Natural Language Processing

Paolo Favaro

UNIVERSITÄT BERN

Contents

- Natural Language Processing
  - Word embeddings (word2vec, n-grams, skip-grams, CBoW, GloVe), language models and generation (ELMO, BERT, GPT)
- Based on Stanford University's course on NLP by Christopher Manning and material from the deep learning course of Pieter Abbeel at the University of Berkeley

What is Language?

- **Nouns** — to describe things in the words
- **Verbs** — to describe actions
- **Adjectives** — to describe properties

*From Philipp Koehn, AI: NLP course at Johns Hopkins University

Natural Language Processing

- In NLP the goal is to build a machine that can understand and generate text and speech data in natural languages
- It is an interdisciplinary field of linguistics, computer science and artificial intelligence
- Examples of NLP tasks are sentiment analysis/toxicity classification, machine translation, named entity recognition, grammatical error correction, speech recognition/generation, text completion and generation, information retrieval
- Can be applied to more general language processing, such as programming languages, biological/chemical sequences, etc

Applications: Digital Assistants

- Today there are many digital assistants on mobile phones and at home
  - Alexa (Amazon)
  - Siri (Apple)
  - Cortana (Microsoft)
  - Google Assistent (Google)

- They can understand speech, have a conversation and execute instructions

Applications: Question/Answering

https://huggingface.co/spaces/abidlabs/question-answering-demo

Applications: ChatBots

User Tell me about when Christopher Columbus came to the US in 2015

ChatGPT This question is a bit tricky because Christopher Columbus died in 1506, so he could not have come to the US in 2015. But let's pretend for a moment that he did!

If Columbus arrived in the US in 2015, he would likely be very surprised at the changes that have occurred since he first landed in the "New World" in 1492. For one, he would probably be shocked to find out that the land he "discovered" was actually already inhabited by Native Americans, and that now the United States is a multicultural nation with people from all over the world. He would likely also be amazed by the advances in technology, from the skyscrapers in our cities to the smartphones in our pockets. Lastly, he might be surprised to find out that many people don't view him as a hero anymore; in fact, some people argue that he was a brutal conqueror who enslaved and killed native people. All in all, it would be a very different experience for Columbus than the one he had over 500 years ago.

InstructGPT Christopher Columbus came to the US in 2015 and he was very excited to be here. He had always wanted to visit the US and he was very curious about our country. He was very impressed with our country and he enjoyed his time here.

https://openai.com/blog/chatgpt

How to Model Language*?

- **Rationalism**: A significant part of the knowledge in the human mind is fixed in advance (and not learned through the senses) → create intelligent systems by finding out how the human brain works and by hardcoding its rules

- **Empiricism**: Only very basic skills are hardcoded in the brain (eg, association, pattern recognition) and these are then applied to sensory input to learn the details of natural language → create intelligent systems by fitting general probabilistic models to data

*C. Manning and H. Schutze, Foundations of Statistical Natural Language Processing

Probabilistic Language Models

- Probabilistic language models describe the relationships between constituent parts of language

- Language parts (eg letter, words, sentences, paragraphs) need to be mapped to numbers (vectors) so that we can perform computations with them (this step is called **text preprocessing**)

- One important text preprocessing step is **tokenization**: Breaking text into words, sentences, symbols or some intermediate elements called tokens

- Preprocessing is about simplifying text without changing its meaning

Word Representations

- Word representations should relate the written symbols to their meaning
- What is “meaning”? The idea expressed by a word, phrase etc
- One first mapping is **WordNet**
  - It categorizes and relates nouns, verbs, adjectives and adverbs
  - It has basic relationships such as synonyms (same or similar meanings), hypernyms (broader categories), hyponyms (more specific categories), meronyms (whole-part relationship), antonyms (opposite meaning)

WordNet

- Good general reference
- Does not specify nuances of meanings (synonymity may depend on context)
- Difficult to maintain and update
- Cannot be used to compute word similarity

Representing Words

- To perform calculations, words x_i must be mapped to vectors $\phi(x_i) \in \mathbb{R}^m$, where m is the vocabulary size

- One choice for such mapping is the so-called **one-hot encoding**

$$\phi_j(x_i) = \begin{cases} 0 & \text{if } j \neq i \\ 1 & \text{if } j = i \end{cases}$$

