	were	strong	but	the	tacos	were	bland

V

$\boldsymbol{w}$ =

- Useful for coding *identity*
- Can do matrix operations:
  - Feed into machine learning models
  - Matrix decompositions

How to represent a word

- Until the ~2010s, in NLP words == atomic symbols
- **One-hot** representations in vector space:

1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	0
…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	1	0	0
the	drinks	were	strong	but	the	tacos	were	bland

V

w =

- Sparse representations that scale with vocabulary size
- "tacos" is orthogonal to "burritos"
- How can we encode word *similarity* (not just identity)

Distributional hypothesis

- Encode word similarity (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: *tezgüino*
  - A bottle of *tezgüino* is on the table
  - Everybody likes *tezgüino*
  - Don't have *tezgüino* before you drive
  - We make *tezgüino* out of corn

context

term	1	2	3	4
tezgüino	1	1	1	1
loud	0	0	0	0
motor oil	1	0	0	1
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

Word-word co-occurrence matrix

Apples are green and red.
Red apples are sweet.
Green oranges are sour

-	apples	are	green	and	red	sweet	oranges	sour
apples	2	2	1	1	2	1	0	0
are	2	3	1	1	2	1	1	1
green	1	1	2	1	1	0	1	1
and	1	1	1	1	1	0	0	0
red	2	2	1	1	2	1	0	0
sweet	1	1	0	0	1	1	0	0
oranges	0	1	1	0	0	0	1	1
sour	0	1	1	0	0	0	1	1

https://www.baeldung.com/cs/co-occurrence-matrices

Distributional hypothesis

- These representations encode **distributional** properties of each word.
- The **distributional hypothesis**: words with similar meaning are used in similar contexts.

"The meaning of a word is its use in the language." [Wittgenstein 1943]

"If A and B have almost identical environments we say that they are synonyms." [Harris 1954]

"You shall know a word by the company it keeps." [Firth 1957]

How to encode context

Really really big

context

term	1	2	3	4	…
tezgüino	1	1	1	1	
loud	0	0	0	0	…
motor oil	1	0	0	1	
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	

sparse

How to encode context

- **TF-IDF**
- **Word2Vec**
- Not covering other methods: e.g. Brown clusters, Matrix factorization

Encoding Context with TF-IDF

- Consider a matrix of word counts across documents: **term-document matrix**

Words like *the*, *it*, *they* are not very discriminative, we can do better than raw counts

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

word vector

Bag-of-words document representation

Encoding Context with TF-IDF

- TF-IDF incorporates two terms that capture these conflicting constraints:
  - **Term frequency (tf):** frequency of the word t in the document

$$tf_{t,d} = \log(count(t,d) + 1)$$

Encoding Context with TF-IDF

- TF-IDF incorporates two terms that capture these conflicting constraints:
  - **Term frequency (tf):** frequency of the word t in the document

$$tf_{t,d} = \log(count(t,d) + 1)$$

  - Document frequency (df): number of documents that a term occurs in
  - **Inverse document frequency (idf):**

$$idf_t = \log(\frac{N}{df_t})$$

Higher for terms that occur in fewer documents

  - (N) is the number of documents in the corpus

Encoding Context with TF-IDF

- TF-IDF incorporates two terms that capture these conflicting constraints:
  - **Term frequency (tf):** frequency of the word t in the document

$$tf_{t,d} = \log(count(t,d) + 1)$$

  - Document frequency (df): number of documents that a term occurs in
  - **Inverse document frequency (idf):**

$$idf_t = \log(\frac{N}{df_t})$$

Higher for terms that occur in fewer documents

  - (N) is the number of documents in the corpus

- **TF-IDF** combines these two terms: $tf{-}idf_{t,d} = tf_{t,d} * idf_t$

Notes about TF-IDF

- Very useful way of creating *document embeddings*
  - Designed for and still excels at **document retrieval**
  - Often useful as features for classification models

- We could use variants of *log-odds with a Dirichlet prior ratios* or *topic models* to create document or word embeddings

- Word-embedding use cases of TF-IDF are not as common

Dimensionality Reduction

- TF-IDF representations are still sparse
  - Wikipedia: ~29 million English documents. Vocab: ~1 million words.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.
  - May better capture synonymy
  - In practice, they just work better [Baroni et al. 2014]
- How do we build dense vectors?

