Bilingual Word Embeddings and Recurrent Neural Networks

Fabienne Braune[1]

[1]LMU Munich

June 28, 2017

Outline

1. Softmax Output Units
2. Word Embeddings
3. Bilingual Word Embeddings
4. Recurrent Neural Networks
5. Recap

Softmax Output Units

Backpropagation

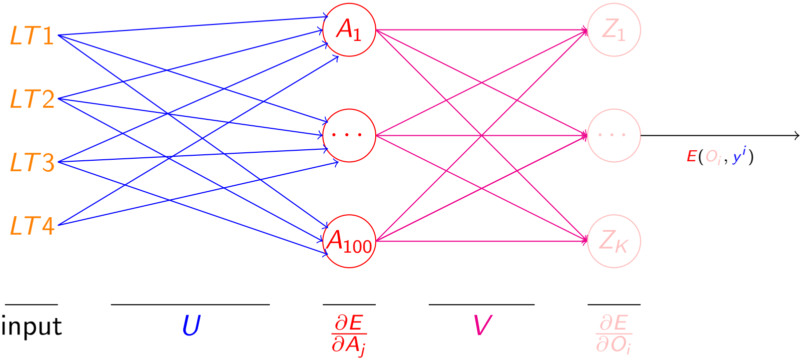
Goal of training: adjust weights such that correct label is predicted

$\rightarrow$ Error between correct label and prediction is minimal

Sketch:

- Compute **derivatives** of Error w.r.t prediction
- Compute **derivatives** in each hidden layer from layer above
  - Backpropagate the error derivative with respect to the output of a unit
- Use **derivatives** w.r.t the activations to get error **derivatives** w.r.t incomming weights

Backpropagation



Backpropagation:

→ Compute E

→ Compute $\frac{\partial E}{\partial O_i}$

Backpropagation 1

Compute error at output E:

Compare output unit with y^i

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - O_i)^2 \text{ (mean squared)}$$

Compute $\frac{\partial E}{\partial O_i}$:

$$\frac{\partial E}{\partial O_i} = -(y_i - O_i)$$

Backpropagation 2

Compute **derivatives** in each hidden layer from layer above:

Compute derivative of error w.r.t logit

$\frac{\partial E}{\partial Z_i} = \frac{\partial E}{\partial O_i} \frac{\partial O_i}{\partial Z_i} = \frac{\partial E}{\partial O_i} O_i(1 - O_i)$ (Note: $O_i = \frac{1}{1+e^{-Z_i}}$)

Compute derivative of error w.r.t previous hidden unit

$\frac{\partial E}{\partial A_j} = \sum_i \frac{\partial Z_i}{\partial A_j} \frac{\partial E}{\partial Z_i} = \sum_i w_{ji} \frac{\partial E}{\partial Z_i}$

Compute derivative w.r.t. weights

$\frac{\partial E}{\partial w_{ji}} = \frac{\partial Z_i}{\partial w_{ji}} \frac{\partial E}{\partial Z_i} = O_i \frac{\partial E}{\partial Z_i}$

$\rightarrow$ Use **recursion** to do this for every layer

Problems with least squares

1. Poor gradient although **big error**

Suppose $Y_i = 1$ and $O_i = 0.00000001 \rightarrow$ Very wrong

Least squares:

- $E = \frac{1}{2}\sum_{i=1}^{n}(1 - 0.00000001)^2$ (mean squared)
  $\rightarrow \frac{\partial E}{\partial O_i} = -(1 - 0.00000001)$
  $\rightarrow \frac{\partial E}{\partial Z_i} = \frac{\partial E}{\partial O_i} * 0.00000001(1 - 0.00000001)$

Suppose $Y_i = 0$ and $O_i = 0.00000001 \rightarrow$ Quite right

Suppose $Y_i = 0$ and $O_i = 0 \rightarrow$ right

Suppose $Y_i = 1$ and $O_i = 1 \rightarrow$ right

Problems with least squares

1. Poor gradient although **big error**

2. Mutually exclusive classes

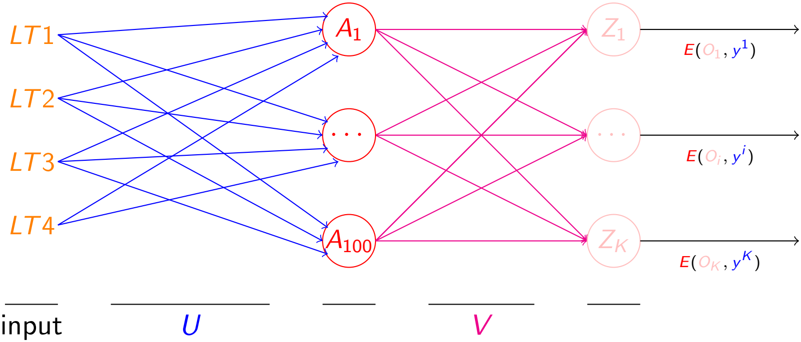
$\rightarrow$ Probabilities should sum up to 1