Representing Words

- Issues with one-hot encodings
  - There is no meaningful similarity measure; eg, “hotel” and “motel” are at the same distance as to any other word
  - The vector size is quickly very large (grows linearly with the vocabulary size)
  - Lots of zeros in the vectors (ie, lots of useless computations)
- Instead of one-hot encodings **embed the similarity directly in the vectors**

Building Embeddings

- **Embeddings** in mathematics are structure-preserving mappings
- Eg, to embed a graph in vector space: assign vectors to nodes such that their relative distance corresponds to the connectivity within the graph
- Embeddings can be defined either through

1. A hard-coded rule
2. End-to-end training (ie, train them with the model on the “downstream” task)
3. Pre-training (ie, train them in advance on a “general-purpose” task)

Bag of Words

- BoW defines embeddings via a hard-coded rule
- It captures the set of words in a text (order is discarded) and their multiplicity
- Eg*, the text "John likes to watch movies. Mary likes movies too." is encoded as BoW = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1}

*https://en.wikipedia.org/wiki/Bag-of-words_model

Bag of Words

- In document retrieval: The term frequency-inverse document frequency (tf-idf)

$$v = [t_1, \ldots, t_V]^\top$$

where $t_i = \dfrac{n_i}{n} \log \dfrac{N}{N_i}$

n_i is the number of occurrences of the i-th word
n is the number of word occurrences
N_i is the number of documents with the i-th word and
N is the number of documents in the dataset

n-Grams

- These models assign probabilities to words (or groups of words) by looking at the frequency of words
- They extend the bag of words (BoW) representation to n-dimensional sequences of words (BoW is a 1-gram model)
- Eg, a bi-gram of the sentence “Terry goes to school to study.” is the set {“Terry goes”, “goes to”, “to school”, “school to”, “to study”}

n-Grams

- The goal is to approximate the probability of a word given the previous history

$$p(w_k \mid w_{1:k-1})$$

- An exact calculation of this conditional probability or even the corresponding joint probability would require counting how many sentences with the given words appear in a dataset; this may give a poor estimate when k is large

- n-Grams introduce the following (Markov) approximation

$$p(w_k \mid w_{1:k-1}) \simeq p(w_k \mid w_{k-n+1:k-1}) \qquad \text{where } n < k$$

Word2Vec

- Word2Vec[$] represents words through their context[*] (also related to polysemy)
- The **context** of a word is the set of words that appear within a fixed-size window around the word
- Example[#] of context to the word "banking"

...government debt problems turning into	***banking***	*crises as happened in 2009...*
...saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge...*
...India has just given its	***banking***	*system a shot in the arm...*

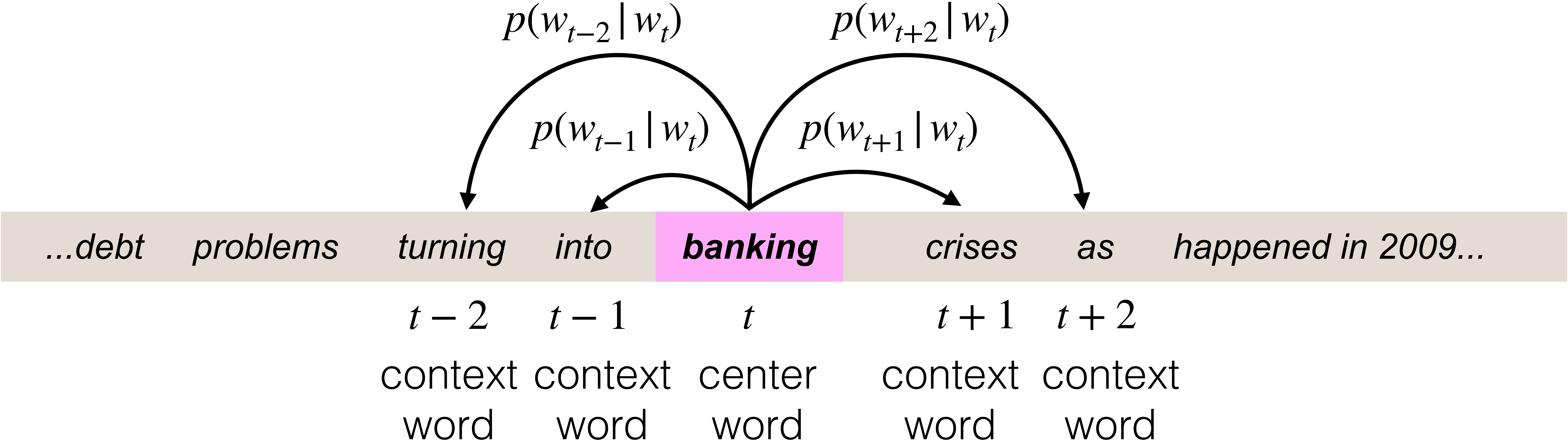
[$]Mikolov et al, Efficient Estimation of Word Representations in Vector Space, ArXiv 2013

