Automatic Speech Recognition (CS753)

Lecture 16: Language Models (Part III)

Instructor: Preethi Jyothi

Mar 16, 2017

Mid-semester feedback ↠ Thanks!

- Work out more examples esp. for topics that are math-intensive

  https://tinyurl.com/cs753problems

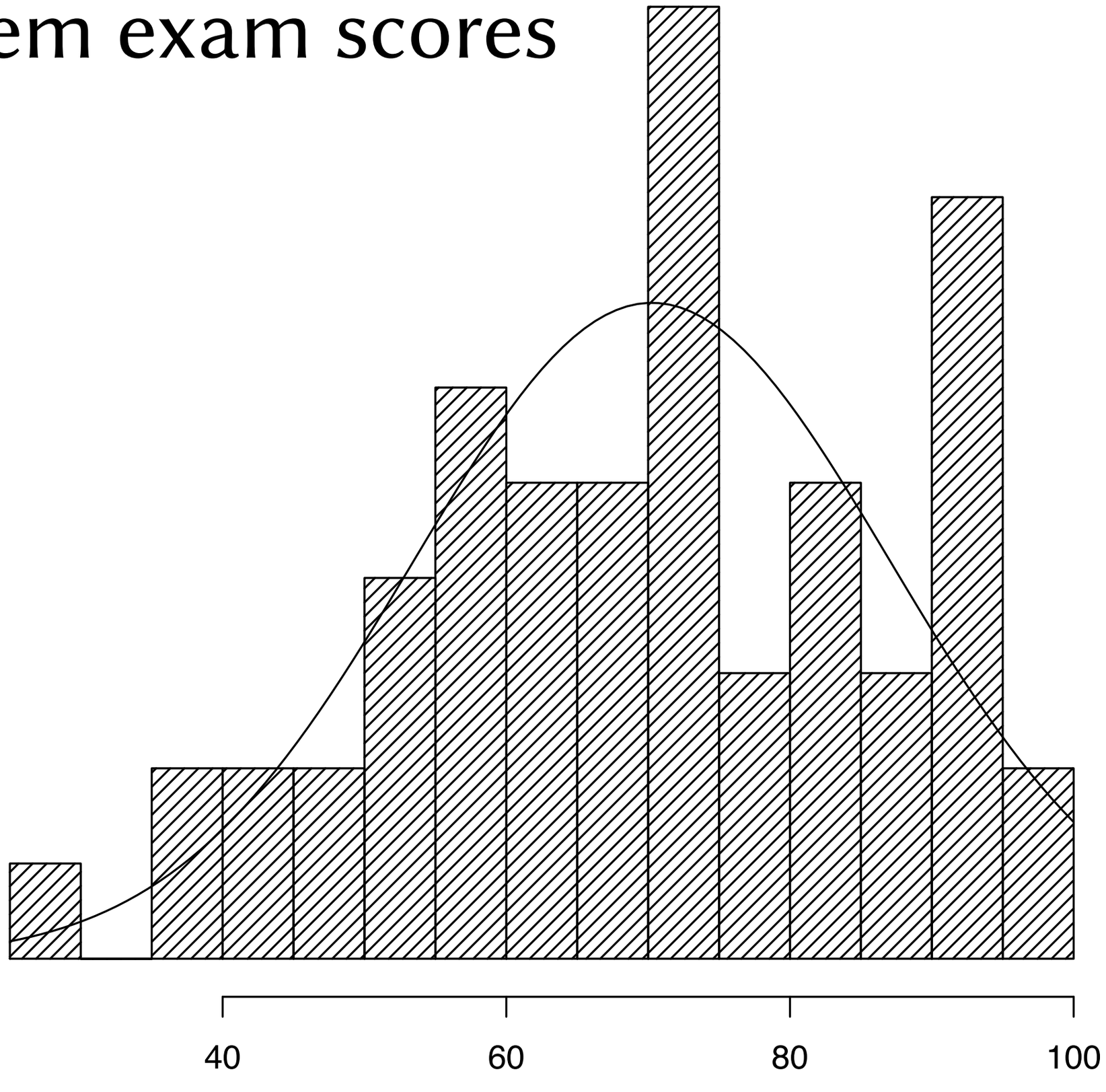
- Give more insights on the “big picture”

  Upcoming lectures will try and address this

- More programming assignments.

