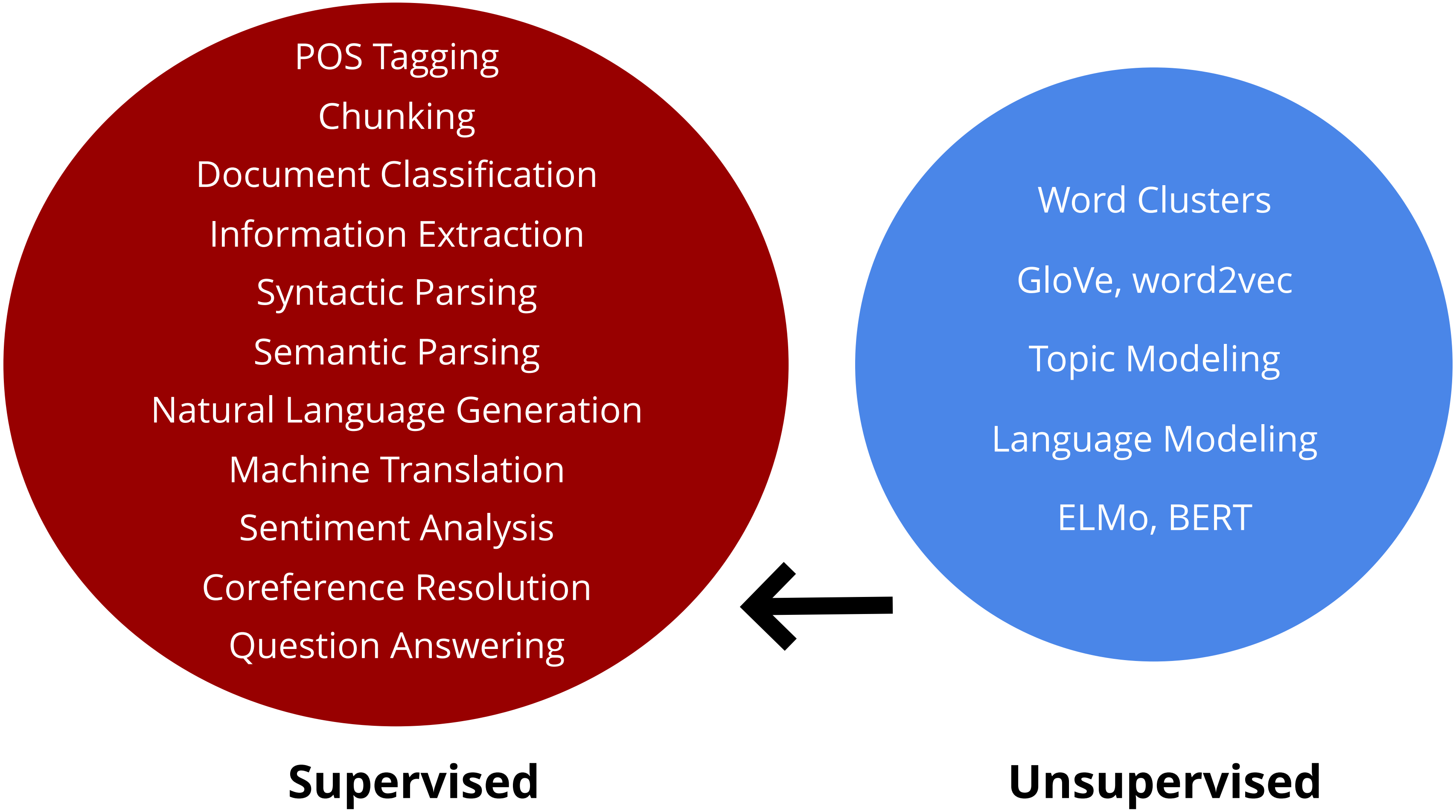
50.040
Natural Language Processing

Lu, Wei

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Tasks in NLP



POS Tagging
Chunking
Document Classification
Information Extraction
Syntactic Parsing
Semantic Parsing
Natural Language Generation
Machine Translation
Sentiment Analysis
Coreference Resolution
Question Answering
Word Clusters
GloVe, word2vec
Topic Modeling
Language Modeling
ELMo, BERT
Supervised
Unsupervised