Word2Vec

- Instead of counting how often each word w occurs near “corn”, train a classifier on a binary prediction task: Is w likely to show up near “corn”?
- Don’t actually care about performing this task, but we’ll take the learned classifier weights as the word embeddings
- Training is self-supervised: no annotated data required, just raw text!

Word2Vec: Two Algorithms

- Context bag-of-words (CBOW): predict current word using context
  - $P(w_t \mid w_{t+1}, \dots, w_{t+k}, w_{t-1}, \dots, w_{t+k})$
- Skip-gram: predict each context word using current word
  - $P(w_{t+1}, \dots, w_{t+k}, w_{t-1}, \dots, w_{t+k} \mid w_t)$

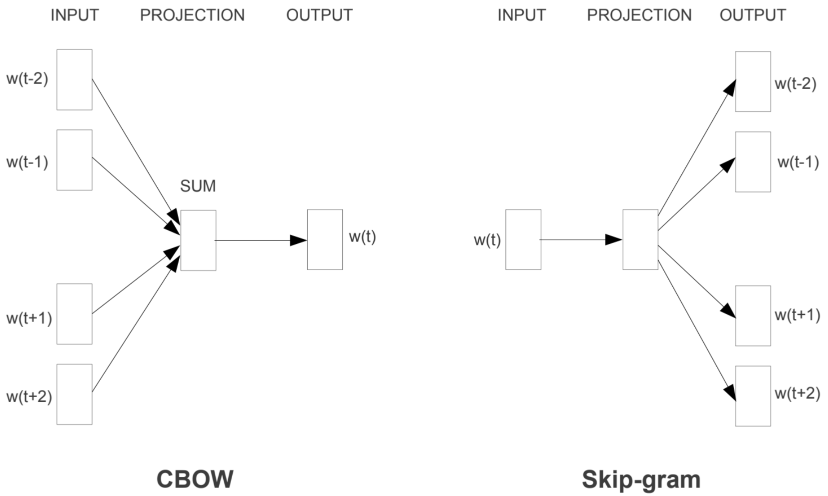


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Skip-gram: Probabilities

… that Europe needs unified **banking** regulation to replace the hodgepodge …

w_{t-3} w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2} … w_{t+5}

We want to train a model to output $P\left(w_{t+j} \middle| w_t\right)$. We define:

$$P\left(w_{t+j} \middle| w_t\right) = P(o \mid c) = \frac{\boxed{\exp}\left(\boxed{u_o^T v_c}\right)}{\boxed{\sum_{i=1}^{V} \exp\left(u_i^T v_c\right)}}$$

Dot product (similarity metric)
Larger dot product = larger similarity

softmax function

o = index of outside (context) word
c = index of center word (w_t)
V = vocab size

u = vector for word as outside (context)
v = vector for word as center

Skip-gram: How do we learn u and w?

… that Europe needs unified **banking** regulation to replace the hodgepodge …

w_{t-3} w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2} … w_{t+5}

m = 5

Data Likelihood: probability of any context word given center word (maximize)

[Note we're assuming conditional independent]

$$L = \frac{1}{T}\prod_{t=1}^{T}\prod_{-m\le j\le m, j\ne 0} P(w_{t+j}|w_t,\theta)$$

Objective Function: negative log probability (minimize)

$$L = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\ne 0} \log P(w_{t+j}|w_t,\theta)$$