[*]"You shall know a word by the company it keeps" (Firth 1957: 11)

[#]CS224n Course on NLP

Skip-Gram: Context Prediction

- Compute the probability $p(w_{t+j} | w_t)$ of context words w_{t+j} given the center word w_t ("banking")



Word Mapping

- Words are mapped to vectors that are **similar** to the vectors of words in the context

$$\text{banking} = \begin{bmatrix} 0.432 \\ -0.332 \\ 0.245 \\ 0.897 \\ -0.112 \\ 0.312 \end{bmatrix} \qquad \text{debt} = \begin{bmatrix} 0.132 \\ -0.732 \\ 0.145 \\ 0.697 \\ -0.312 \\ 0.012 \end{bmatrix}$$

Word Mapping and Similarity

- **Similarity** between words w and v is measured via the vector dot product

- Each word has 2 representations: One as center w and one as context word u (but one can also use one word for both from the start; also at the end, one computes the average of the 2 words to have a single representation)

- We define the probability of a context word given the center word as

$$p(u_{t+j}|w_t) = \frac{e^{u_{t+j}^\top w_t}}{\sum_k e^{u_k^\top w_t}}$$

- To avoid the cost of normalization over the vocabulary, use hard negatives

Optimization

- We can formulate the optimization as a Maximum Likelihood problem

$$\min_{\theta} J(\theta) = \min_{\theta} -\frac{1}{|S|} \sum_{\substack{i=1,\ldots,|S| \\ t=S(i)}} \sum_{\substack{1 \le |j| \le m \\ c=S(i+j)}} \log p(u_c \mid w_t)$$

where θ is the collection of all context and center embeddings and S is the sequence of word indices in the corpus

Implementation

- Mikolov et al, 2013 implemented skip-gram by using

$$\min_\theta J(\theta) = \min_\theta \frac{1}{|S|} \sum_{\substack{i=1,\ldots,|S| \\ t=S(i) \\ |j| \in [1,\ldots,m] \\ c=S(i+j)}} \log \sigma\left(u_c^\top w_t\right) + \sum_{k \sim \mathcal{U}^{3/4}[1,\ldots,|S|]/Z} \log \sigma\left(-u_k^\top w_t\right)$$

n negative samples

3/4 makes less frequent words be sampled more often

where the sigmoid is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$

- They maximize the probability of words in context to co-occur and of out of context words not to co-occur

Continuous Bag of Words

- In the Continuous Bag of Words (CBoW) we instead predict the center word by using the context

- But rather than making a prediction from each separate context word, we average all the context words and then make a prediction given the average

$$\min_{\theta} J(\theta) = \min_{\theta} -\frac{1}{|S|} \sum_{\substack{i=1,\ldots,|S| \\ t=S(i)}} \log p\left(w_t \middle| \frac{1}{2m} \sum_{\substack{1 \leq |j| \leq m \\ c=S(i+j)}} u_c \right)$$

- Related to BoW because the average discards the original order of the context

Interesting Properties

- Learned representations have structured relationships
- For example, if we perform the following arithmetic operations on the embeddings corresponding to the listed words and look for the nearest neighbor, we find that

  Paris - France + {Italy, Japan, Florida} = {Rome, Tokyo, Tallahassee}
  Einstein - scientist + {Messi, Mozart, Picasso} = {midfielder, violinist, painter}

Direct Prediction Methods

- Skip-gram and CBoW are direct prediction methods
- **Advantages**
  - Good performance on downstream tasks, can capture complex patterns beyond word similarity
- **Disadvantages**
  - Scales with the corpus size, inefficient use of statistics