Unsupervised Learning

Word Clusters

GloVe, word2vec

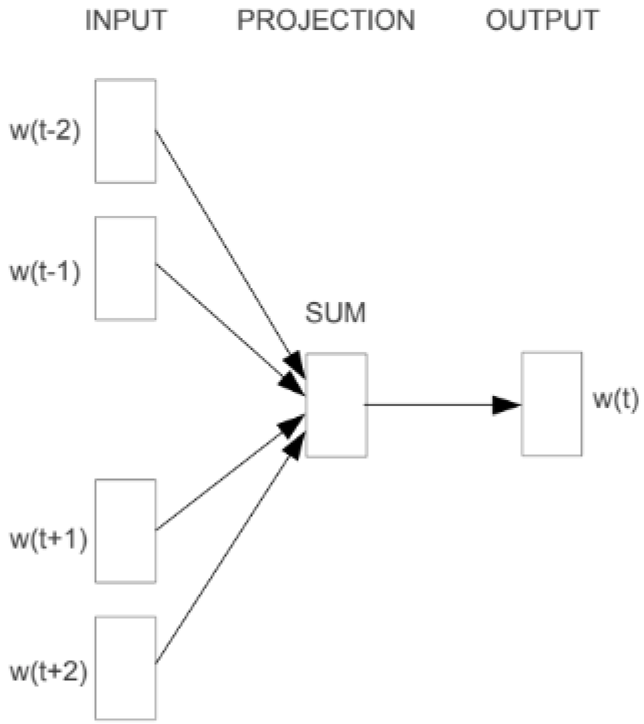
Topic Modeling

Language Modeling

ELMo, BERT

Unsupervised

Word Embeddings



CBOW

How about Sentences

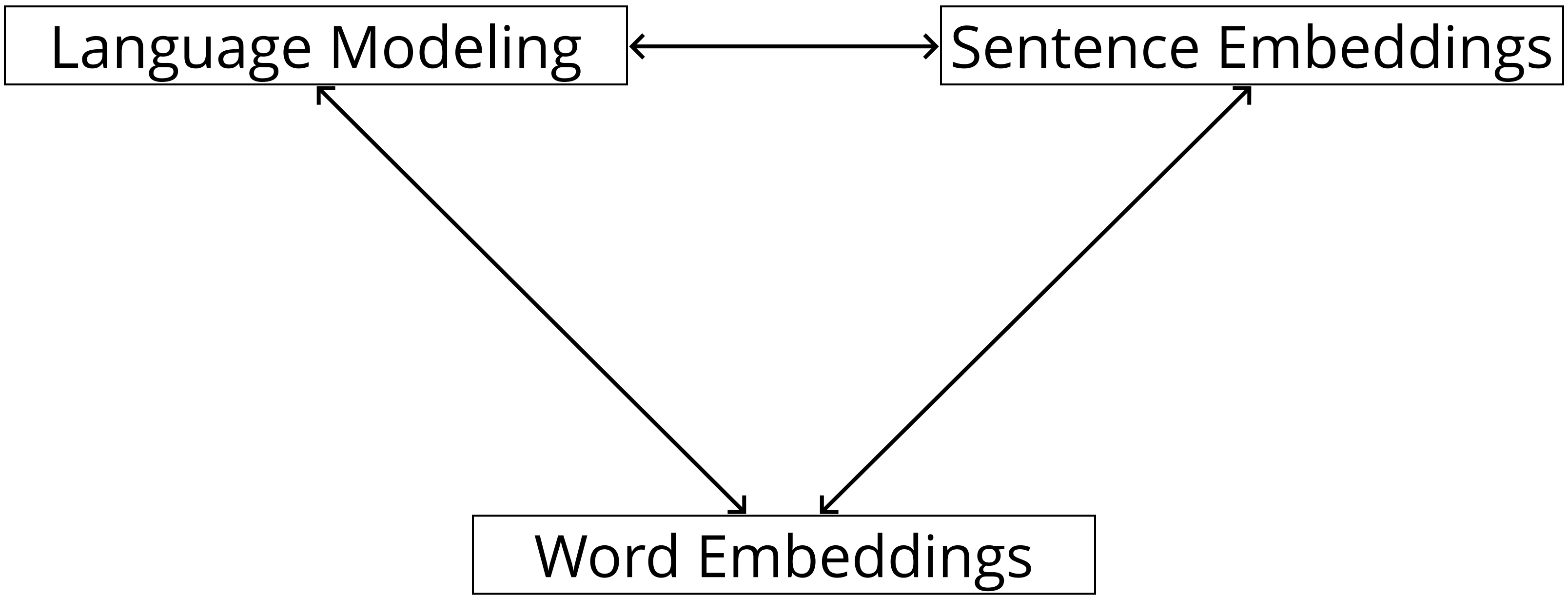
Fruit flies like a banana $\xrightarrow{\text{Language Modeling}}$ 0.00078

How likely can we see this sentence?

Fruit flies like a banana $\xrightarrow{\text{Sentence Embedding}}$ $\begin{pmatrix} 0.980 \\ -0.453 \\ 0.293 \\ 0.659 \\ -0.089 \\ 0.096 \end{pmatrix}$

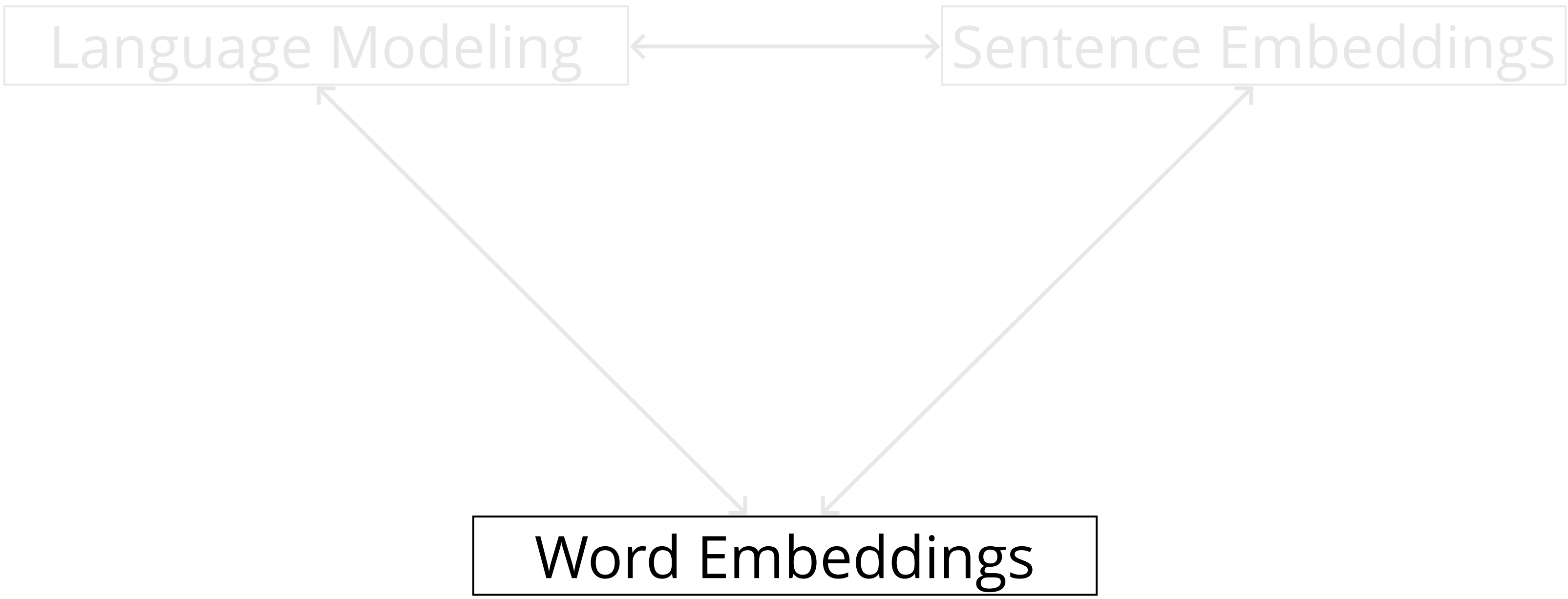
How to represent this sentence?