  Assignment 2 is entirely programming-based!

Mid-sem exam scores



Recap of Ngram language models

- For a word sequence W = $w_1,w_2,\ldots,w_{n-1},w_n$, an Ngram model predicts w_i based on $w_{i-(N-1)},\ldots,w_{i-1}$

- Practically impossible to see most Ngrams during training

- This is addressed using smoothing techniques involving interpolation and backoff models

Looking beyond words

- Many unseen word Ngrams during training

  This guava is yellow

  This *dragonfruit* is yellow [*dragonfruit* $\longrightarrow$ unseen]

- What if we move from word Ngrams to "CLASS" Ngrams?

$$\Pr(\textsc{Color}|\textsc{Fruit},\textsc{Verb}) = \frac{\pi(\textsc{Fruit},\textsc{Verb},\textsc{Color})}{\pi(\textsc{Fruit},\textsc{Verb})}$$

- (Many-to-one) function mapping each word w to one C classes

Computing word probabilities from class probabilities

- $\Pr(w_i \mid w_{i-1}, \ldots, w_{i-n+1}) \cong \Pr(w_i \mid c(w_i)) \times \Pr(c(w_i) \mid c(w_{i-1}), \ldots, c(w_{i-n+1}))$

- We want Pr(Red|Apple,is)

  $= \Pr(\textsc{color} \mid \textsc{fruit, verb}) \times \Pr(\text{Red} \mid \textsc{color})$

- How are words assigned to classes? Unsupervised clustering algorithm that groups "related words" into the same class [Brown92]

- Using classes, reduction in number of parameters:
  $V^N \longrightarrow VC + C^N$

- Both class-based and word-based LMs could be interpolated

Interpolate many models vs build one model

- Instead of interpolating different language models, can we come up with a single model that combines different information sources about a word?

- Maximum-entropy language models [R94]

[R94] Rosenfeld, “A Maximum Entropy Approach to SLM”, CSL 96

Maximum Entropy LMs

Probability of a word w given history h has a log-linear form:

$$P_\Lambda(w|h) = \frac{1}{Z_\Lambda(h)} \exp\left(\sum_i \lambda_i \cdot f_i(w,h)\right)$$

where

$$Z_\Lambda(h) = \sum_{w' \in V} \exp\left(\sum_i \lambda_i \cdot f_i(w',h)\right)$$

Each $f_i(w, h)$ is a feature function. E.g.

$$f_i(w,h) = \begin{cases} 1 & \text{if } w = a \text{ and } h \text{ ends in } b \\ 0 & \text{otherwise} \end{cases}$$

λ's are learned by fitting the training sentences using a maximum likelihood criterion

Word representations in Ngram models

- In standard Ngram models, words are represented in the discrete space involving the vocabulary

- Limits the possibility of truly interpolating probabilities of unseen Ngrams

- Can we build a representation for words in the continuous space?

Word representations

- 1-hot representation:
    - Each word is given an index in {1, … , V}. The 1-hot vector $f_i \in R^V$ contains zeros everywhere except for the i^{th} dimension being 1
- 1-hot form, however, doesn't encode information about word similarity
- Distributed (or continuous) representation: Each word is associated with a dense vector. E.g.
dog ⟶ {-0.02, -0.37, 0.26, 0.25, -0.11, 0.34}

Word embeddings

- These distributed representations in a continuous space are also referred to as “word embeddings”
  - Low dimensional
  - Similar words will have similar vectors
- Word embeddings capture semantic properties (such as *man* is to *woman* as *boy* is to *girl*, etc.) and morphological properties (*glad* is similar to *gladly*, etc.)