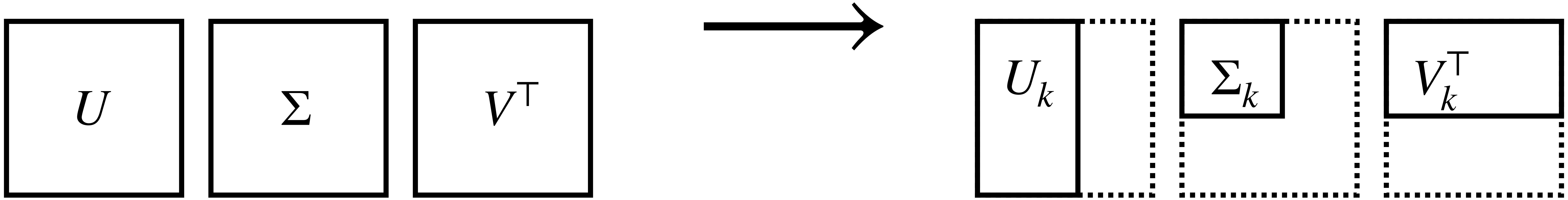
Co-Occurrence Matrix

- So far we looked at the co-occurrence one word pair at a time
- This is not very efficient
- Why not capture co-occurence counts directly?
- Let X be the (word pairs) co-occurrence matrix such that X_{ij} is the number of times word j co-occurs in the context of word i
- **Problem**: The vectors in X are very high-dimensional (lots of storage required) and sparse → can't be used directly for training

Dimensionality Reduction

- One solution is to project the co-occurrence vectors to lower-dimensional ones
- Can be achieved via singular value decomposition (SVD)

$$X = U\Sigma V^{\top} \qquad \hat{X} = U_k \Sigma_k V_k^{\top}$$



- Other improvements: max count of words, ignore very frequent words, weigh more words closer to the center word

Dimensionality Reduction

- **Advantages**
  - Relatively simple, makes efficient use of observed statistics

- **Disadvantages**
  - Computation scales quadratically, difficult to update with new words, captures only word similarities, is heavily influenced by large counts

GloVe*

- GloVe (Global Vectors) combines the advantages of the direct prediction and dimensionality reduction methods

- Let $X_i = \sum_k X_{ik}$ be the number of times words appear in the context of word i

- Then, we can define the probability that a word j appear in the context of a word i as

$$P_{ij} = p(u_j | w_i) = \frac{X_{ij}}{X_i}$$

- The vectors u_j and w_i are the (learned) vectors corresponding to the rows and columns of the co-occurrence matrix (they are different, this helps training)

*Pennington, Socher, and Manning, GloVe: Global Vectors for Word Representation, EMNLP 2014