Three Tasks



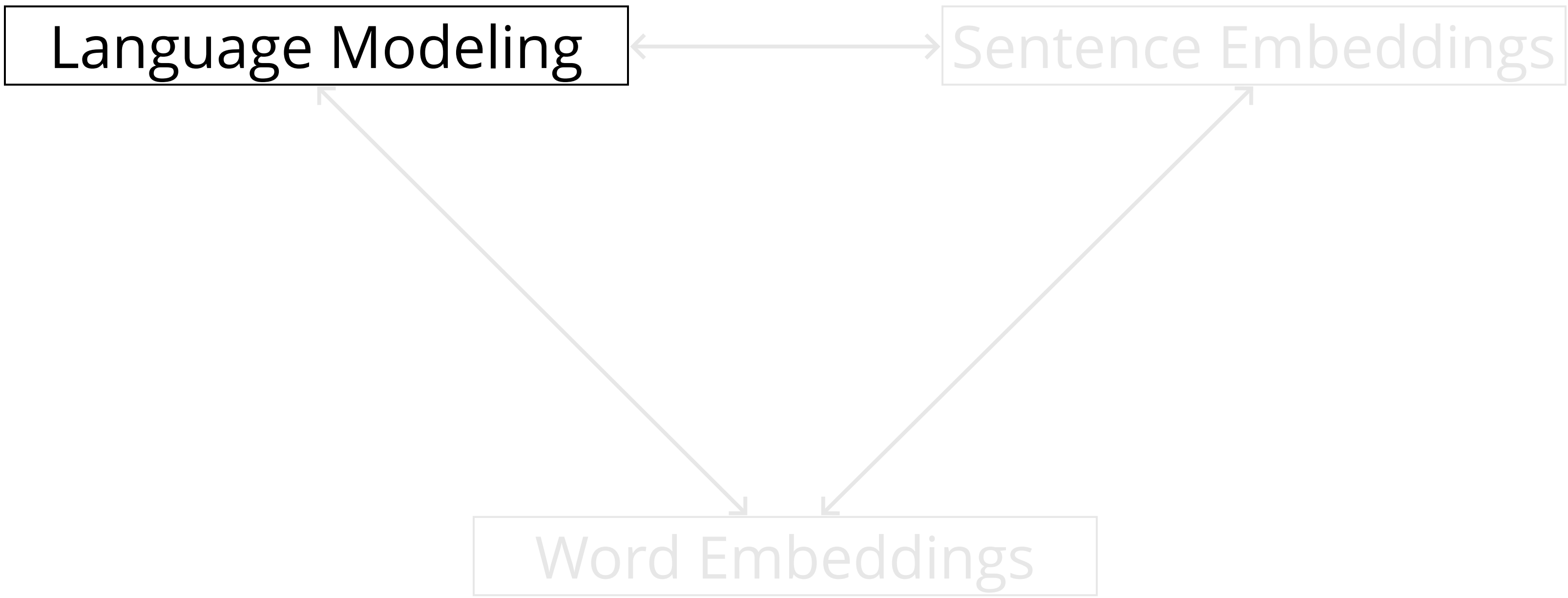
These three tasks are closely related!

Three Tasks



We have already looked at word embeddings

Three Tasks



Now let us look at language modeling

Language Modeling

Language Modeling

How likely can we see this sentence?

Fruit flies like a banana $\longrightarrow$ 0.00078

I love NLP $\longrightarrow$ 0.00428

Fruit flies $\longrightarrow$????

Language Modeling

First of all, similar to HMM, we assume each sentence is attached with a special symbol at its end: `STOP`

Fruit flies like a banana `STOP`

I love NLP `STOP`

Fruit flies `STOP`

Language Modeling

A language model consists of a finite set V, and a function $p(x_1, x_2, \ldots, x_m)$ such that:

1. For any sequence $\langle x_1, \ldots, x_m \rangle \in V^+$

$$p(x_1, \ldots, x_m) \geq 0$$

2. In addition,

$$\sum_{\langle x_1, \ldots, x_m \rangle \in V^+} p(x_1, x_2, \ldots, x_m) = 1$$

Hence, $p(x_1, \ldots, x_m)$ is a probability distribution over the sentences defined by V^+.

Question

How to Learn a Language Model Based on a Corpus?

Data Set $\mathcal{D}$

Language Model

A simple Maximum Likelihood Estimator gives:

$$p(x_1, \ldots, x_m) = \frac{count(x_1, \ldots, x_m)}{\sum_{s \in \mathcal{D}} count(s)}$$

Is this feasible?

Language Model

A simple Maximum Likelihood Estimator gives:

$$p(x_1, \ldots, x_m) = \frac{count(x_1, \ldots, x_m)}{\sum_{s \in \mathcal{D}} count(s)}$$

Is this feasible?

NO! The training set does not contain all possible sentences! It does not generalize to new sentences!

Alternative Approach

Recall what we did in a Generative Model?

$$p(x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_1, \ldots, x_{i-1})$$

Is this feasible?

