How to predict the next word in a sequence?

Sandeep Soni

10/03/2024

Classification

x	y
It was fantastic!!!!	😀
It could have been better but wasn't bad	😐
What a waste of time and money!	😡

- Predict y from x, i.e., $\hat{y} = \hat{h}(x)$
- Model $\hat{h}$ as P(Y|X)
- X is high-dimensional
- P(😀|"the", "movie", "was", "fantastic")

Classifiers

- Naive Bayes
  - Maximize P(X,Y) and then use Bayes theorem
  - Conditional independence assumptions
    $P(\mathrm{X}_i|\mathrm{Y}) = P(\mathrm{X}_i|\mathrm{Y}, \mathrm{X}_{i-1})$
  - $P(\mathrm{X}_i|\mathrm{Y})$ is estimated simply by counting
  - Smoothing is required for estimating probabilities

Generative

Classifiers

- Naive Bayes
  - Maximize P(X,Y)
  - Assume $P(\mathrm{X}_i|\mathrm{Y}) = P(\mathrm{X}_i|\mathrm{Y}, \mathrm{X}_{i-1})$
  - (Smoothed) Count estimates for $P(\mathrm{X}_i|\mathrm{Y})$

Generative

- Logistic Regression
  - Maximize P(Y|X)
  - $P(\mathrm{Y}|\mathrm{X}) = \sigma(\mathbf{w}^\top \mathbf{x})$
  - (Regularized) Weights learned using iterative algorithms

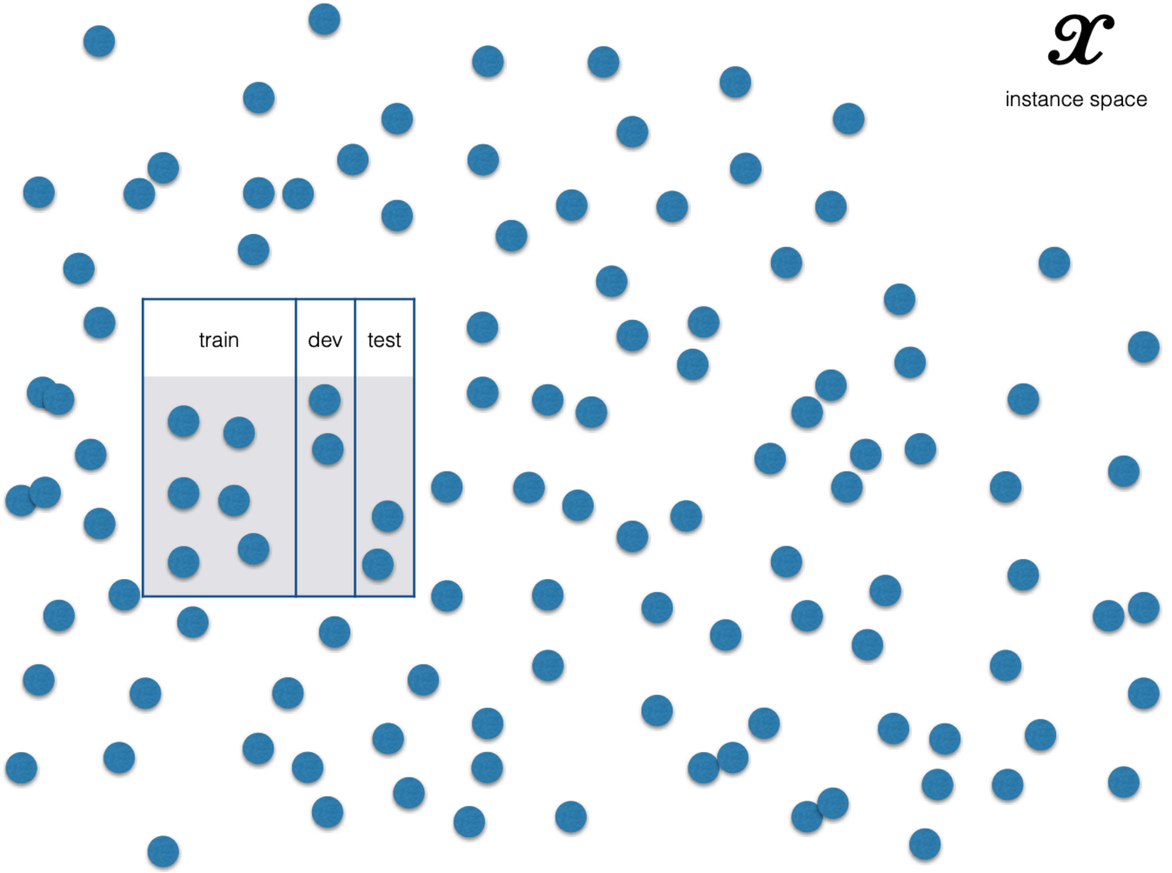
Discriminative

Supervised learning

Train: Estimate weights, set hyperparameters

Dev: Evaluate performance

Test: Do final evaluation



Slide from David Bamman

Evaluation

	1	2	3	4	5	6	7	8	9	10
y	spam	spam	spam	spam	spam	ham	ham	ham	ham	ham
yhat	spam	spam	ham	spam	spam	spam	ham	spam	ham	spam