$\rightarrow$ Give the network this information

Softmax Unit

Softmax unit:

- applied on output logits
- $O_i = \frac{e^{z_i}}{\sum_{j \in K} e^{z_j}}$



Cross Entropy

Cross Entropy:

$$C = -\sum_j y_j log(O_j)$$

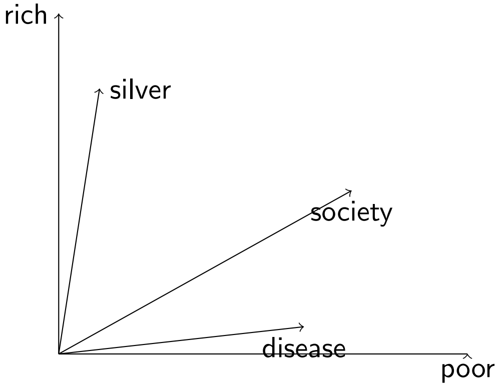
$$\rightarrow \frac{\partial C}{\partial Z_i} = \sum_j \frac{\partial C}{\partial O_j} \frac{\partial O_j}{\partial Z_i} = O_i - y_i$$

- Very big gradient when target is 1 and output near 0
- Mutually exclusive classes taken into account

Word Embeddings

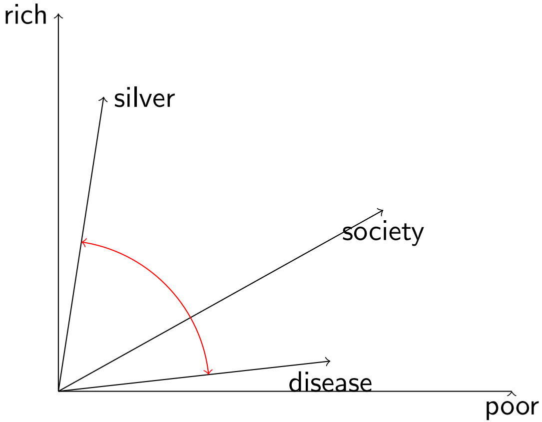
Word Embeddings

- Representation of words in vector space



Word Embeddings

- Similar words are close to each other
  → Similarity is the cosine of the angle between two word vectors



Learning word embeddings

Count-based methods:

- Compute cooccurrence statistics
- Learn high-dimensional representation
- Map sparse high-dimensional vectors to small dense representation

Neural networks:

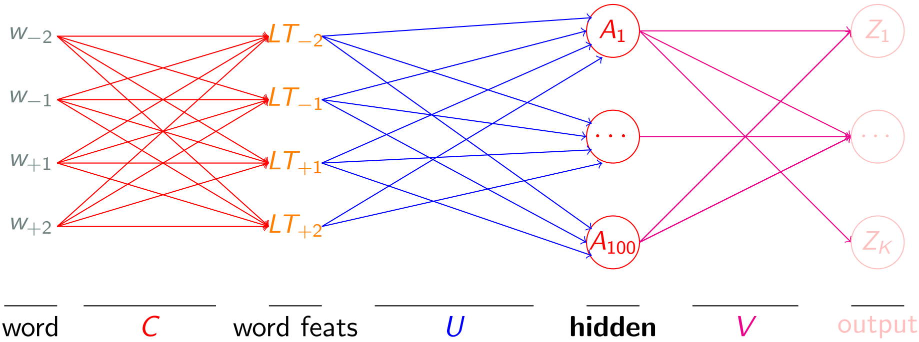
- Predict a word from its neighbors
- Learn (small) embedding vectors

Word2Vec

- Software train word embeddings (Mikolov. 2013)
  $\rightarrow$ very fast

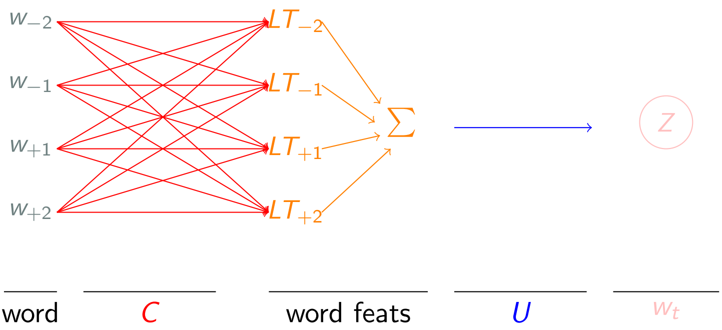
- Two models:
  - BOW model:
    - Input is is w_{t+2}, w_{t+1}, w_{t-1} and w_{t-2}
    - Prediction is w_t
  - Skip-gram model:
    - Input is w_t
    - Prediction is w_{t+2}, w_{t+1}, w_{t-1} and w_{t-2}