Word embeddings

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

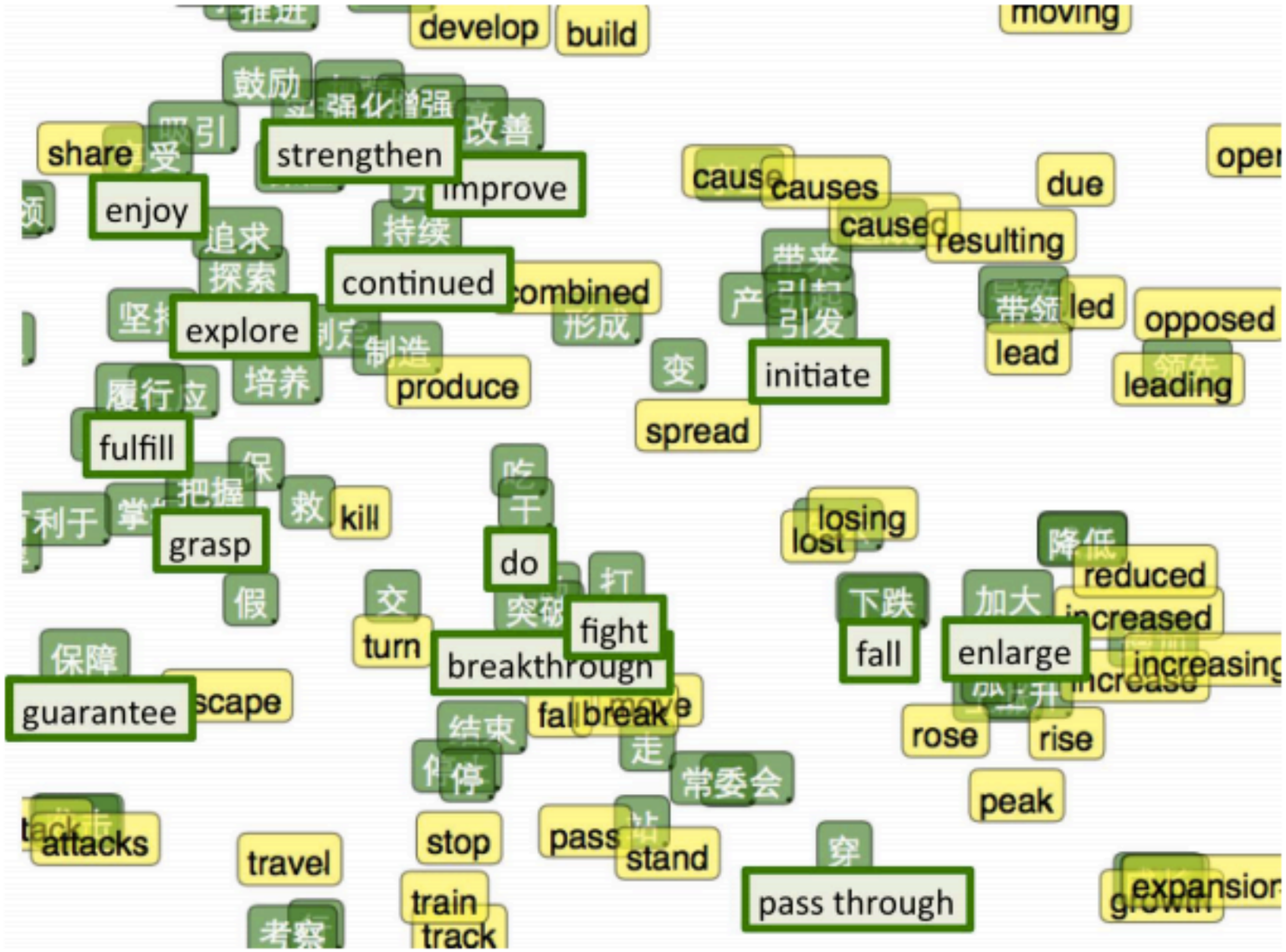
[C01]: Collobert et al.,01

Relationships learned from embeddings

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[M13]: Mikolov et al.,13

Bilingual embeddings



[S13]: Socher et al.,13

Word embeddings

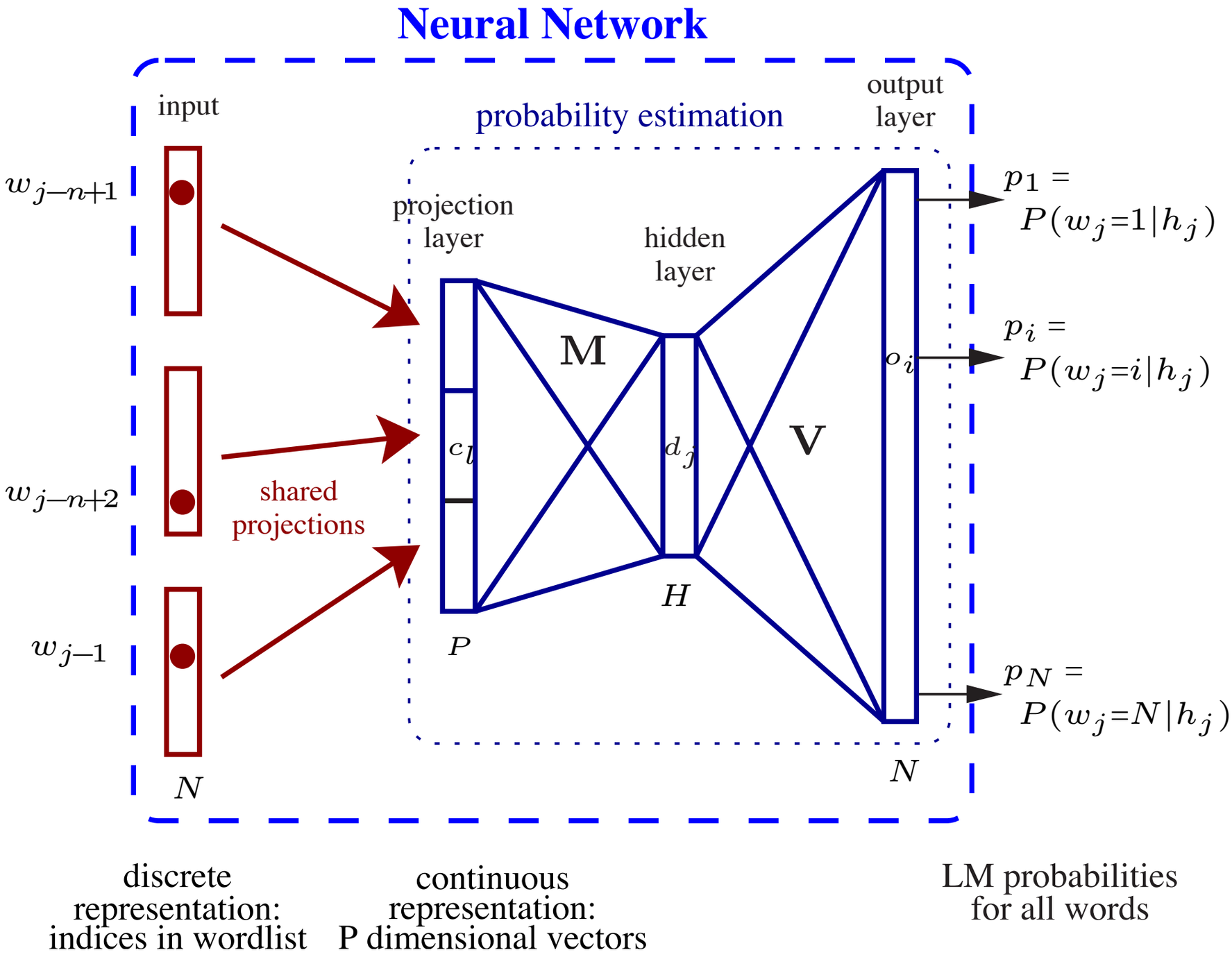
- These distributed representations in a continuous space are also referred to as “word embeddings”
  - Low dimensional
  - Similar words will have similar vectors

- Word embeddings capture semantic properties (such as *man* is to *woman* as *boy* is to *girl*, etc.) and morphological properties (*glad* is similar to *gladly*, etc.)

- The word embeddings could be learned via the first layer of a neural network [B03].