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{N}$$

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

$$\text{F}_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluation

	1	2	3	4	5	6	7	8	9	10
y	spam	spam	spam	spam	spam	ham	ham	ham	ham	ham
yhat	spam	spam	ham	spam	spam	spam	ham	spam	ham	spam

	y=spam	y=ham
yhat = spam	4	3
yhat=ham	1	2

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{N}$$

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluation

	1	2	3	4	5	6	7	8	9	10
y	spam	spam	spam	spam	spam	ham	ham	ham	ham	ham
yhat	spam	spam	ham	spam	spam	spam	ham	spam	ham	spam

	y=spam	y=ham
yhat = spam	4	3
yhat=ham	1	2

	y=spam	y=ham
yhat = spam	True positives	False positives
yhat=ham	False negatives	True negatives

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{N}$$

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Evaluation

	1	2	3	4	5	6	7	8	9	10
y	spam	spam	spam	spam	spam	ham	ham	ham	ham	ham
yhat	spam	spam	ham	spam	spam	spam	ham	spam	ham	spam

	y=spam	y=ham
yhat = spam	4	3
yhat=ham	1	2

	y=spam	y=ham
yhat = spam	True positives	False positives
yhat=ham	False negatives	True negatives

$$\text{Accuracy} = \frac{\text{True positives} + \text{True negatives}}{N}$$

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$N = \text{True positives} + \text{True negatives} + \text{False positives} + \text{False negatives}$$

In class

- Text classification demo

Classification

What is the probability of a label given text i.e. P(y|x)?

What is the probability of text i.e. P(x)?

Why should we model P(x)?

- Many tasks are sequence-to-sequence tasks
  - Translation: source language text -> target language text
  - Summarization: long text -> short text

Why should we model P(x)?

- We want the output sequences for such tasks to be “fluent”
- How can we *quantify* fluency?

El cafe negro me gusta mucho

Machine translation system I

The coffee black me pleases much

El cafe negro me gusta mucho

Machine translation system II

I love dark coffee

Machine translation system I

The coffee black me pleases much

P(“The coffee black me pleases much”)

<

Machine translation system II

I love dark coffee

P(“I love dark coffee”)

Bayes rule

$$P(X_e \mid X_s)$$

$$P(X_e \mid X_s) \propto P(X_e, X_s)$$

$$\propto P(X_s \mid X_e) \times P(X_e)$$

Noisy channel model

$$P_{\texttt{engish} \leftarrow \texttt{spanish}}(\mathbf{w}^e \mid \mathbf{w}^s) \propto P_{\texttt{engish,spanish}}(\mathbf{w}^e, \mathbf{w}^s)$$

$$= \boxed{P_{\texttt{spanish} \leftarrow \texttt{english}}(\mathbf{w}^{(s)} \mid \mathbf{w}^{(e)})} \times \boxed{P_{\texttt{english}}(\mathbf{w}^{(e)})}$$

Translation model

Language model

We want the noisy channel model to have high fidelity to the source text and high fluency in the generated output

“When I look at an article in Russian, I say: `This is really written in English but it has been coded in some strange symbols. I will now proceed to decode.”

—Warren Weaver (1955)

Language model

- $\mathscr{V}$ is a vocabulary of symbols that need to be generated
- $\mathcal{S}$ is a infinite set of sequences of symbols taken from $\mathscr{V}$; each sequence ends with a special token STOP
- A sequence $x \in \mathcal{S}$

Language model

- $P(x) = P(x_1, x_2, \ldots, x_n)$
- P("I will be back") = P("I", "will", "be", "back", STOP)
- $0 \leq P(x) \leq 1$
- $\sum_{x' \in \mathcal{S}} P(x') = 1$

Language model

- Language modeling is a task of estimating a probabilistic model over words i.e. P(x)
- This is not easy!
- What is the P(“to be or not to be”)?

Chain Rule

Chain Rule

$P(x_1, x_2, \ldots, x_n)$

$P(\texttt{to be or not to be})$

Chain Rule

$P(x_1, x_2, \ldots, x_n)$

$P(\texttt{to be or not to be})$

$$P(x_1, x_2, \ldots, x_n) = P(x_1)$$

$$\times P(x_2 \,|\, x_1)$$

$$\times P(x_3 \,|\, x_1, x_2)$$

$$\ldots$$

$$\times P(x_n \,|\, x_1, \ldots, x_{n-1})$$

Chain Rule

$$P(x_1, x_2, \ldots, x_n)$$

$$P(\texttt{to be or not to be})$$

$$P(x_1, x_2, \ldots, x_n) = P(x_1)$$

$$P(\texttt{to be or not to be}) = P(\texttt{to})$$

$$\times P(x_2 \mid x_1)$$

$$\times P(\texttt{be} \mid \texttt{to})$$

$$\times P(x_3 \mid x_1, x_2)$$

$$\times P(\texttt{or} \mid \texttt{to}, \texttt{be})$$

$$\ldots$$

$$\ldots$$

$$\times P(x_n \mid x_1, \ldots, x_{n-1})$$

$$\times P(\texttt{be} \mid \texttt{to}, \texttt{be}, \ldots, \texttt{to})$$

Chain Rule

$$P(\texttt{to be or not to be}) = P(\texttt{to})$$

$$\times P(\texttt{be}|\texttt{to})$$

$$\times P(\texttt{or}|\texttt{to},\texttt{be})$$

$$\ldots$$

$$\times P(\texttt{be}|\texttt{to},\texttt{be},\ldots,\texttt{to})$$

P(“be” | “to be or not to”) is hard to estimate. Why?