Feedforward Neural Network with Lookup Table



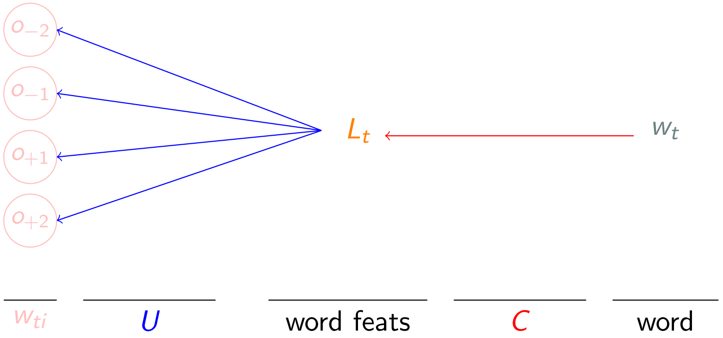
Note: Bias terms omitted for simplicity

Learning word embeddings with CBOW



Note: Bias terms omitted for simplicity

Learning word embeddings with skip-gram



Note: Bias terms omitted for simplicity

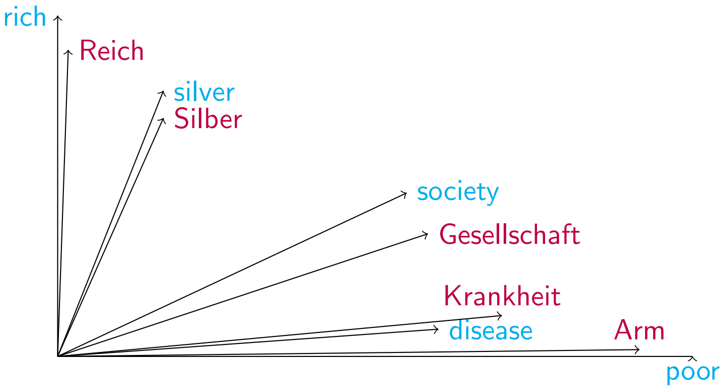
Bilingual Word Embeddings

Bilingual Word Spaces

Representation of words in two languages in same semantic space:

→ Each word is one dimension

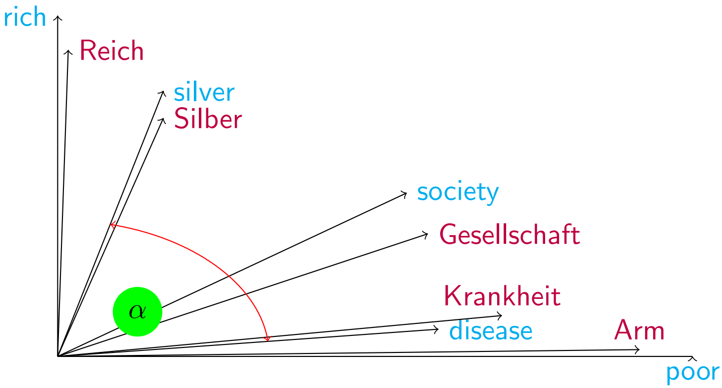
→ Each word represented respective to all others



Bilingual Word Spaces

Representation of words in two languages in same semantic space:

- $\rightarrow$ Similar words are close to each other
- $\rightarrow$ Given by cosine



Exercise

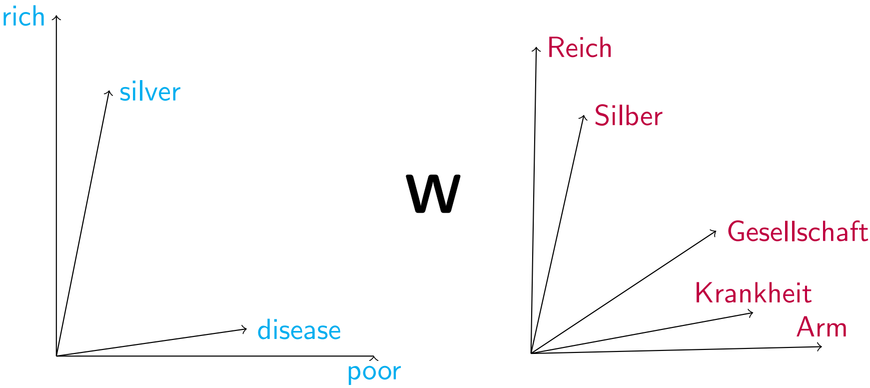
How is this related to translation?