Alternative Approach

Recall what we did in a Generative Model?

$$p(x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_1, \ldots, x_{i-1})$$

Is this feasible?

NO! The sequence that we condition on may only appear a few times in the training set. Poor generalization again!

Markov Assumption

Recall what we did in Naive Bayes / HMM?

$$p(x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_1, \ldots, x_{i-1})$$

In other words, we only consider the previous $(n-1)$ words.

n-Gram Language Model

$$p(x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_{i-n+1}, \ldots, x_{i-1})$$

Bigram Language Model

$$n = 2$$

$$p(x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_{i-1})$$

Similar to what we did for HMM, we may introduce

$$x_0 = \texttt{START}$$

Bigram Language Model

$\texttt{START}$ $\texttt{STOP}$

$$p(x_0, x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_{i-1})$$

Question

How to Learn a Bigram Language Model?

$$p(x_0, x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_{i-1})$$

Bigram Model

How to do parameter estimation?

$$p(x_0, x_1, x_2, \ldots, x_m) = \prod_{i=1,...,m} \underline{p(x_i | x_{i-1})}$$

These are the Model Parameters

Objective:

$$\mathcal{L} = \prod_{\langle x_1,...,x_m \rangle \in \mathcal{D}} p(x_0, x_1, \ldots, x_m)$$

Bigram Model

The number of times we see the word u is followed by the word v in $\mathcal{D}$.

The number of times we see the word v in $\mathcal{D}$.

$$p(u|v) = \frac{\text{count}(u,v)}{\text{count}(v)}$$

$$\text{for all } u \in V, v \in V \cup \{\texttt{START}\}$$

Trigram Model

$\mathtt{START}$ $\mathtt{STOP}$

$$p(x_{-1}, x_0, x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i | x_{i-2}, x_{i-1})$$

$$p(u|w,v) = \frac{\text{count}(w,v,u)}{\text{count}(w,v)},\ \text{for all}\ u \in V, w, v \in V \cup \{\mathtt{START}\}$$

Unigram Model

$\texttt{STOP}$

$$p(x_1, x_2, \ldots, x_m) = \prod_{i=1,\ldots,m} p(x_i)$$

$$p(u) = \frac{\text{count}(u)}{\text{c}}, \text{ for all } u \in V.$$

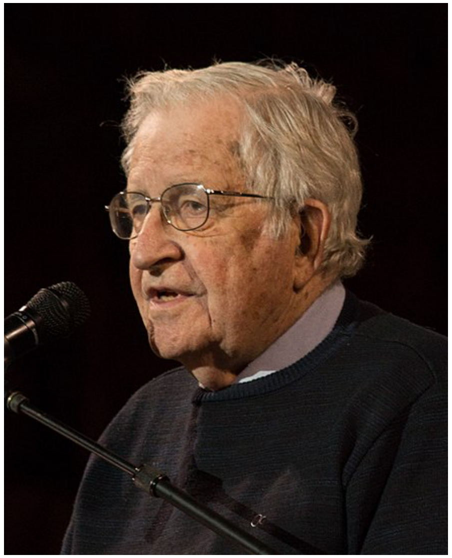
The total number of words in the corpus $\mathcal{D}$.

n-Grams
Are they useful?

n-Gram Model

“It must be recognized that the notion ‘probability of a sentence’ is an entirely useless one, under any known interpretation of this term.”

- Noam Chomsky (1967)



☹

n-Gram Model

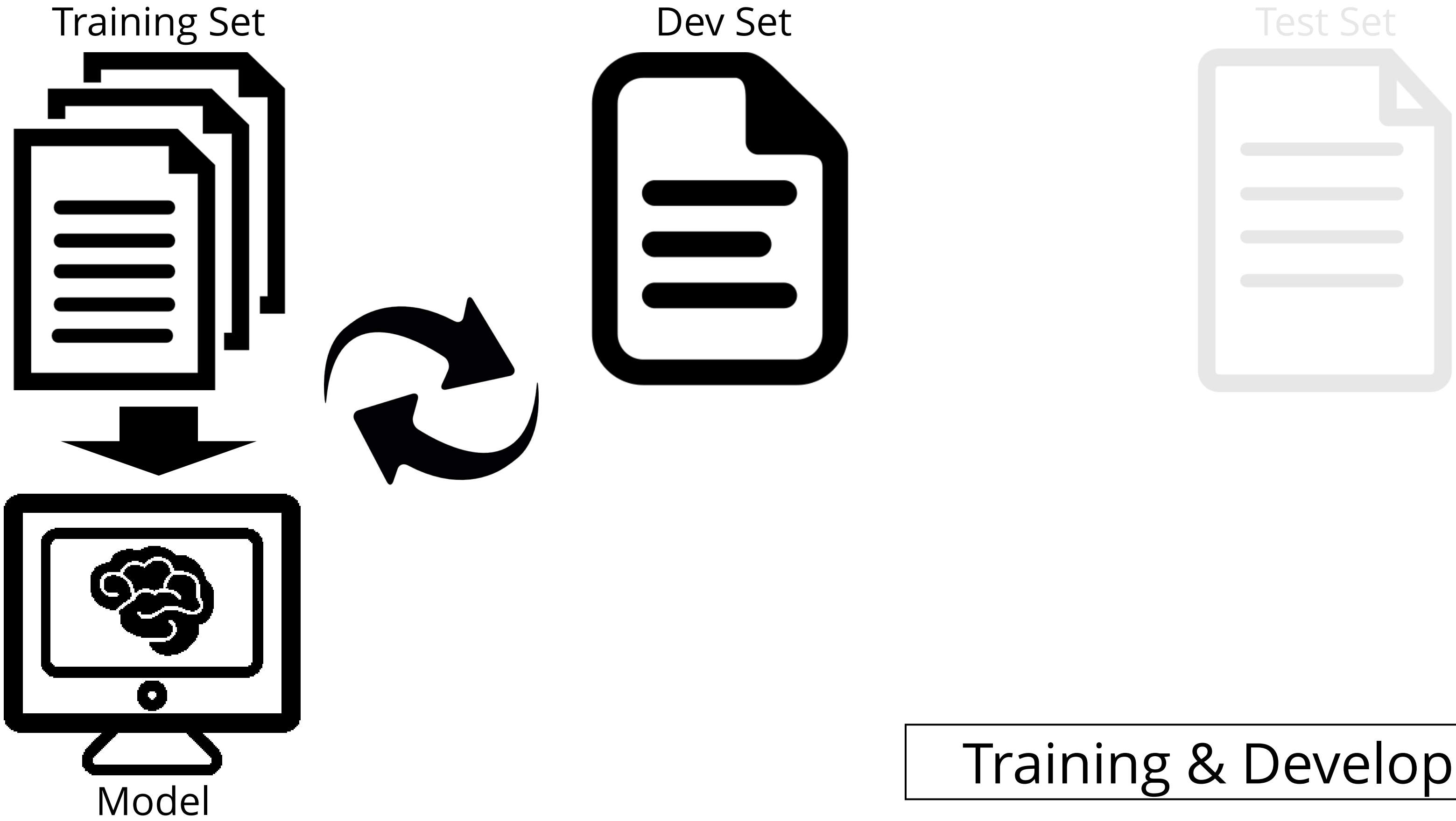
"Every time I fire a linguist, the performance of the recognition system goes up"