Chain Rule

$$P(\texttt{to be or not to be}) = P(\texttt{to})$$

$$\times P(\texttt{be}|\texttt{to})$$

$$\times P(\texttt{or}|\texttt{to},\texttt{be})$$

$$\ldots$$

$$\times P(\texttt{be}|\texttt{to},\texttt{be},\ldots,\texttt{to})$$

P(“be” | “to be or not to”) is hard to estimate. Can we make some assumptions to make it tractable?

Markov assumptions

$$P(x_i \mid x_{i-1}, \ldots, x_1) \approx P(x_i)$$

zero-order

$$P(x_i \mid x_{i-1}, \ldots, x_1) \approx P(x_i \mid x_{i-1})$$

first-order

$$P(x_i \mid x_{i-1}, \ldots, x_1) \approx P(x_i \mid x_{i-2}, x_{i-1})$$

second-order

N-gram language models

$$P(x) = \prod_i P(x_i) \times P(\text{STOP})$$

unigram

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

$$P(x) = \prod_i P(x_i \mid x_{i-2}, x_{i-1}) \times P(\text{STOP} \mid x_{n-1}, x_n)$$

trigram

Bigram Generation

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

START _____

Previous word	Current word	Probability
START	I	0.2
START	like	0.0008
…	…	…
I	like	0.1
I	coffee	0.0003
…	…	…
coffee	STOP	0.06
I	STOP	0.0001

Bigram Generation

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

Previous word	Current word	Probability
START	I	0.2
START	like	0.0008
…	…	…
I	like	0.1
I	coffee	0.0003
…	…	…
coffee	STOP	0.06
I	STOP	0.0001

START I

START I _____

Bigram Generation

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

Previous word	Current word	Probability
START	I	0.2
START	like	0.0008
…	…	…
I	like	0.1
I	coffee	0.0003
…	…	…
coffee	STOP	0.06
I	STOP	0.0001

START I

START I like

START I like _____

Bigram Generation

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

Previous word	Current word	Probability
START	I	0.2
START	like	0.0008
…	…	…
I	like	0.1
I	coffee	0.0003
…	…	…
coffee	STOP	0.06
I	STOP	0.0001

START I

START I like

START I like black

START I like black _____

Bigram Generation

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

Previous word	Current word	Probability
START	I	0.2
START	like	0.0008
…	…	…
I	like	0.1
I	coffee	0.0003
…	…	…
coffee	STOP	0.06
I	STOP	0.0001

START I

START I like

START I like black

START I like black coffee

START I like black coffee ____

Bigram Generation

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

Previous word	Current word	Probability
START	I	0.2
START	like	0.0008
…	…	…
I	like	0.1
I	coffee	0.0003
…	…	…
coffee	STOP	0.06
I	STOP	0.0001

START I

START I like

START I like black

START I like black coffee

START I like black coffee STOP

ESTIMATION

Maximum likelihood estimation

$$P(x) = \prod_i P(x_i) \times P(\text{STOP})$$

unigram

$$P(x_i) = \frac{c(x_i)}{N}$$

$$P(x) = \prod_i P(x_i \mid x_{i-1}) \times P(\text{STOP} \mid x_n)$$

bigram

$$P(x_i \mid x_{i-1}) = \frac{c(x_i, x_{i-1})}{c(x_{i-1})}$$

$$P(x) = \prod_i P(x_i \mid x_{i-2}, x_{i-1}) \times P(\text{STOP} \mid x_{n-1}, x_n)$$

trigram

$$P(x_i \mid x_{i-2}, x_{i-1}) = \frac{c(x_i, x_{i-1}, x_{i-2})}{c(x_{i-1}, x_{i-2})}$$

Data sparsity

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 4.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

SLP 4.3

If probabilities are estimated from a fixed finite sample, we'll encounter situations where we won't see some terms, bigrams, etc?

Data sparsity

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 4.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

SLP 4.3

How to calculate P(x)? How to estimate P(x_i)?

Smoothing

Maximum likelihood estimate

$$P(x_i) = \frac{c(x_i)}{N}$$

Smoothed estimate

$$P(x_i) = \frac{c(x_i) + \alpha}{N + V\alpha}$$

Various different ways to do smoothing; other solutions include interpolation

Evaluation

- How do we know if our language model is good?
- Intrinsic: How good is an LM at generating text?
- Extrinsic: How much does an LM improve the performance of a MT model?

Evaluation

- A good language model should assign high probability to plausible unseen sequences
- A good language model should be less perplexed or surprised
- Perplexity = inverse probability of test data, averaged over words

Perplexity

- Perplexity = $2^{-\frac{l(w)}{M}}$, where M is the total number of unseen tokens
- $l(w) = \sum_{m=1}^{M} \log P(w_m \mid w_{m-1}, \ldots, w_1)$
- Based on the language model of your choice, you'll calculate l(w)
- Smaller perplexity is better!