Learning Bilingual Word Embeddings

- Learn monolingual word embeddings and map using seed lexicon
  Mikolov et al. (2013); Faruqui and Dyer (2014); Lazaridou et al. (2015)
  Need seed lexicon
- Learn bilingual embeddings or lexicon from document-aligned data
  Vulic and Moens (2015); Vulic and Korhonen (2016)
  Need document-aligned data
- Learn bilingual embeddings from parallel data
  Hermann and Blunsom (2014), Gouws et al. (2015), Gouws and Søgaard (2015), Duong et al. (2016)
  Need for parallel data

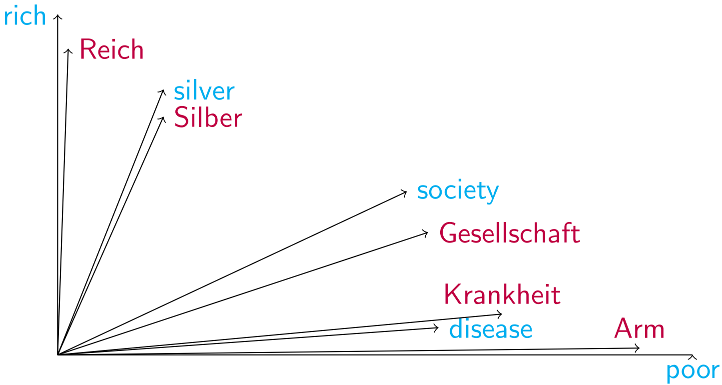
Post-hoc mapping (with seed lexicon)

- Learn monolingual word embeddings
- Learn a linear mapping W



Post-hoc mapping

- Project source words into target space



Post-hoc Mapping with seed lexicon

1. Train monolingual word embeddings (Word2vec) in English
   - Need English monolingual data
2. Train monolingual word embeddings (Word2vec) in German
   - Need German monolingual data
3. Learn mapping **W** using a seed lexicon
   - Need a list of 5000 English words and their translation

Learning W with Ridge Regression

Ridge regression (Mikolov et al. (2013))

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{\mathbf{i}}^{\mathbf{n}} \| \mathbf{x_i} \mathbf{W} - \mathbf{y_i} \|^{\mathbf{2}}$$

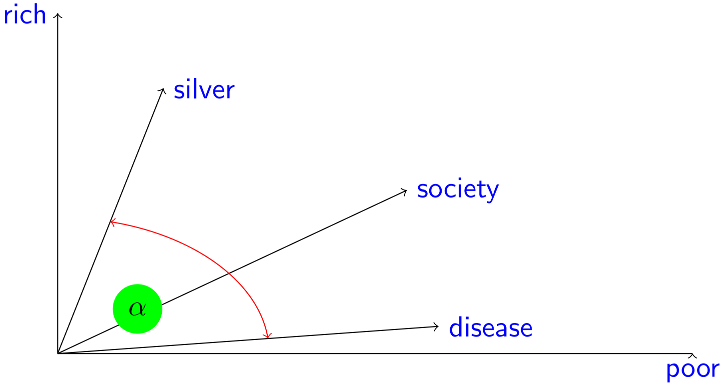
x_i : **embedding** of i-th source (English) word in the seed lexicon.

y_i : **embedding** of i-th target (German) word in the seed lexicon.

Learning W with Ridge Regression

x_i : **embedding** of i-th source (English) word in the seed lexicon.

$\rightarrow$ **vector** representing disease in **monolingual** word embedding



Learning W with Ridge Regression

Ridge regression (Mikolov et al. (2013))

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{\mathbf{i}}^{\mathbf{n}} \| \mathbf{x_i} \mathbf{W} - \mathbf{y_i} \|^{\mathbf{2}}$$

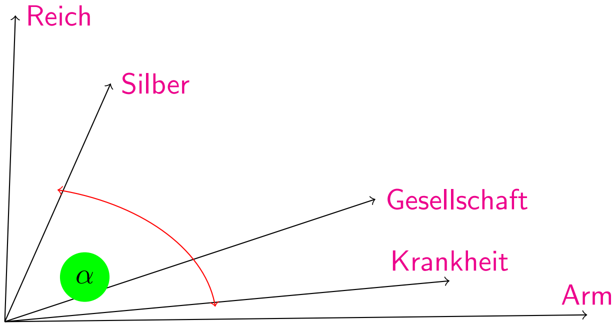
x_i : **embedding** of i-th source (English) word in the seed lexicon.

y_i : **embedding** of i-th target (German) word in the seed lexicon.

Learning W with Ridge Regression

y_i : **embedding** of i-th target (German) word in the seed lexicon.