- Fred Jelinek (1988)

☺

Language Model Evaluation

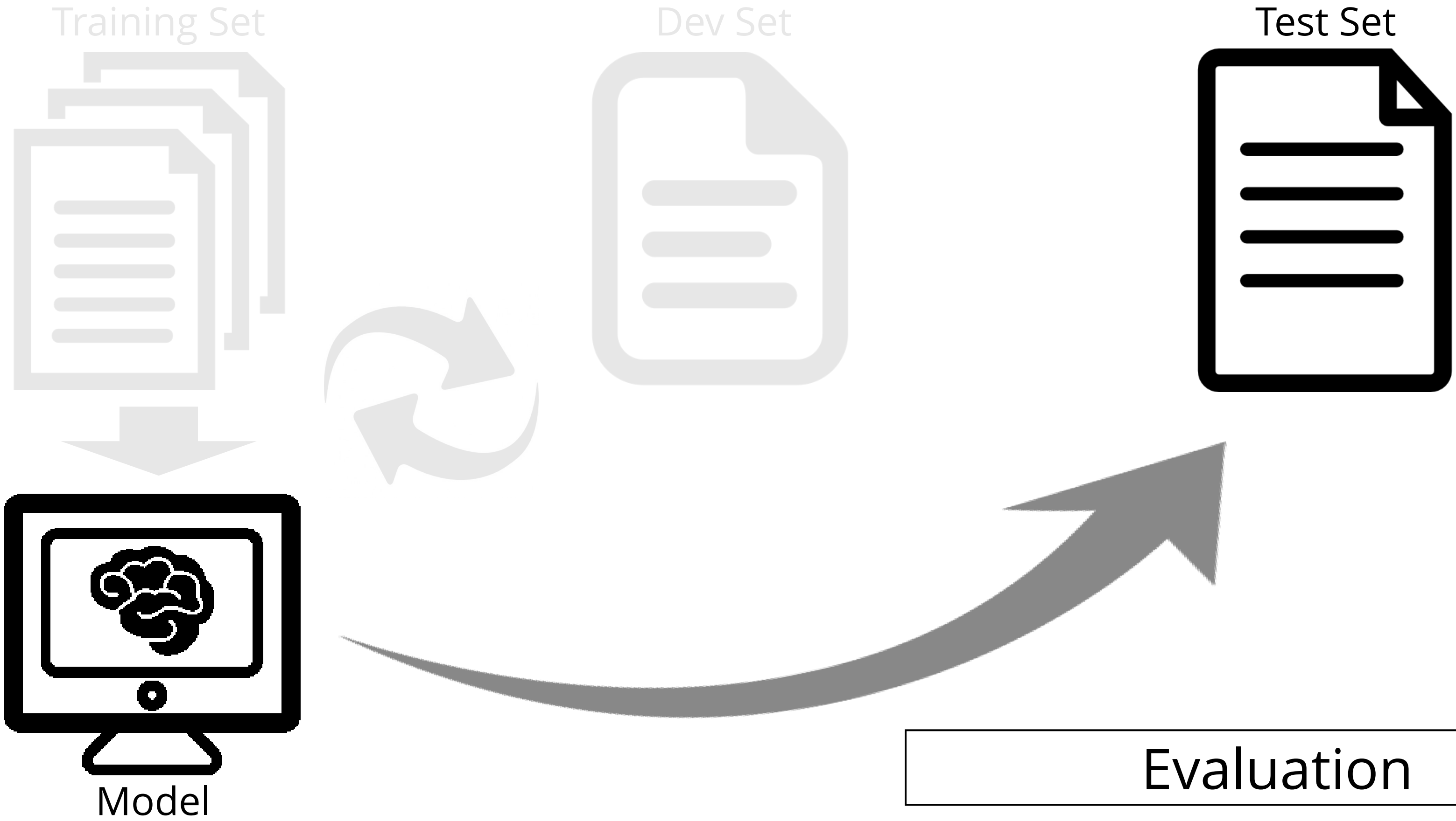
How do we evaluate a model?



Training & Development

Language Model Evaluation

How do we evaluate a model?