No Country for Old Members: User Lifecycle and Linguistic Change in Online Communities

Cristian Danescu-Niculescu-Mizil
Stanford University
Max Planck Institute SWS
cristiand@cs.stanford.edu

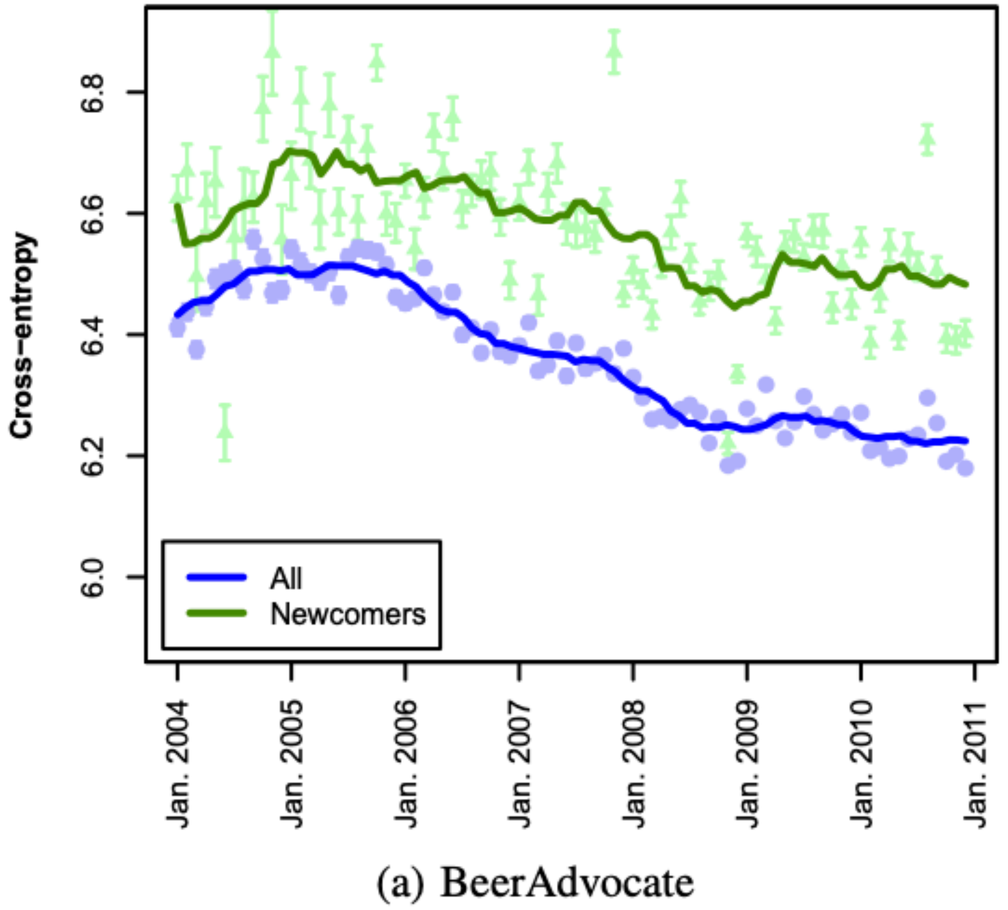
Robert West
Stanford University
west@cs.stanford.edu

Dan Jurafsky
Stanford University
jurafsky@stanford.edu

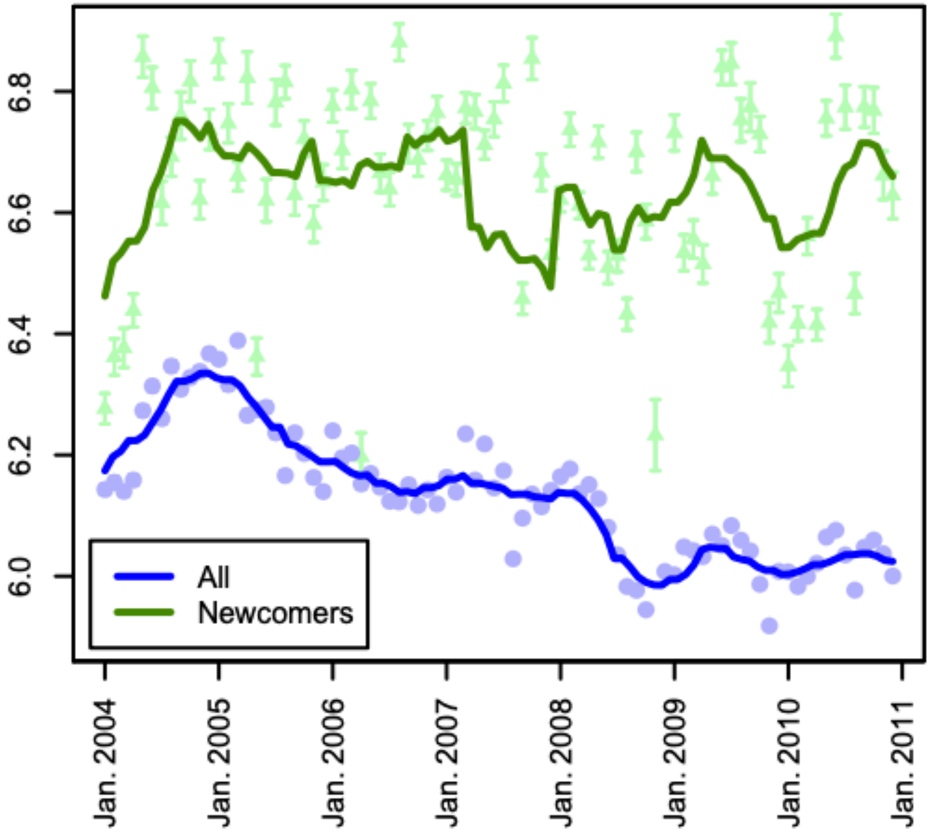
Jure Leskovec
Stanford University
jure@cs.stanford.edu