$\rightarrow$ **vector** representing Krankheit in **monolingual** word embedding



Learning W with Ridge Regression

Ridge regression (Mikolov et al. (2013))

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{\mathbf{i}}^{\mathbf{n}} \mid\mid \mathbf{x_i} \cdot \mathbf{W} - \mathbf{y_i} \mid\mid^{\mathbf{2}}$$

- Predict projection $\mathbf{y^*}$ by computing $\mathbf{x_i} \cdot \mathbf{W}$
- Compute squared error between $\mathbf{y^*}$ and $\mathbf{y_i}$
  - Correct translation t_i given in seed lexicon
  - Vector representation $\mathbf{y_i}$ is given by embedding of t_i
- Find $\mathbf{W}$ such that squared error over training set is minimal

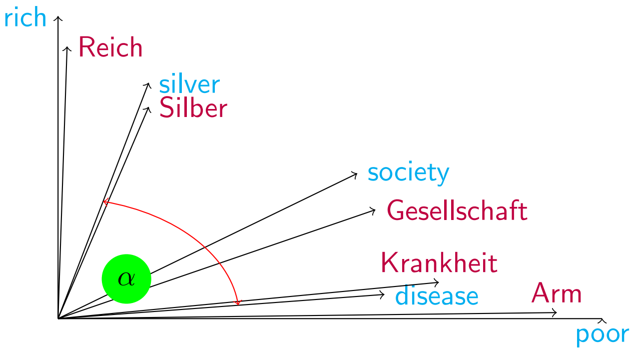
Adding Regularization

If $\mathbf{W}$ is too complex the model overfits the data
$\rightarrow$ Add regularization term that keeps $\mathbf{W}$ small
$\rightarrow$ Add weighted norm of $\mathbf{W}$ to cost function

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{\mathbf{i}}^{\mathbf{n}} \| \mathbf{x_i} \cdot \mathbf{W} - \mathbf{y_i} \|^{\mathbf{2}} + \lambda \| \mathbf{W} \|$$

Bilingual lexicon induction

- Task to evaluate bilingual word embeddings extrinsically
- Given a set of source words, find the corresponding translations:
  - Given silver, find its vector in the BWE
  - Retrieve the German word whose vector is closest (cosine distance)



Bilingual lexicon induction with ridge regression

Data: WMT 2011 training data for English, Spanish, Czech
Seed: 5000 most frequent words translated with Google Translate
Test: 1000 next frequent words translated with Google Translate

→ Removed digits, punctuation and transliterations

Languages	top-1	top-5
En-Es	33 %	51 %
Es-En	35 %	50 %
En-Cz	27 %	47 %
Cz-En	23 %	42 %
+ Es-En	53 %	80 %

→ with spanish google news

Learning W with Max Margin Ranking

Max-margin ranking loss (Lazaridou et al. (2015)):

- Predict projection $\mathbf{y*}$ by computing $\mathbf{x_i} \cdot \mathbf{W}$
- Compute ranking loss between:
  - $\mathbf{y*}$
  - Vector of correct translation $\mathbf{y_i}$
  - Negative samples $\mathbf{y_j}$
- $\sum_{i \neq j}^{k} \max\{0, \gamma + Sdist(\vec{y^*}, \vec{y_i}) - Sdist(\vec{y^*}, \vec{y_j})\}$
  - $Sdist(\vec{x}, \vec{y})$: inverse cosine
  - $\rightarrow$ measures semantic distance between $\vec{y^*}$ and $\vec{y_i}$
  - γ and k tuned on held-out data

Learning W with Max Margin Ranking

Max-margin ranking loss (Lazaridou et al. (2015)):

- $\sum_{i \neq j}^{k} \max\{0, \gamma + Sdist(\vec{y^*}, \vec{y_i}) - Sdist(\vec{y^*}, \vec{y_j})\}$
  - $Sdist(\vec{x}, \vec{y})$: inverse cosine
  - $\rightarrow$ measures semantic distance between $\vec{y^*}$ and $\vec{y_i}$
- For each source (English) vector x_i, distance of $\vec{y^*}$ to correct translation $\vec{y_i}$ should be **smaller** than distance to wrong translation $\vec{y_j}$

Bilingual lexicon induction with max margin ranking

Data: 4 mio sentences from Europarl, News, Common Crawl
Seed: 5000 most frequent words-pairs computed with parallel data
Test: 1000 next words-pairs computed with parallel data

Setup	top-1	top-5
En-De all	18.6 %	27.4 %
En-De	23.1 %	33.61 %

$\rightarrow$ max-margin outperforms ridge

Building bilingual corpora

Idea:

- Create bilingual corpus and build bilingual word embeddings
- Combine monolingual texts to create bilingual data
- Learn word embeddings with skip-gram or CBOW on bilingual data
  - Simply run word2vec on the bilingual data
  - Just need to create bilingual data

