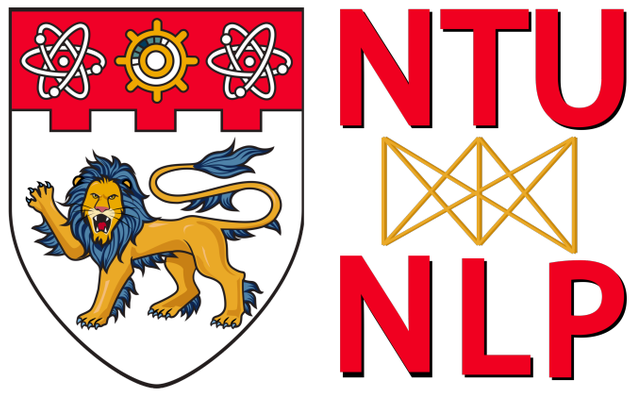
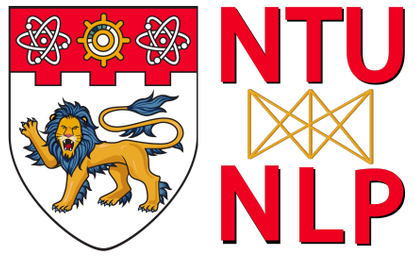
Deep Learning for Natural Language Processing

Shafiq Joty



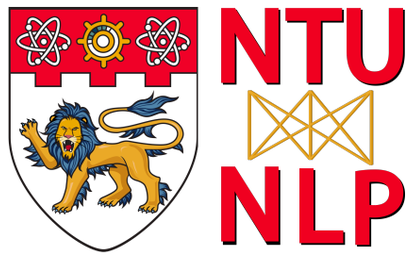
Lecture 4: Word Vectors

Announcements



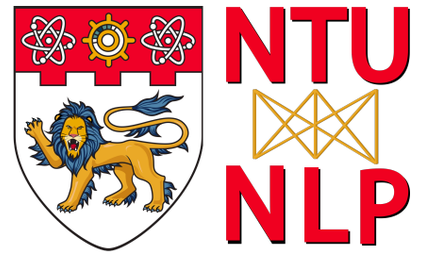
- Assignment 1 is already out
- Project proposal instruction is out
- Project proposal is due just before recess
- There will be a short presentation

Lecture Plan



- Recap
  - Word meaning & Distributed representations
  - LM for word vector
- Word2Vec models & Negative sampling
- Incorporating subword information (FastText)
- Glove
- Word vector evaluation
- Cross-lingual word embeddings

Where we are



Models/Algorithms

- Linear models
- Feed-forward Neural Nets

NLP tasks/applications

- **Word** meaning

Word Meaning

NTU NLP

- The idea that is represented by a word, phrase, etc.
- How do we represent it in a computer?

Denotational Semantics

Common solution: Use e.g. WordNet, a thesaurus containing lists of **synonym sets** and **hypernyms** ("is a" relationships).