Christopher Potts
Stanford University
cgpotts@stanford.edu

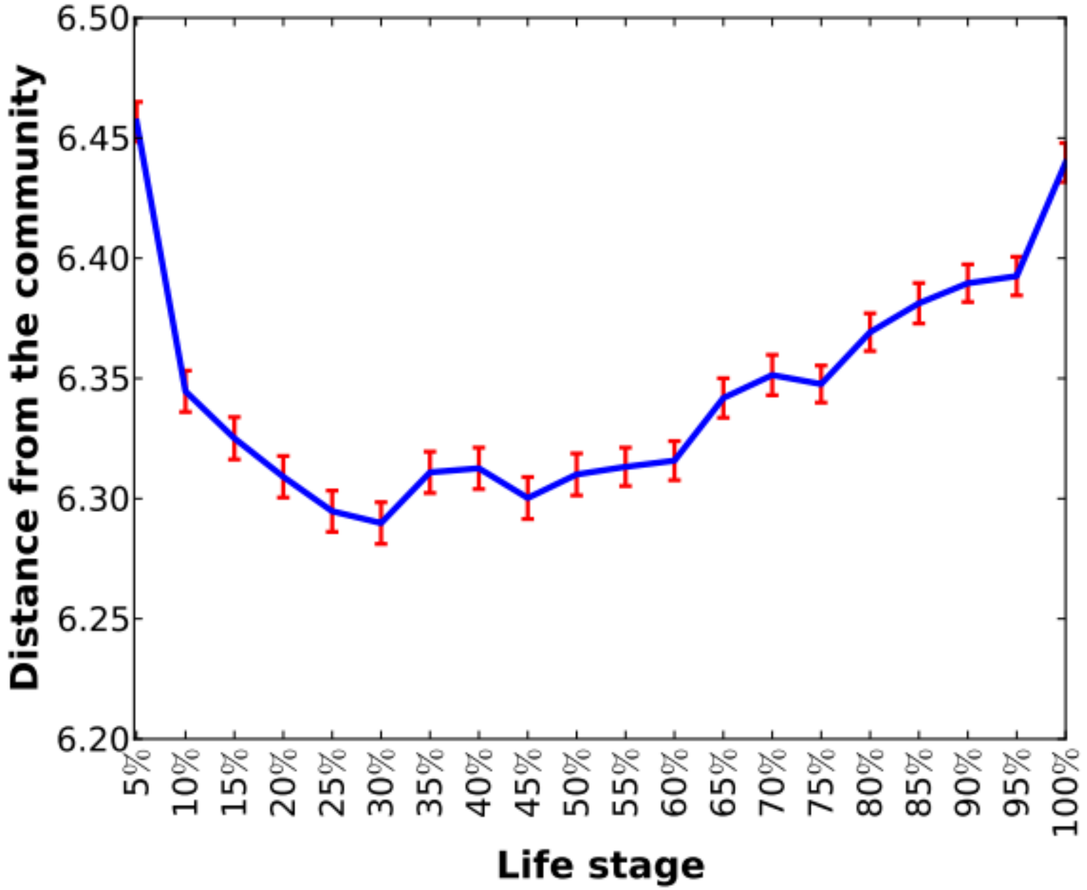
Who contributes to changing linguistic norms?