[B03]: Bengio et al., “A neural probabilistic LM”, JMLR, 03

Continuous space language models

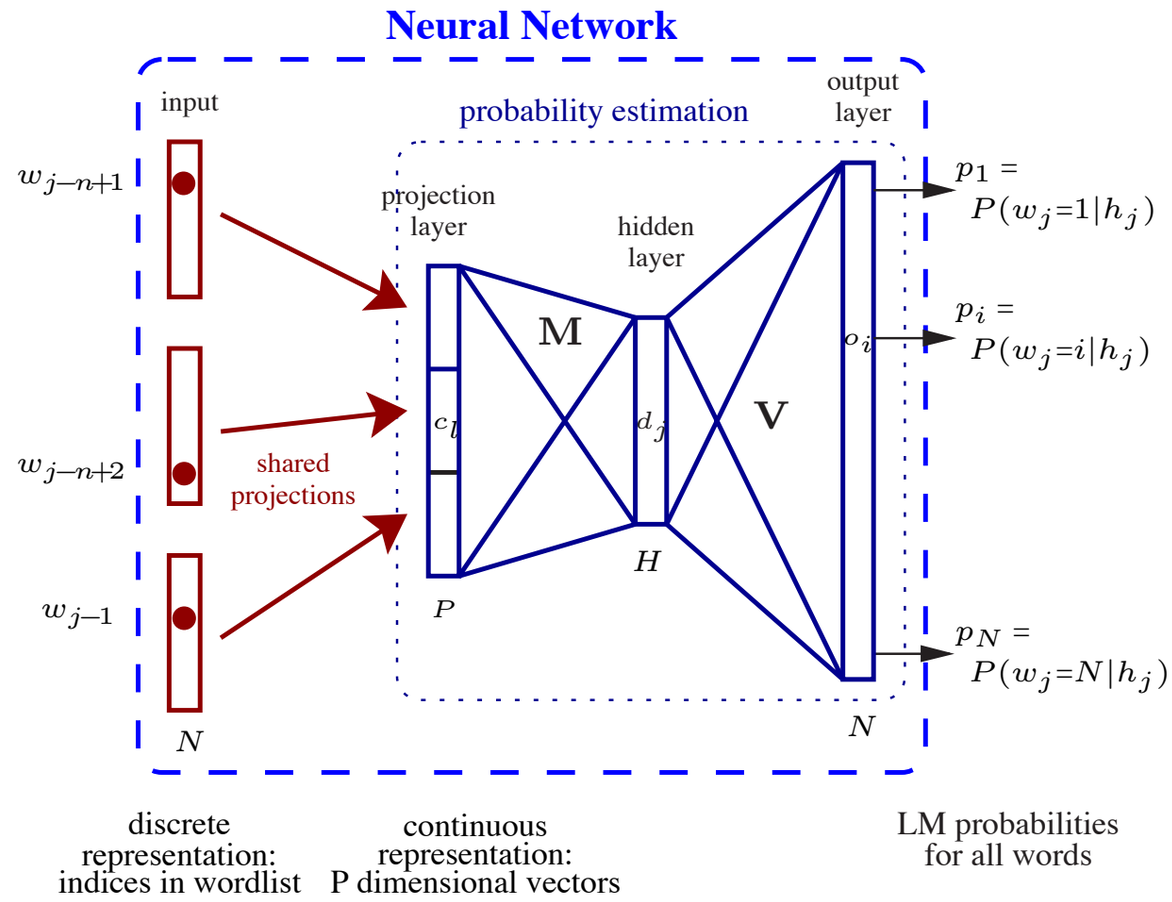


[S06]: Schwenk et al., “Continuous space language models for SMT”, ACL, 06

NN language model

- Project all the words of the context $h_j = w_{j-n+1},\ldots,w_{j-1}$ to their dense forms

- Then, calculate the language model probability $Pr(w_j = i | h_j)$ for the given context h_j