Evaluation

Evaluation Metric

One idea: the likelihood of the entire test/dev set $\mathcal{D}'$!

$$\prod_{j=1}^{|\mathcal{D}'|} p(\mathbf{x}^{(j)})$$

$$\log_2 \prod_{j=1}^{|\mathcal{D}'|} p(\mathbf{x}^{(j)})$$

$$\sum_{j=1}^{|\mathcal{D}'|} \log_2 p(\mathbf{x}^{(j)})$$

Perplexity

$$\sum_{j=1}^{|\mathcal{D}'|} \log_2 p(\mathbf{x}^{(j)})$$

$$\frac{1}{c'} \sum_{j=1}^{|\mathcal{D}'|} \log_2 p(\mathbf{x}^{(j)})$$

Total number of words in the set $\mathcal{D}'$.

$$2^{-\ell} \text{ where } \ell = \frac{1}{c'} \sum_{j=1}^{|\mathcal{D}'|} \log_2 p(\mathbf{x}^{(j)})$$

Perplexity

Perplexity

$$2^{-\ell} \text{ where } \ell = \frac{1}{c'} \sum_{j=1}^{|\mathcal{D}'|} \log_2 p(\mathbf{x}^{(j)})$$

$$1 / \sqrt[c']{\prod_{j=1}^{|\mathcal{D}'|} p(\mathbf{x}^{(j)})}$$

c' terms after expansions

Multiplicative inverse of the geometric mean of the terms $p(x_k^{(j)} | x_{k-2}^{(j)}, x_{k-1}^{(j)})$.

Trigram Model

$$p(u|w,v) = \frac{\text{count}(w,v,u)}{\text{count}(w,v)}$$

What are the potential problems with such a model (or in general, an *n*-gram model)?

Trigram Model

$$p(u|w,v) = \frac{\text{count}(w,v,u)}{\text{count}(w,v)}$$

What are the potential problems with such a model (or in general, an *n*-gram model)?

Many counts could be zeros!

Trigram Model

$$p(u|w,v) = \frac{\mathrm{count}(w,v,u)}{\mathrm{count}(w,v)}$$

What are the potential problems with such a model (or in general, an *n*-gram model)?

$\boxed{\text{Many counts could be zeros!}}$

One solution:

Smoothing

Smoothing: Interpolation

$$p(u|w,v) = \frac{\text{count}(w,v,u)}{\text{count}(w,v)}$$

$$p(u|v) = \frac{\text{count}(v,u)}{\text{count}(v)}$$

Smoothed probability

$$p(u) = \frac{\text{count}(u)}{c}$$

$$\boxed{q(u|w,v)} = \lambda_1 p(u) + \lambda_2 p(u|v) + \lambda_3 p(u|w,v)$$

$$\boxed{\lambda_1 + \lambda_2 + \lambda_3} = 1$$

hyperparameters

Smoothing: Interpolation

$$q(u|w,v) = \lambda_1 p(u) + \lambda_2 p(u|v) + \lambda_3 p(u|w,v)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Hyperparameters will be tuned on the development set.

Each is indicating the significance / contribution / confidence on the corresponding model.

Laplace Smoothing

"Add-one" Smoothing

$$p(u|w,v) = \frac{\text{count}(w,v,u)}{\text{count}(w,v)}$$

$$p(u|w,v) = \frac{\text{count}(w,v,u)+1}{\text{count}(w,v)+U}$$

What is U? The U is chosen such that the sum of these p terms will be 1!

In other words, $U = |V|$

Smoothing

Other smoothing techniques exist:

- Good-turing smoothing
- Kneser-Ney smoothing
- Witten-Bell smoothing
- Katz smoothing
- Church and Gale smoothing

Read relevant book chapters to learn more.

Trigram Model

$$p(u|w,v) = \frac{\text{count}(w,v,u)}{\text{count}(w,v)}$$

What are the potential problems with such a model (or in general, an *n*-gram model)?

Many counts could be zeros!

One solution:

Smoothing ✔

Question

After smoothing, how many model parameters do we have to store?

Curse of Dimensionality

$$\text{in the order of } |V|^n$$

Too many model parameters!

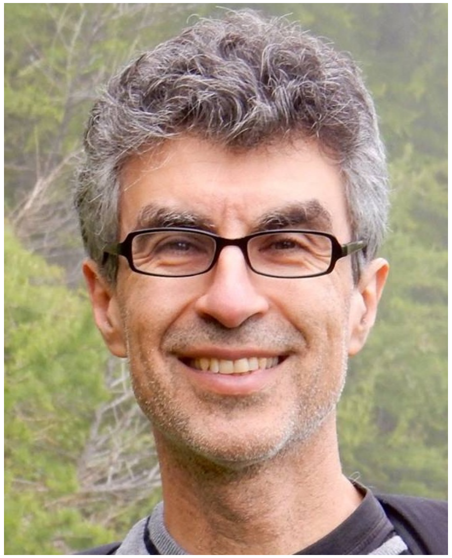
One Solution:

Learn language model with word embeddings!

Neural Language Model

"The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations."