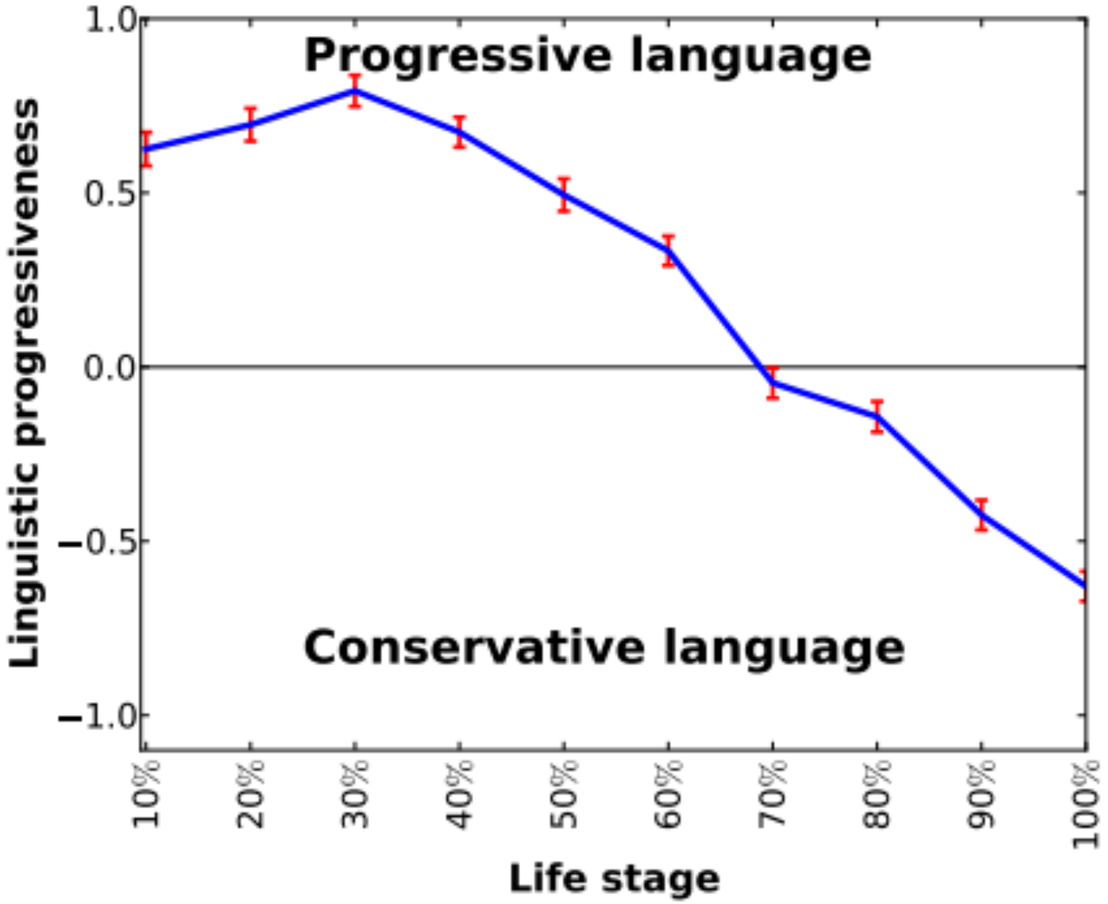


(a) BeerAdvocate



(b) RateBeer





- Users conform to the community's language initially but then stop adapting after some point

- Users are more innovative and trend-setting with their language use initially but then stabilize to rely increasingly more on past language

Can we do better than N-gram language models?

Generative Vs Discriminative

	Naive Bayes	Logistic Regression
Type of classifier	Generative	Discriminative
Model	P(x,y)	P(y | x)
Objective	Generate the data and label jointly	Predict the label from the data

Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors

Marco Baroni and **Georgiana Dinu** and **Germán Kruszewski**
Center for Mind/Brain Sciences (University of Trento, Italy)
(marco.baroni|georgiana.dinu|german.kruszewski)@unitn.it

Abstract

Context-predicting models (more commonly known as embeddings or neural language models) are the new kids on the distributional semantics block. Despite the buzz surrounding these models, the literature is still lacking a systematic comparison of the predictive models with classic, count-vector-based distributional semantic approaches. In this paper, we perform such an extensive evaluation, on a wide range of lexical semantics tasks and across many parameter settings. The results, to our own surprise, show that the buzz is fully justified, as the context-predicting models obtain a thorough and resounding victory against their count-based counterparts.

1 Introduction

A long tradition in computational linguistics has shown that contextual information provides a good approximation to word meaning, since semanti-

optimization process is generally unsupervised, and based on independent considerations (for example, context reweighting is often justified by information-theoretic considerations, dimensionality reduction optimizes the amount of preserved variance, etc.). Occasionally, some kind of indirect supervision is used: Several parameter settings are tried, and the best setting is chosen based on performance on a semantic task that has been selected for tuning.

The last few years have seen the development of a new generation of DSMs that frame the vector estimation problem directly as a supervised task, where the weights in a word vector are set to maximize the probability of the contexts in which the word is observed in the corpus (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2013a; Turian et al., 2010). The traditional construction of context vectors is turned on its head: Instead of first collecting context vectors and then reweighting these vectors based on various criteria, the vector weights are directly set to optimally predict the contexts in which the corresponding

Learn representations from data by predicting parts of the data, instead of counting

Language modeling

- Instead of modeling P(x), why not model P(w|c)?
- Hint: cast language modeling as a learning task

Language modeling

Context	Word
May the Force be with	____

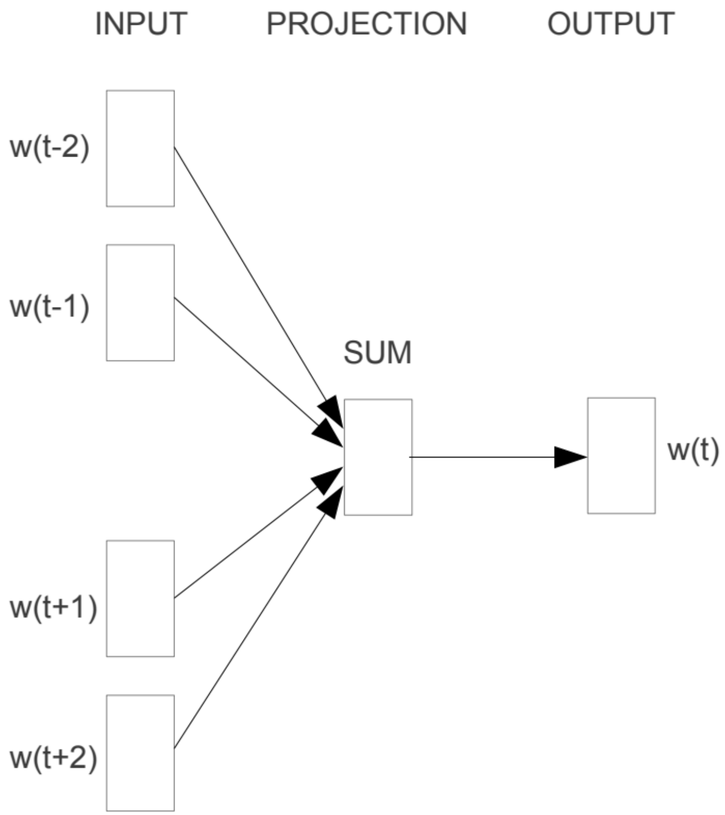
- Reparametrize the probability P(w|c) to depend on dense vectors
- $P(w \mid c) = \frac{\exp(\beta_w v_c)}{\sum_{w'} \exp(\beta_{w'} v_c)}$, where β_w is a vector representation of w and v_c is the vector representation of the context