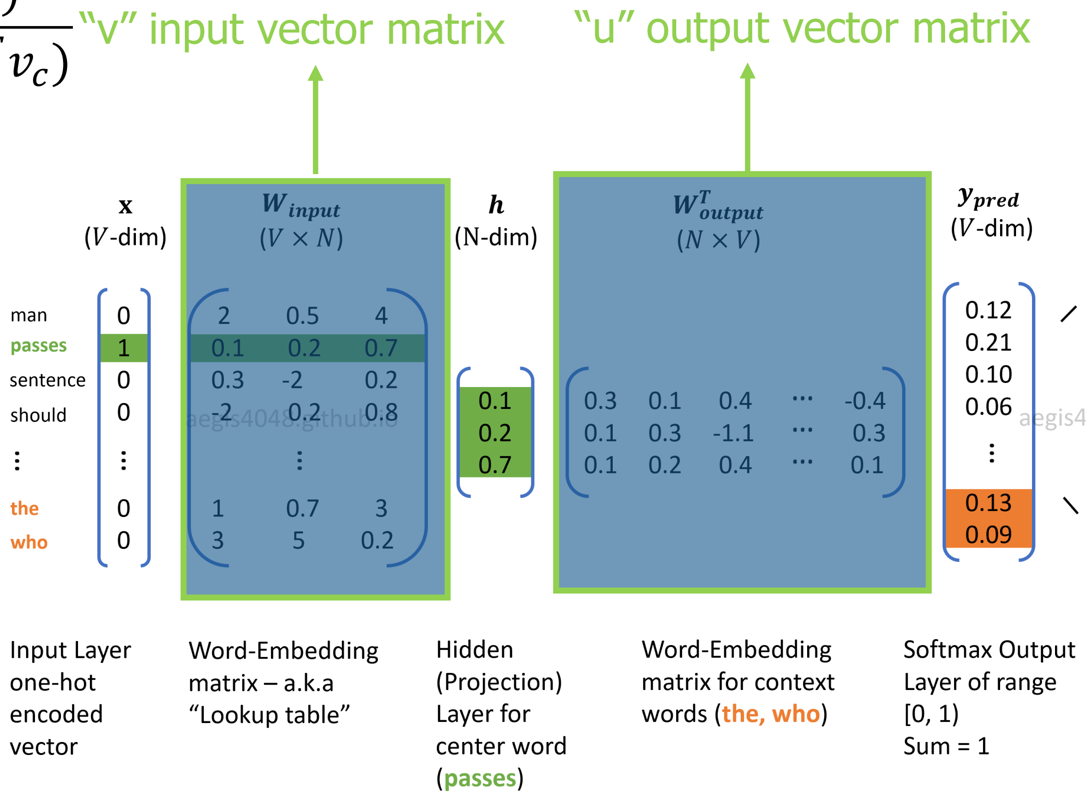
Skip-gram: How do we learn u and w?

$$L = -\frac{1}{T}\sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log P(w_{t+j} | w_t, \theta)$$

$$P\left(w_{t+j} \middle| w_t\right) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

- Gradient-based estimation (e.g. stochastic gradient descent)
  - Start with uninformed guess for u and w (e.g. random)
  - Iteratively change u and w in the way that locally best-improves the guess
  - Computing gradients (e.g. derivatives) of the objective function with respect to u and w inform how to change them

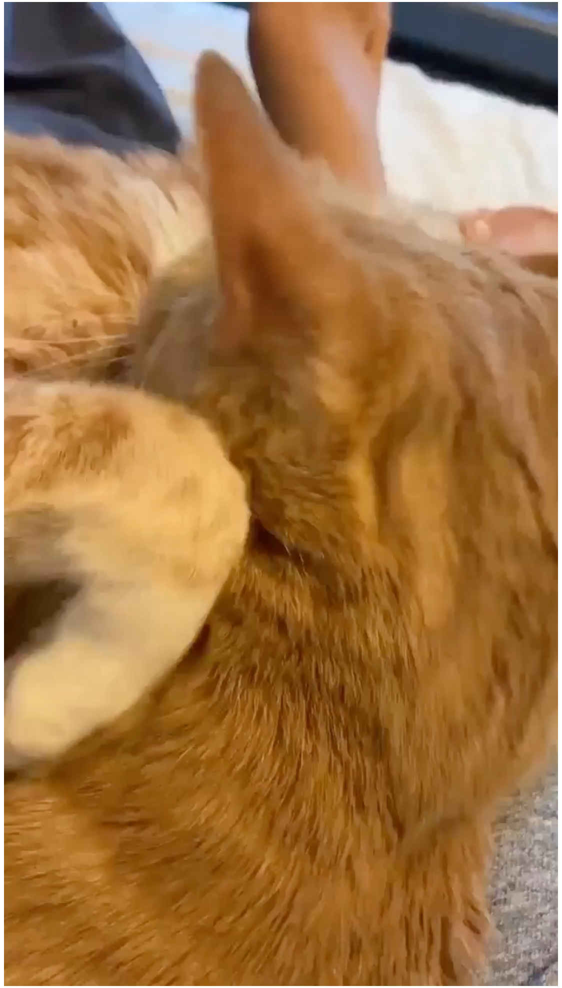
$$\frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$