- Yoshua Bengio et al. (2003)

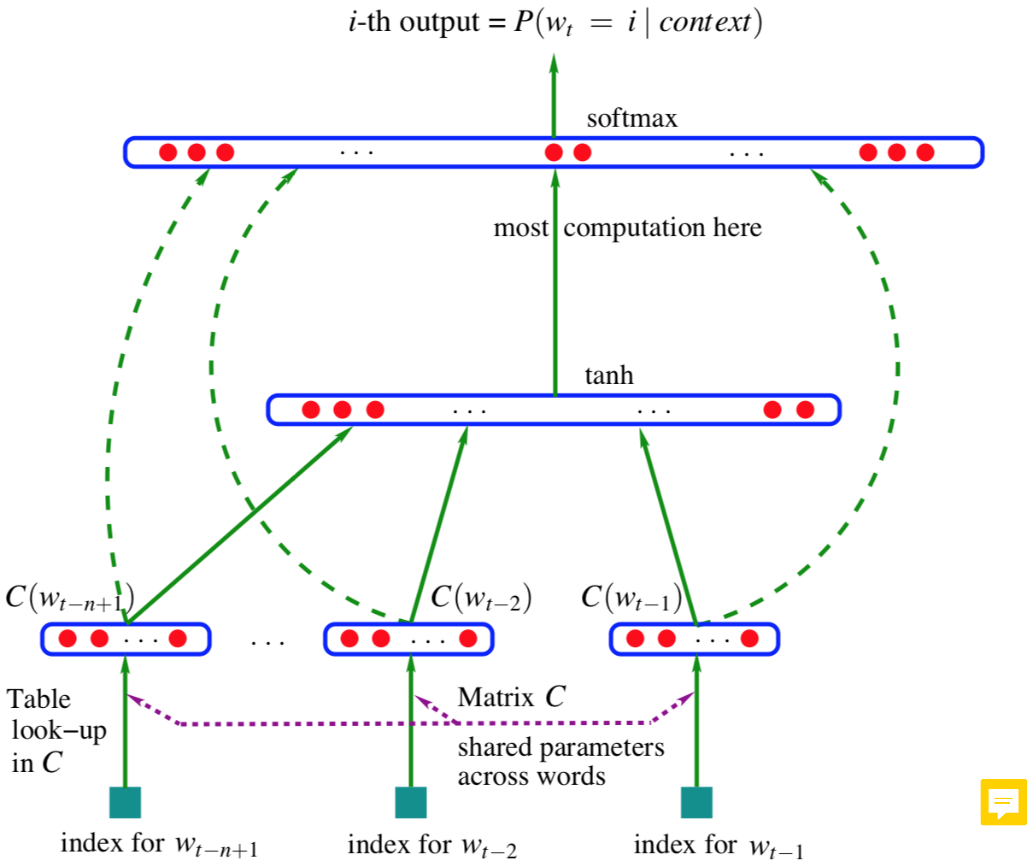


Neural Language Model

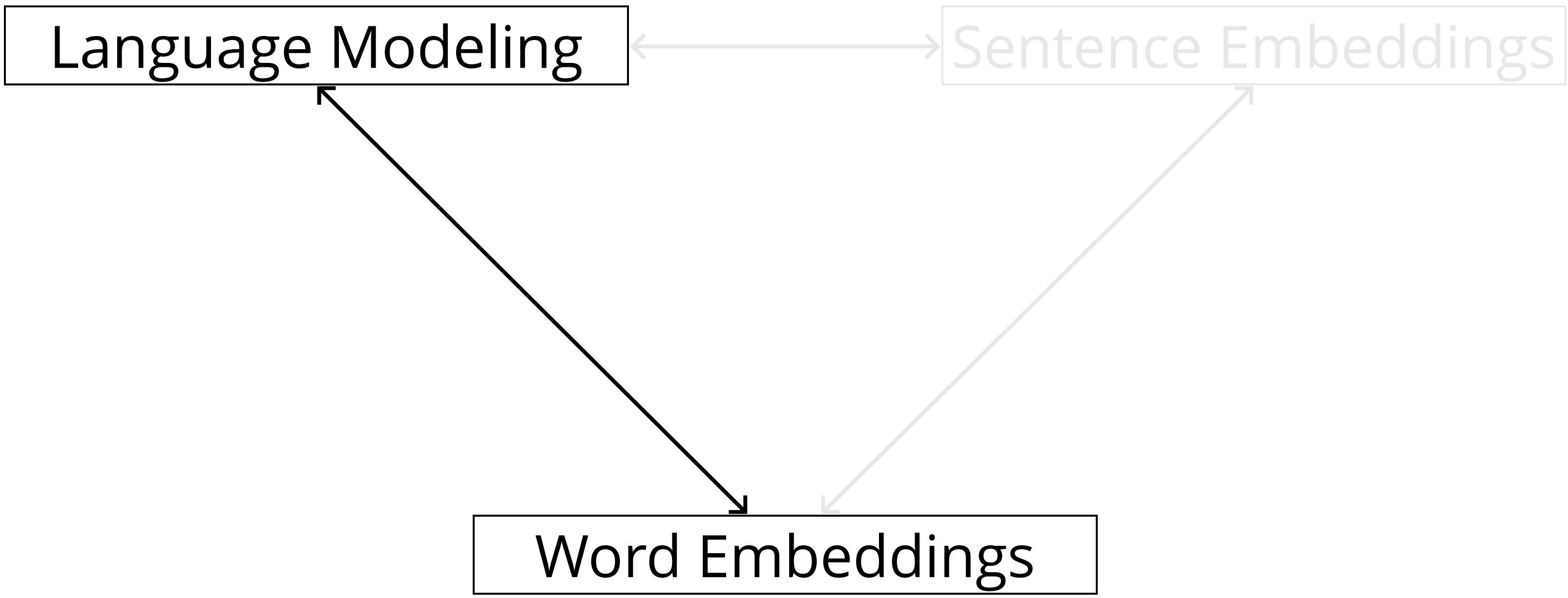
Bengio et al. (2003)

1. associate with each word in the vocabulary a distributed *word embedding*,
2. express the joint probability function of word sequences in terms of the embeddings of these words in the sequence, and
3. learn simultaneously the *word embeddings* and the parameters of that *probability function*.