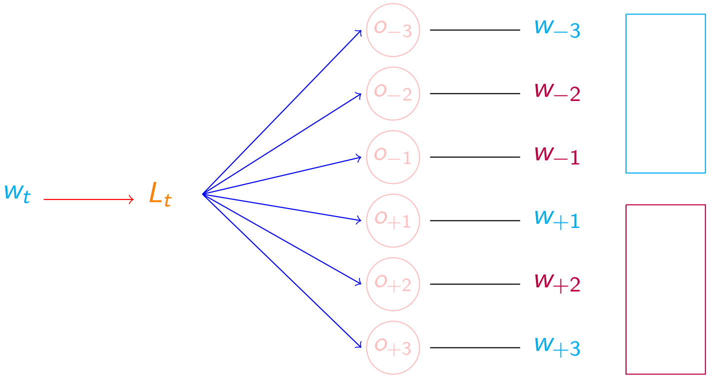
Document Merge and Shuffle

Merge and shuffle document-aligned monolingual data (Vulic and Moens (2015)):

- Document-pairs $P = \{(D_1^S, D_1^T), \ldots, (D_n^S, D_n^T)\}$
- Merge each pair (D_i^S, D_i^T) into pseudo-bilingual document B_i
- Shuffle each B_i
  - Random permutation of words w_j in B_i
  - Assures that each word w_j obtains collocates from both languages
- Train word embeddings (word2vec) on pseudo-bilingual document B_i

Building bilingual corpora

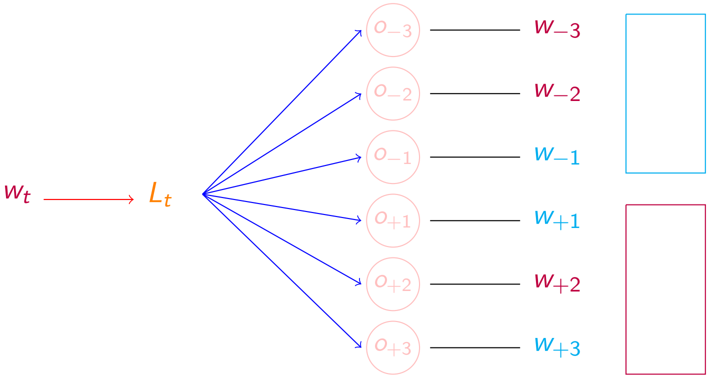
English word with bilingual context



Note: Bias terms omitted for simplicity

Building bilingual corpora

German word with bilingual context



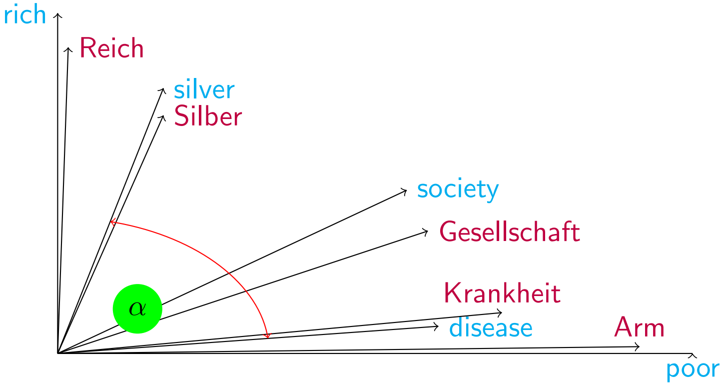
Note: Bias terms omitted for simplicity

Bilingual Word Spaces

Representation of words in two languages in same semantic space:

→ Similar words are close to each other

→ Given by cosine



Merge and Shuffle with seed lexicon

Merge and shuffle monolingual data with seed lexicon (Gouws and Søgaard (2015)):

- Document-pair $P = (D_1^S, D_1^T)$
  - Merge each pair P into pseudo-bilingual document B
  - Shuffle B

- Seed lexicon $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Each y_i is translation of x_i
  - In bilingual document B replace each x_i with y_i with proba 0.5
  - Allows to consider k translations of x_i and draw with proba $\frac{0.5}{k}$

Bilingual lexicon induction

- Task to evaluate bilingual word embeddings extrinsically
- Merge and shuffle document-aligned monolingual data (Vulic and Moens (2015))
- A bit worse than post-hoc mapping with ridge regression
- Merge and shuffle monolingual data with seedLexicon (Gouws and Søgaard (2015))
- Evaluated on cross-lingual POS tagging

Recurrent Neural Networks

Neural language model

- Early application of neural networks (Bengio et al. 2003)
- Task: Given k previous words, predict the current word

  Estimate: $P(w_t | w_{t-k}, \cdots, w_{t-2}, w_{t-1})$

- Previous (non-neural) approaches:

  Problem: Joint distribution of consecutive words difficult to obtain
  $\rightarrow$ chose small history to reduce complexity (n=3)
  $\rightarrow$ predict for unseen history through back-off to smaller history

  Drawbacks:

  Takes into account small and fixed context
  Does not model similarity between words

Neural language model

- Early application of neural networks (Bengio et al. 2003)
- Task: Given k previous words, predict the current word