NN language model

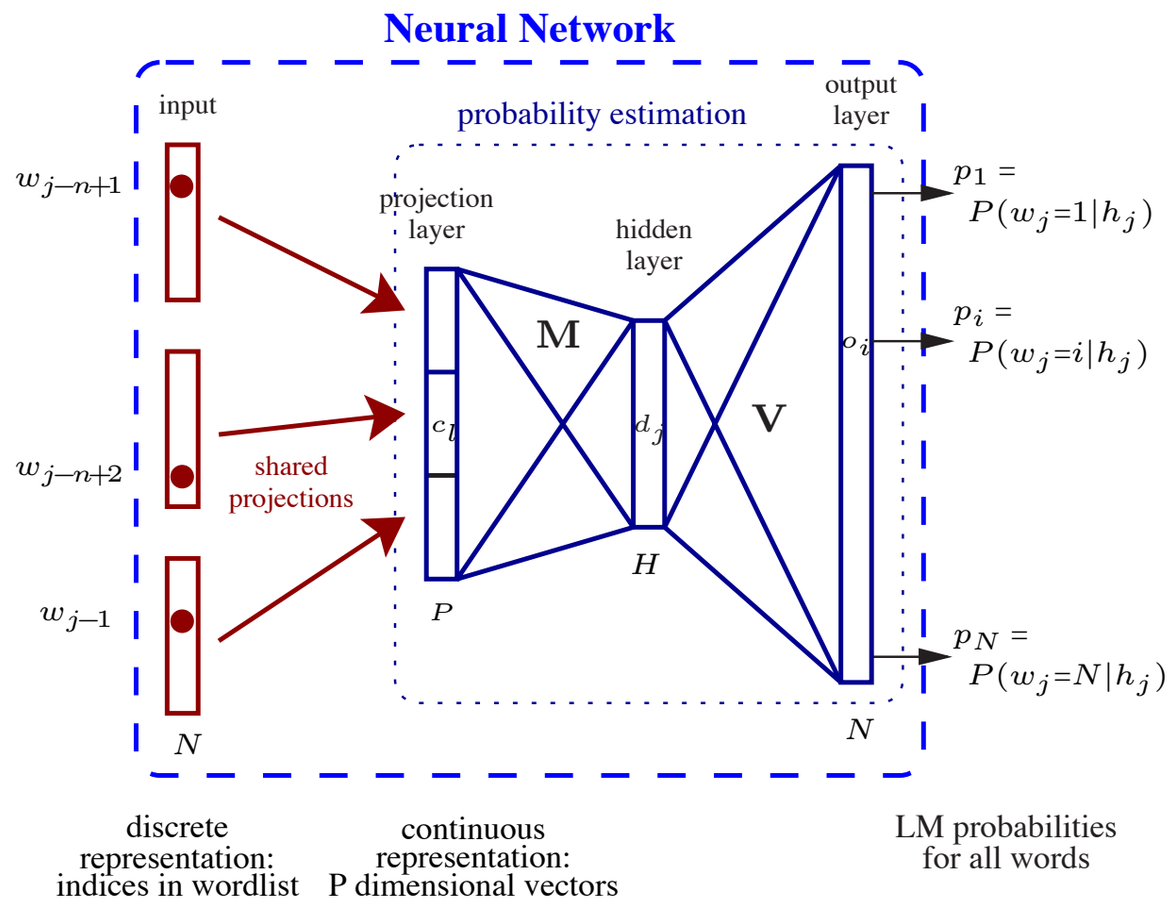
- Dense vectors of all the words in context are concatenated forming the first hidden layer of the neural network
- Second hidden layer:

$$d_k = \tanh(\sum m_{kj} c_j + b_k) \; \forall k = 1, \ldots, H$$

- Output layer:

$$o_i = \sum v_{ik} d_k + \tilde{b}_i \; \forall i = 1, \ldots, N$$

- $p_i \longrightarrow$ softmax output from the ith neuron $\longrightarrow Pr(w_j = i | h_j)$



NN language model

- Model is trained to minimise the following loss function:

$$L = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{kl} m_{kl}^2 + \sum_{ik} v_{ik}^2 \right)$$

- Here, t_i is the target output 1-hot vector (1 for next word in the training instance, 0 elsewhere)

- First part: Cross-entropy between the target distribution and the distribution estimated by the NN