Word2Vec (CBoW)

- $$P(w \mid c) = \frac{\exp(\beta_w v_c)}{\sum_{w'} \exp(\beta_{w'} v_c)}$$

- In CBoW model of word2vec, w is a word and c are words on the left and right of w

- v_c was calculated as a sum of output vectors



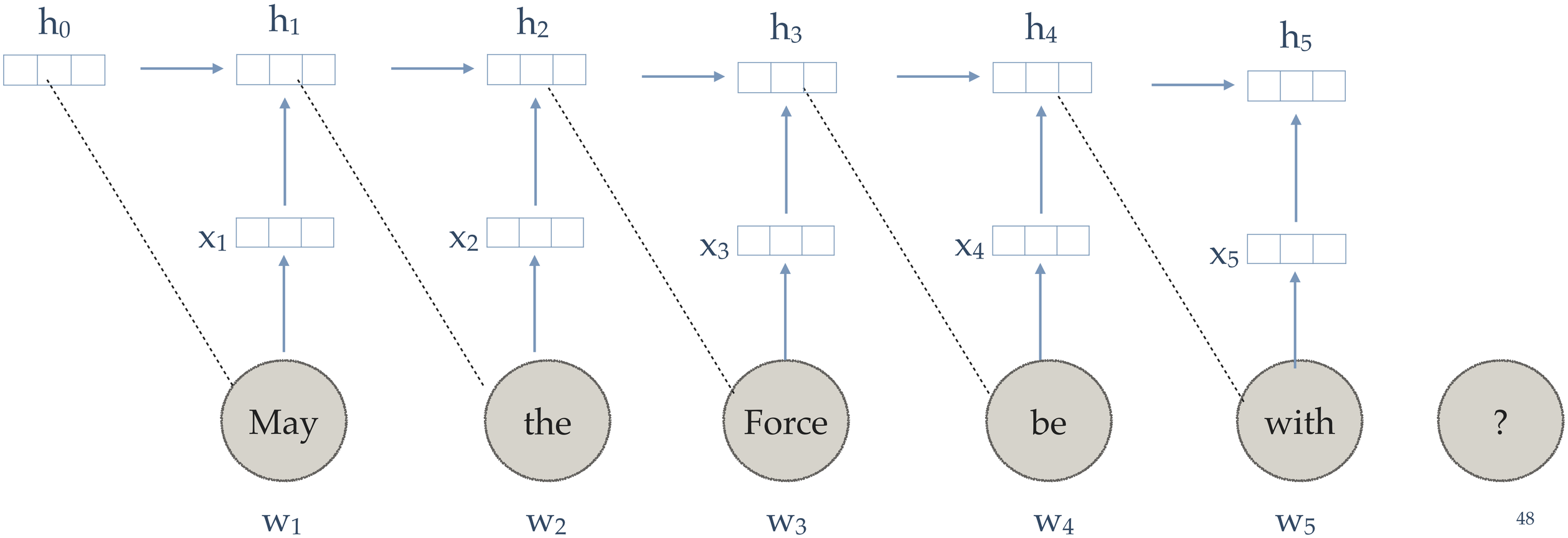
Mikolov et. al. 2013

What can be better ways of coming up with a vector representation of the context?

Context

Word

May the Force be with ____



Recurrent Neural Network LM

At every position m:

$$x_m = \text{Lookup}(\phi, w_m)$$

$$h_m = \text{RNN}(x_m, h_{m-1})$$

$$\boxed{h_m = g(\Theta h_{m-1} + x_m)}$$ Elman unit

$$P(w_{m+1} \mid w_1, w_2, \ldots, w_m) = \text{softmax}(\beta_{w_m}, \mathbf{h}_m)$$