Neural Language Model

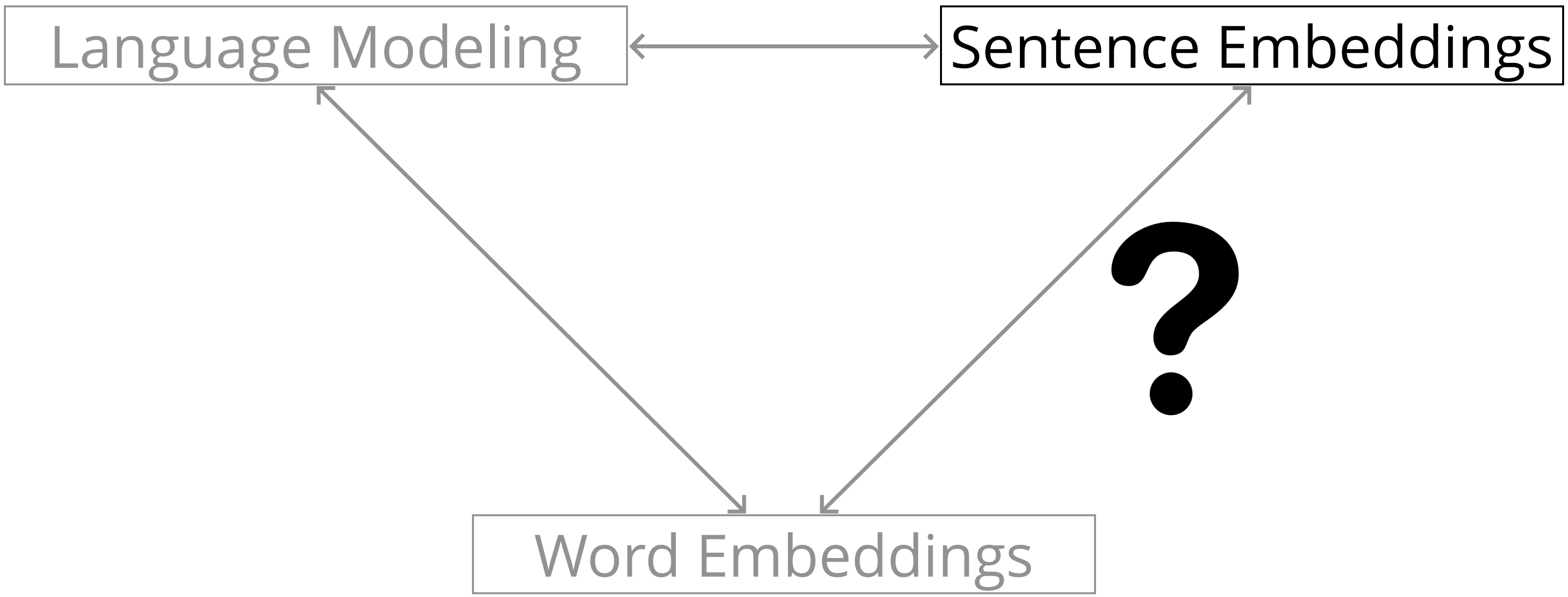


Three Tasks



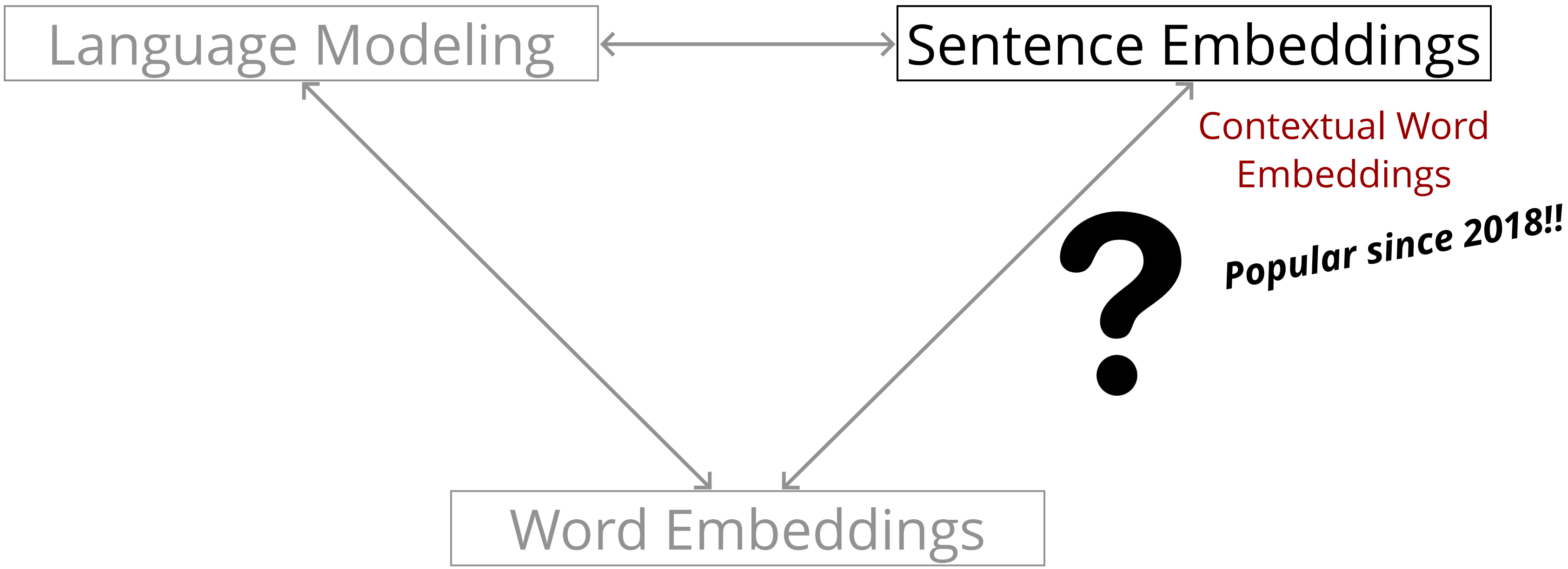
These three tasks are closely related!

Three Tasks



These three tasks are closely related!

Three Tasks



These three tasks are closely related!