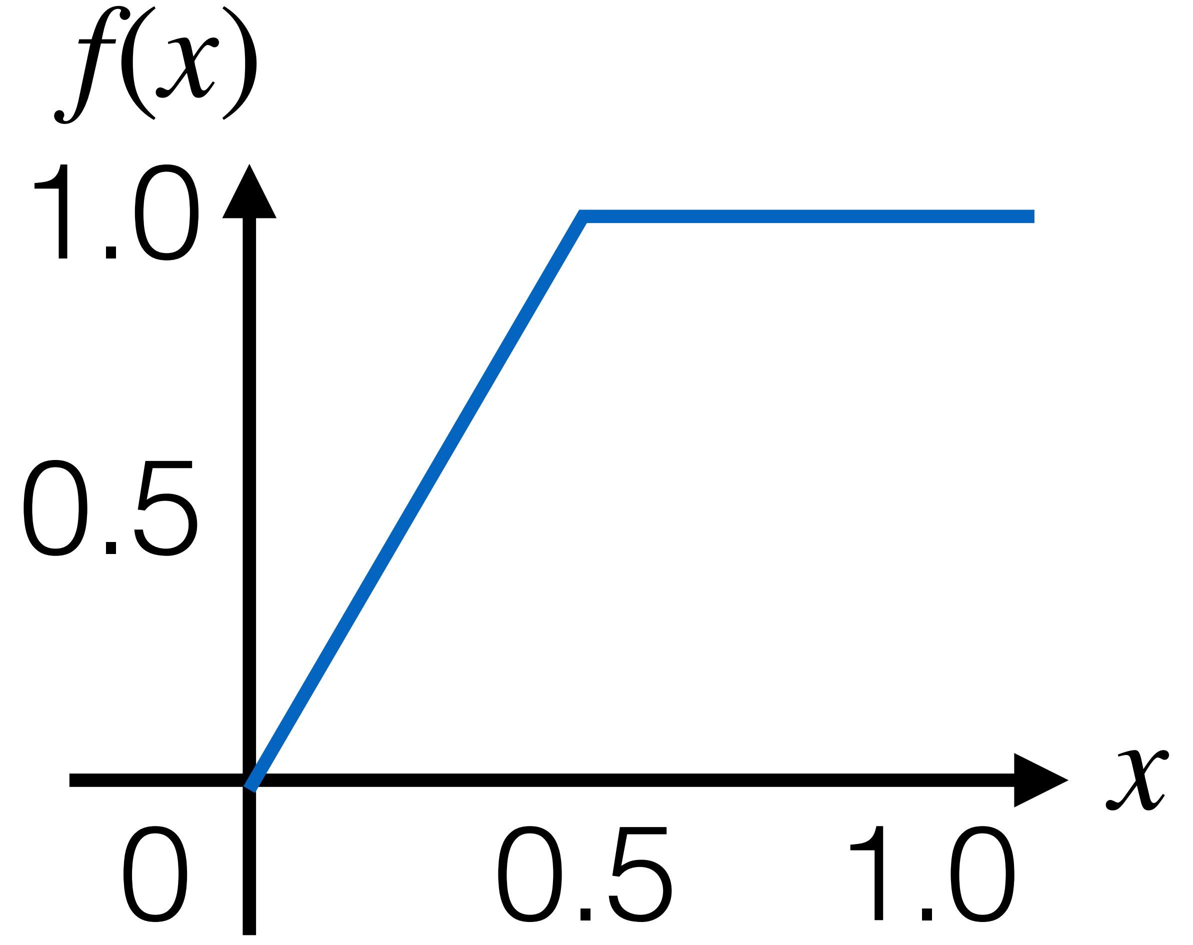
Training GloVe*

- Optimize

$$\min_{\theta} J(\theta) = \min_{\theta} \frac{1}{2} \sum_{i,j=1}^{T} f\left(P_{ij}\right)\left(u_i^{\top} w_j - \log P_{ij}\right)$$

- The aim is to make the inner product of word pairs similar to the log of the co-occurrence matrix at that pair

- The function f can apply a weight to the different pairs (as discussed in the dimensionality reduction)



*Pennington, Socher, and Manning, GloVe: Global Vectors for Word Representation, EMNLP 2014

Training GloVe*

- Optimize

$$\min_{\theta} J(\theta) = \min_{\theta} \frac{1}{2} \sum_{i,j=1}^{T} f\left(P_{ij}\right) \left(u_i^{\top} w_j - \log P_{ij}\right)$$

- **Advantages**
  - Fast training, can work on a huge corpus (needs only one pass to build X), performs well even with small corpora and vectors

*Pennington, Socher, and Manning, GloVe: Global Vectors for Word Representation, EMNLP 2014

Language Models

- Is the task of predicting a word given the preceding context

the students opened their _____

- books
- laptops
- exams
- minds

- Can be formulated as: Predict the probability of a word x^{t+1} given words from the past $x^1, \ldots, x^t$

$$p\left(x^{t+1} | x^1, \ldots, x^t\right)$$

Language Models

- Given the estimated conditional probabilities $p\left(x^{t+1} \mid x^1, \ldots, x^t\right)$, one can also estimate the probability of a text $x^1, \ldots, x^T$

$$p\left(x^1, \ldots, x^T\right) = \prod_{t=1}^{T} p\left(x^t \mid x^{t-1}, \ldots, x^1\right)$$

- and also generate text by sampling from the conditional probability as in an autoregressive model

A First Language Model

- n-Grams are examples of language models
- Use the frequencies of words given the context to define the prediction probability
- **Problems**: Sparsity of co-occurrences leaves some probabilities undefined (or poorly defined), storage requirements for all n-grams are demanding

Deep Learning Language Models

- Train a neural net model to directly approximate the predictive probability (eg, use Maximum Likelihood on the corpus)

  1. Can use a fixed window (eg, Y. Bengio, et al. A Neural Probabilistic Language Model, JMLR 2003)

  2. Can use an infinite window (eg, RNNs)

  3. Can use a varying-size window (eg, Transformers)

Some Important NLP Models

- Embeddings from Language Models: **ELMo** (Peters et. al 2017)

- Bidirectional Encoders from Transformers: **BERT** (Devlin et. al 2018)

- Generative Pretrained Transformer: **GPT-3** (Radford et al. 2018)

OpenAI