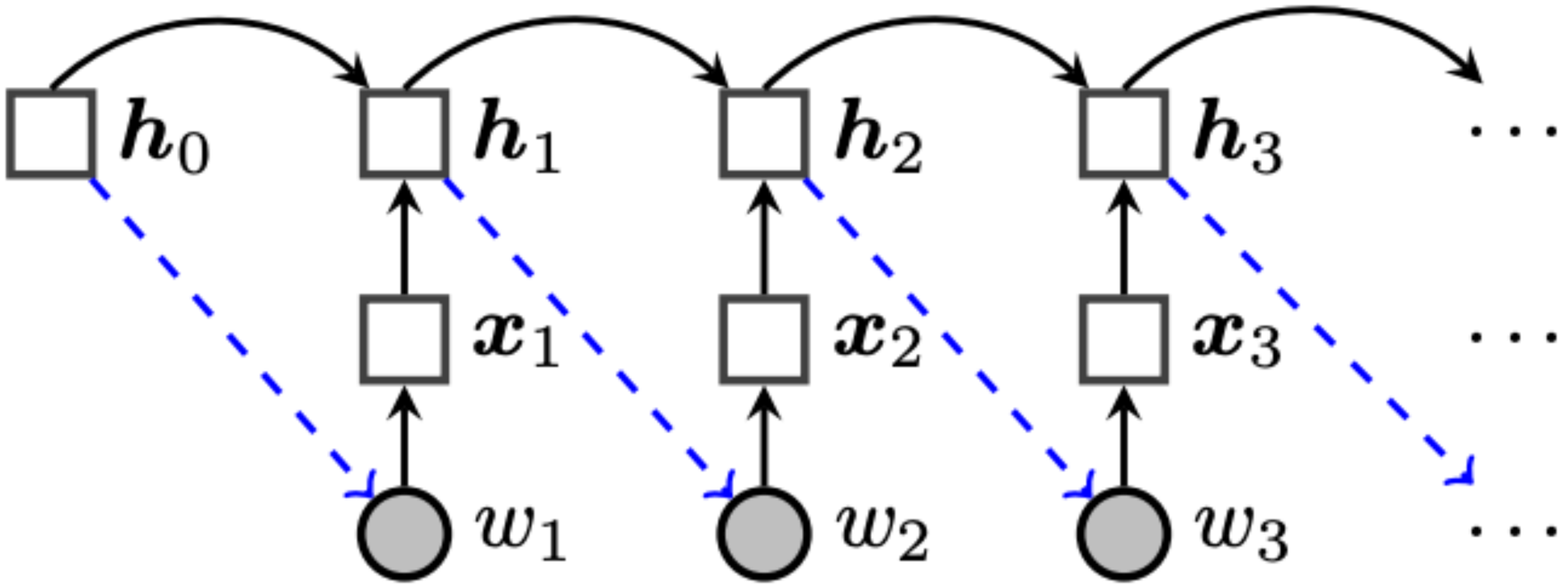
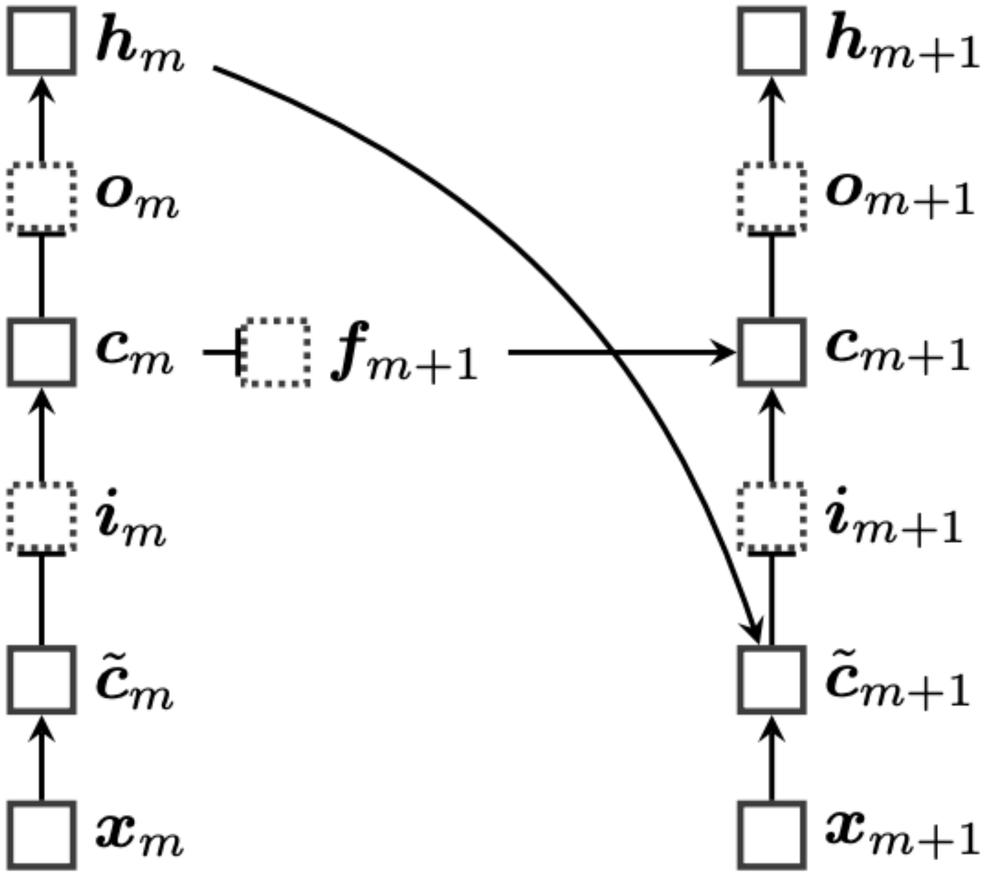


Figure taken from Eisenstein 2018

Long short-term Memories (LSTM)



- Transform x to h by passing x through gating units
- Preserves information propagation over long distances and downweights unimportant contexts in the past

Summary

- Language Modeling is a foundational task in NLP
- Count-based language models are easy to interpret but not very powerful for longer sequences
- Neural LMs (RNNs, LSTMs) are extremely powerful
- Can we improve any further?

In class

- lm exploration