  Estimate: $P(w_t|w_{t-k}, \cdots, w_{t-2}, w_{t-1})$
- Feedforward NN for LM:

  Does model similarity between words

  Restricted to small and fixed context

Neural language model

Take into account context of any size:

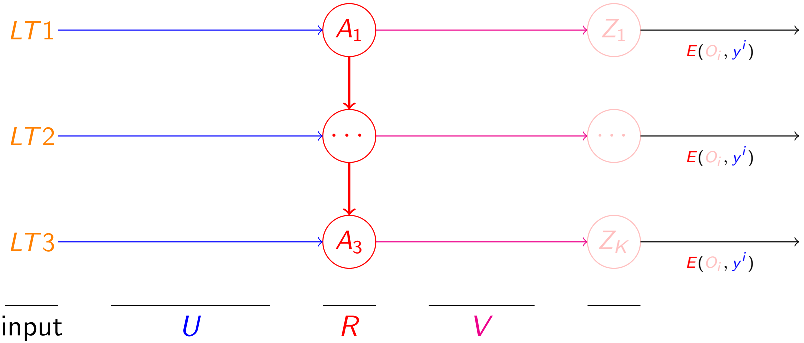
- Need a way to model sequentiality
- Introduce notion of time in neural network

  $\rightarrow$ Recurrent Neural Networks

Recurrent Neural Networks

Connection between hidden states

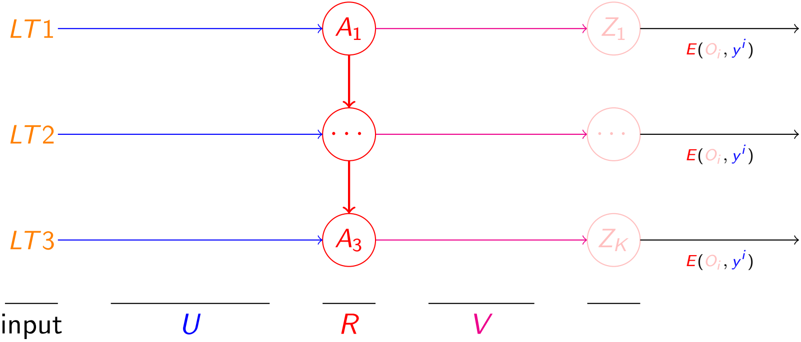
$\rightarrow$ connections between time units, models sequentiality



Recurrent Neural Networks

Input weights U are **shared** among each time step
Output weights V are **shared** among each time step

$\rightarrow$ **Less** parameters as in feedworward NN with many layers

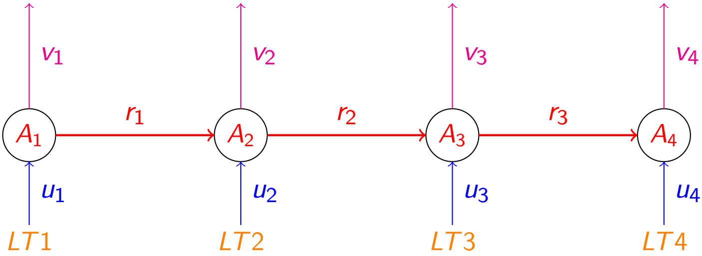


Forward Propagation

Input embeddings passed forward through time

Each hidden unit is one time step

$\rightarrow$ Acts as memory of what happened before



Forward Propagation

Specify initial state $\mathbf{A_0}$:

Input layer (X): Word features LT^t

Weight matrices U, $\mathbf{R}$, V

Time Step (A^t): $\sigma(LT^t \cdot U + A^{t-1} \cdot R + d)$

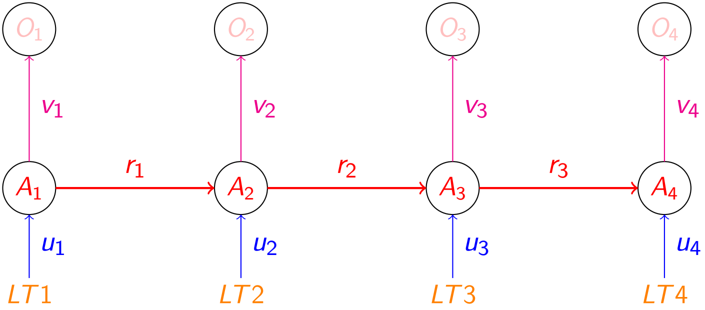
Output layer (0^t): $A^t \cdot V + b$

Prediction: $h^t(X) = softmax(0^t)$

Forward Propagation

Compute prediction for each time step

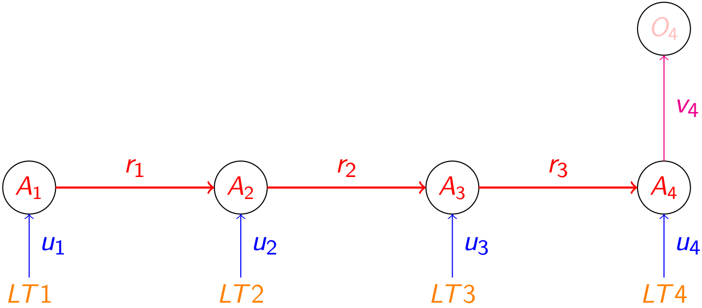
Apply softmax on each output



Forward Propagation

Compute prediction for one time step

Apply softmax on last output $\rightarrow$ Language model architecture



Backpropagation

Goal of training: adjust weights such that correct label is predicted

$\rightarrow$ Error between correct label and prediction is minimal

Sketch:

- Compute **derivative** of Error w.r.t. prediction
- Compute **derivatives** in each hidden layer from layer above
  - Backpropagate the error derivative with respect to the output of a unit
- Use derivatives w.r.t the activities to get error derivatives w.r.t incomming weights

Backpropagation through time

Sketch:

- Compute **derivative** of Error w.r.t. prediction
- Compute **derivatives** from layer above and **previous time step**

  $\rightarrow$ Each time step can be represented by a feedforward neural network

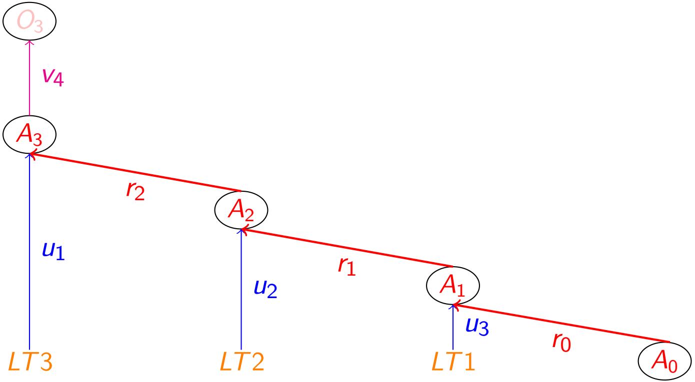
  $\rightarrow$ Shared connections represented by constrained weights (same)

  $\rightarrow$ Sum derivatives over each time step

Backpropagation through time

→ Each time step can be represented by a feedforward neural network

→ Here feedforward neural network for time step 3



Backpropagation through time

Sketch:

- Compute **derivative** of Error w.r.t. prediction
- Compute **derivatives** from layer above and **previous time step**

  $\rightarrow$ Each time step can be represented by a feedforward neural network

  $\rightarrow$ Shared connections represented by constrained weights (same)

  $\rightarrow$ Sum derivatives over each time step

Backpropagation through time

Difficulties:

- Multiply many derivatives together
  $\rightarrow$ Gradients tend to explode or vanish

LSTM handle this

- LSTM for Long Short Term Memory Network
- Improve memory capacity of hidden states
  Will be presented next week!

Recap

- Squared error not good loss function
  - Softmax units with cross-entropy
- Bilingual word embeddings represent words in two languages
- Induction with post-hoc mapping:
  - Train monolingual word embeddings
  - Map with seed lexicon
- Induction with bilingual corpora:
  - Create bilingual corpora
  - Train monolingual word embeddings

Recap

Recurrent neural networks for language modeling:

- Task: Given k previous words, predict the current word

  Estimate: $P(w_t | w_{t-k}, \cdots, w_{t-2}, w_{t-1})$

- Problems with feedforward approach

  $\rightarrow$ chose fixed history to reduce complexity

- Recurrent neural networks as solution
  - Model sequentiality with recurrent units
  - Allow to model history of any size

References I

Duong, L., Kanayama, H., Ma, T., Bird, S., and Cohn, T. (2016). Learning crosslingual word embeddings without bilingual corpora. In *Proc. EMNLP*.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proc. EACL*.

Gouws, S., Bengio, Y., and Corrado, G. (2015). Bilbowa: Fast bilingual distributed representations without word alignments. In *Proc. ICML*.

Gouws, S. and Søgaard, A. (2015). Simple task-specific bilingual word embeddings. In *Proc. NAACL*.

Hermann, K. M. and Blunsom, P. (2014). Multilingual models for compositional distributed semantics. In *Proc. ACL*, pages 58–68, Baltimore, Maryland. Association for Computational Linguistics.

Lazaridou, A., Dinu, G., and Baroni, M. (2015). Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proc. ACL*.

References II

Mikolov, T., Le, Q. V., and Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Vulic, I. and Korhonen, A. (2016). On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. In *Proc. ACL*, pages 247–257.

Vulic, I. and Moens, M. (2015). Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proc. ACL*.

Recurrent Neural Networks

Can be bidirectional