- Second part: Regularization term

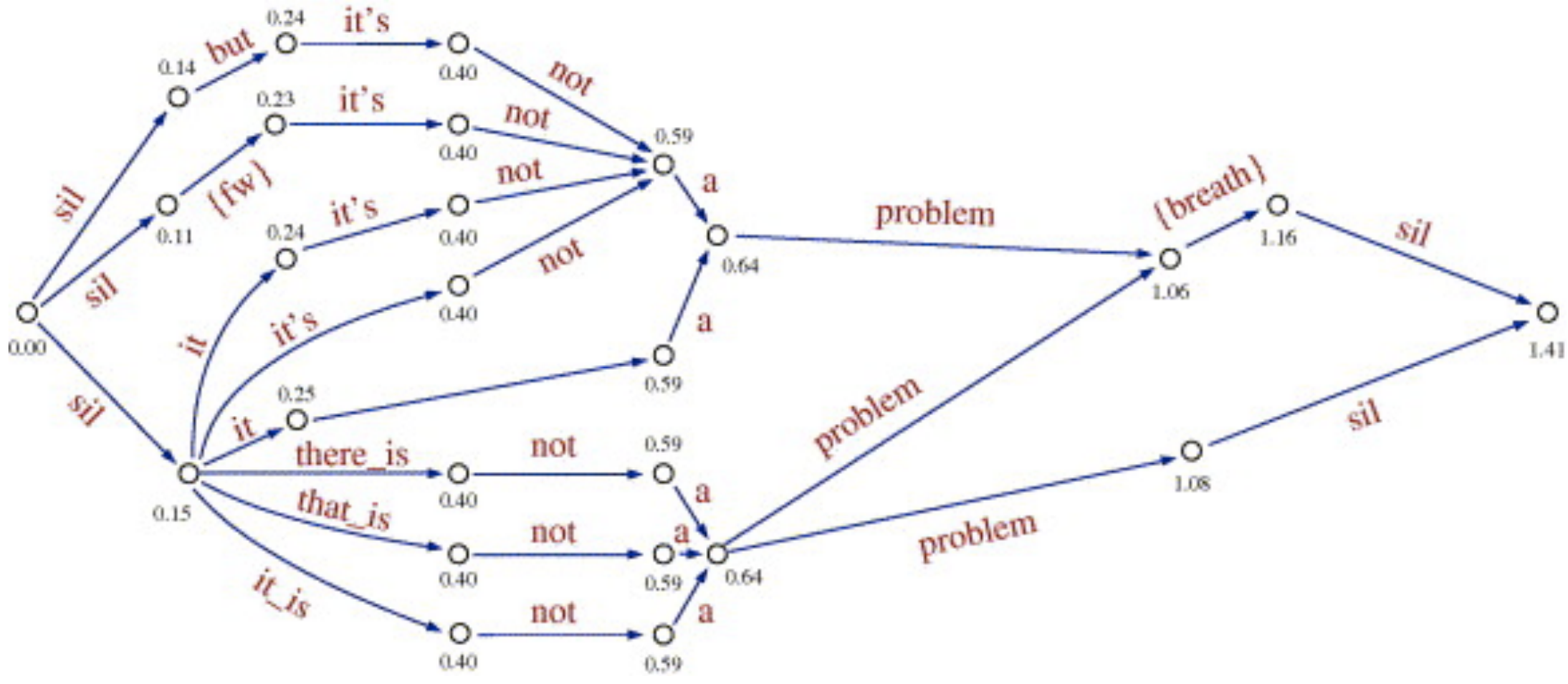
Decoding with NN LMs

- Two main techniques used to make the NN LM tractable for large vocabulary ASR systems:

1. Lattice rescoring

2. Shortlists

Use NN language model via lattice rescoring



- Lattice — Graph of possible word sequences from the ASR system using an Ngram backoff LM
- Each lattice arc has both acoustic/language model scores.
- LM scores on the arcs are replaced by scores from the NN LM

Decoding with NN LMs

- Two main techniques used to make the NN LM tractable for large vocabulary ASR systems:

1. Lattice rescoring

2. Shortlists

Shortlist

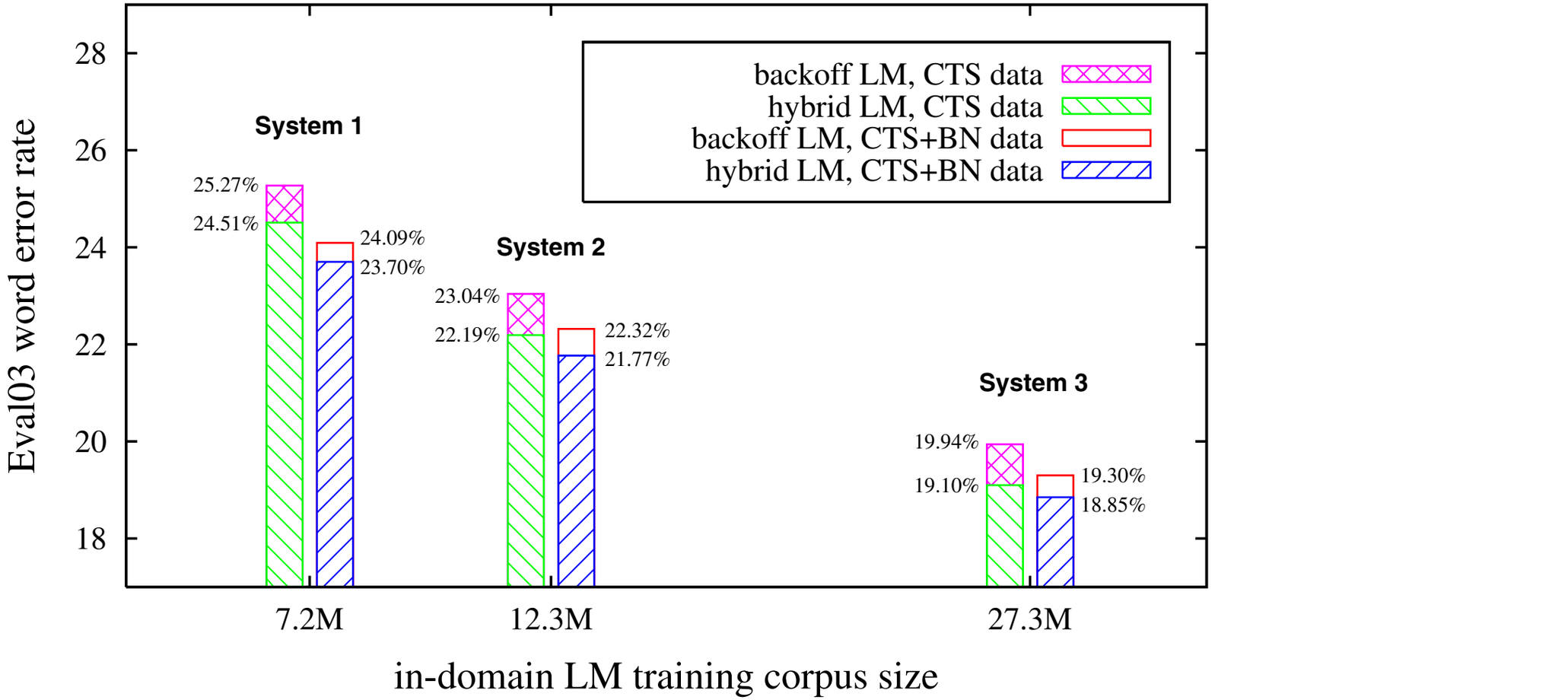
- Softmax normalization of the output layer is an expensive operation esp. for large vocabularies

- Solution: Limit the output to the *s* most frequent words.
  - LM probabilities of words in the short-list are calculated by the NN
  - LM probabilities of the remaining words are from Ngram backoff models

Results

Table 3
Perplexities on the 2003 evaluation data for the back-off and the hybrid LM as a function of the size of the CTS training data

CTS corpus (words)	7.2 M	12.3 M	27.3 M
In-domain data only			
Back-off LM	62.4	55.9	50.1
Hybrid LM	57.0	50.6	**45.5**
Interpolated with all data			
Back-off LM	53.0	51.1	**47.5**
Hybrid LM	50.8	48.0	44.2



[S06]: Schwenk et al., “Continuous space language models for SMT”, ACL, 06

Longer word context?

- What have we seen so far: A feedforward NN used to compute an Ngram probability $Pr(w_j = i|h_j)$ (where h_j is the Ngram history)

- We know Ngrams are limiting: Alice who had attempted *the assignment asked* the lecturer

- How can we predict the next word based on the entire sequence of preceding words? Use recurrent neural networks.

- Next class!