At the end of training we've learned 2 sets of embeddings: we can average them or just keep one of them

https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling

Short Break



Skip-gram

$$\frac{\exp(u_o^T v_c)}{\boxed{\sum_{i=1}^{V} \exp(u_i^T v_c)}}$$

- Problem:
  - Denominator is computationally expensive! O(VK)
  - Solutions:
    - Hierarchical softmax O(log V)
    - Negative Sampling O(1)

Skip-gram: Negative sampling

$$\frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

Encourage center word and context word to have similar vectors

Encourage center word and all other words to have different vectors

- Intuition: we don't need to down-weight all other words at once, we can chose a small number of negative samples

Skip-gram: Negative sampling

P(o | c) = $\frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$ $\longrightarrow$ $\frac{1}{1 + \exp(-u_o^T v_c)}$

- New objective (single context word, k negative samples)

$$\log P(o_+|c) + \sum_{i=1}^{k} \log(1 - P(o_i|c))$$

- (Problem changes from multiclass to binary)

Choosing negative samples

- Generally choose frequent words
- Could choose purely based on frequency P(w)
- In practice, $P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$ with $\alpha = 0.75$ works well (gives rare words slightly higher probability)

Recap

- We want meaningful representations of words that we can use for corpus analytics (and other things)

- By defining a fake task, predicting context from a word (skip-gram) or a word from context (CBOW), we can learn vector matrices

- Actual implementation requires additional tricks for reducing computational complexity

Pre-trained Word2Vec Embeddings

- https://code.google.com/archive/p/word2vec/
- You can train embeddings on your own data
- Depending on your application, you can also start with embeddings trained on large data set

Other word embeddings: GloVe [Pennington et al. 2014]

- https://nlp.stanford.edu/projects/glove/
- "Global Vectors"
- Model is based on capturing global corpus statistics
- Incorporates ratios of probabilities from the word-word cooccurrence matrix (intuitions of count-based models) with linear structures used by methods like word2vec

Other word embeddings: fasttext [Bojanowsi et al. 2017]

- Word2vec can’t handle unknown words and sparsity of rare word-forms (e.g. we should be able to infer ”milking” from “milk” + “ing”)
- Uses subword models, representing each word as itself plus a bag of constituent n-grams, with special boundary symbols < and > added to each word.
- Each word is represented by the sum of all of the embeddings of its constituent n-grams. Unknown words can be represented by just the sum of the constituent n-grams.

Gensim: Python Package for working with word embeddings

```
>>> from gensim.test.utils import common_texts
>>> from gensim.models import Word2Vec
>>>
>>> model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, workers=4)
>>> model.save("word2vec.model")
```

https://radimrehurek.com/gensim/models/word2vec.html

Takeaways

- Intuitive ideas behind representing words as vectors
- Distributional Hypothesis
- Basic ideas behind TF-IDF weighting
- Basic ideas behind Word2Vec
  - Difference between CBOW and Skip-gram
  - Practical challenges
- *Know where your embeddings came from and how they were made*

Next Class

- How do we know if our embeddings work?
- What do we do with them?

HW 1

- Released today
- Individual assignment (ok to discuss at a high-level, but all code and written responses must be your own)
- 3 Parts:
  - 1. Log-odds
  - 2. Topic Modeling
  - 3. Word Embeddings (we will have finished this material by Monday)
- Policy reminder:
  - 5 late days, no other extensions
  - Can use late days in any way (except not for final project report)

HW 5 →"Project"

- We're going to refer to HW 5 as course project for scheduling reasons (more details on the assignment later)

Acknowledgements and Resources

- Slide content drew heavily from Emma Strubell and Yulia Tsvetkov's slides: http://demo.clab.cs.cmu.edu/11711fa20/slides/11711-04-word-vectors.pdf

- Resources:
  - Lecture Notes from Stanford NLP class on word embeddings https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf
  - Efficient Estimation of Word Representations in Vector Space (original word2vec paper) https://arxiv.org/pdf/1301.3781.pdf
  - Distributed Representations of Words and Phrases and their Compositionality (negative sampling paper) https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf
  - Jurafsky and Martin textbook Chap 6: https://web.stanford.edu/~jurafsky/slp3/6.pdf