e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

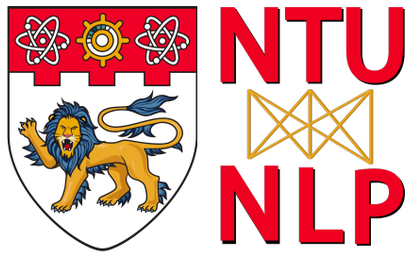
```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g. hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

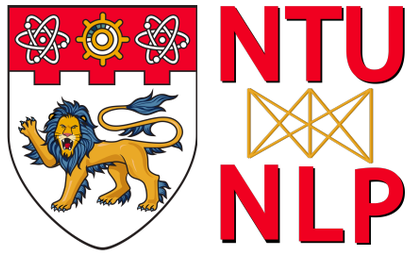
Problems with WordNet



- Great as a lexical resource but missing nuance
  - e.g. "proficient" is listed as a synonym for "good". This is only correct in some contexts.
- Missing new meanings of words, e.g., wicked, badass, nifty, wizard, genius, ninja, bombest
- Impossible to keep up-to-date!
- Subjective
- Requires human labor to create and adapt

Source: Stanford 224N

Problems with Discrete Representation



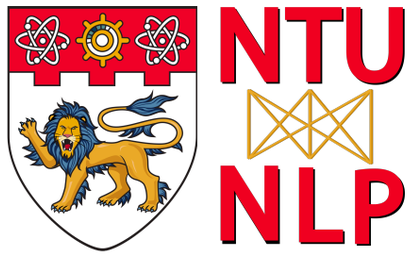
- Hard to compute accurate word similarity
- In traditional NLP, we regard words as discrete symbols: hotel, conference, motel

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

These two vectors are orthogonal.

- Could try to rely on WordNet's list of synonyms to get similarity?
  - But it is well-known to fail badly: incompleteness, etc.
  - How to represent polysemous words?

Representing words by their context



- Instead: learn to encode similarity in the vectors themselves

- Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**
    - "You shall know a word by the company it keeps"

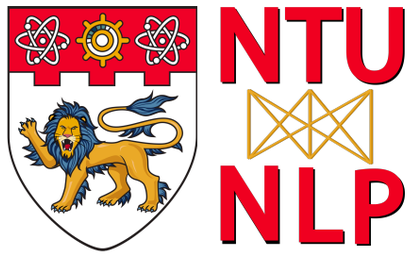
(J. R. Firth 1957: 11)

- When a word w appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).

- Use the many contexts of w to build up a representation of w

Source: Stanford 224N

Representing words by their context



- When a word w appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).

- Use the many contexts of w to build up a representation of w

…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…*

These context words will represent ***banking***

Word Vectors

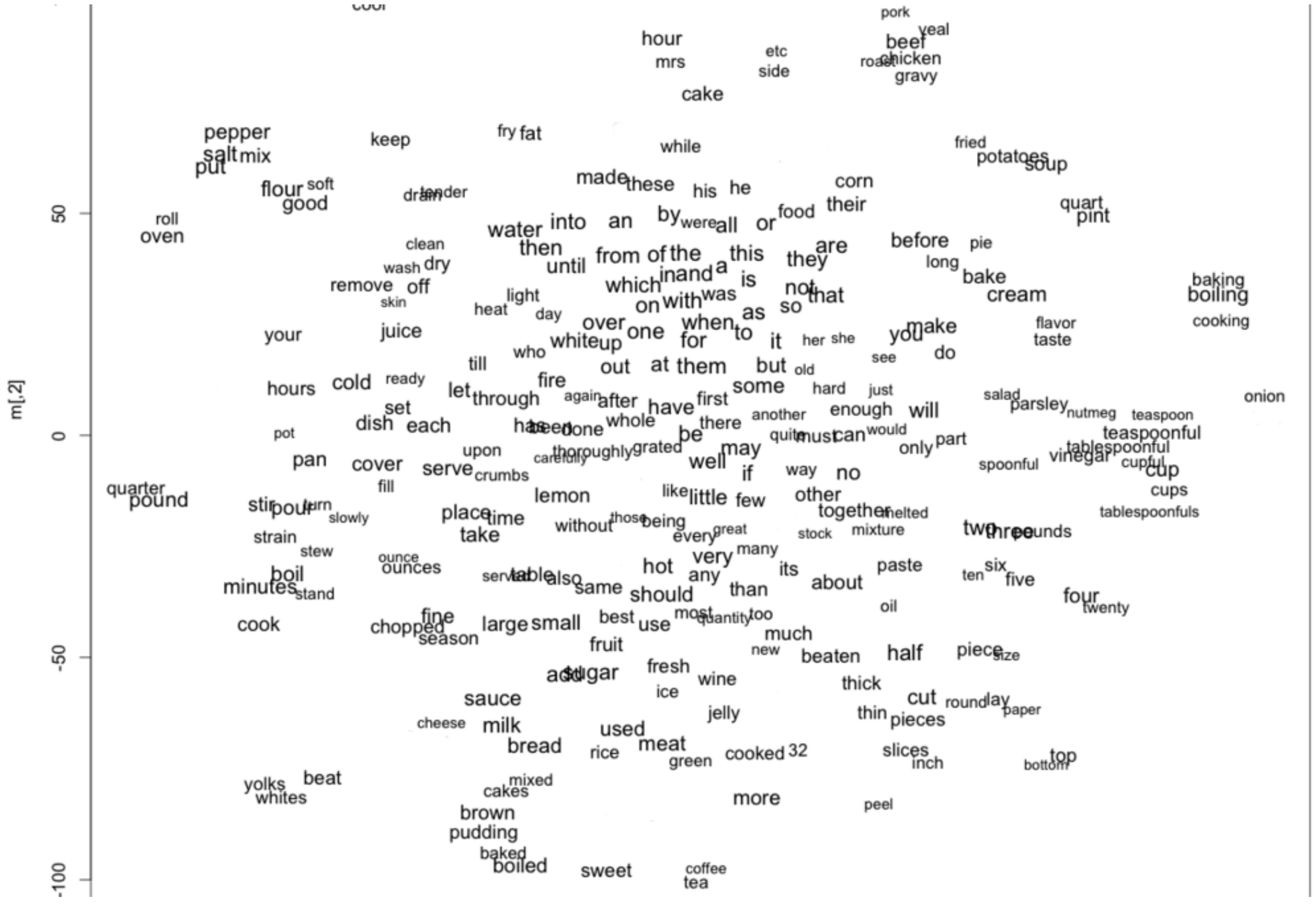
We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

word vectors are sometimes called word embeddings or word representations.
They are a distributed representation.

Source: Stanford 224N

Word Vectors

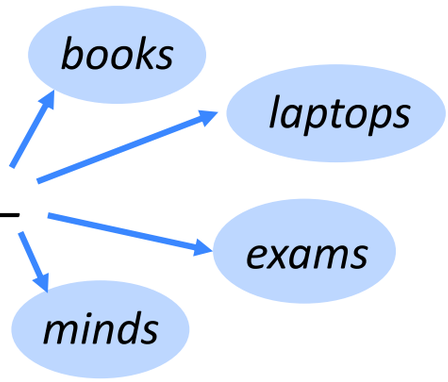


The Earliest Neural Model to Learn Word Embeddings

NTU NLP

A **language model** takes a list of words (history), and attempts to predict the word that follows them

the students opened their _____



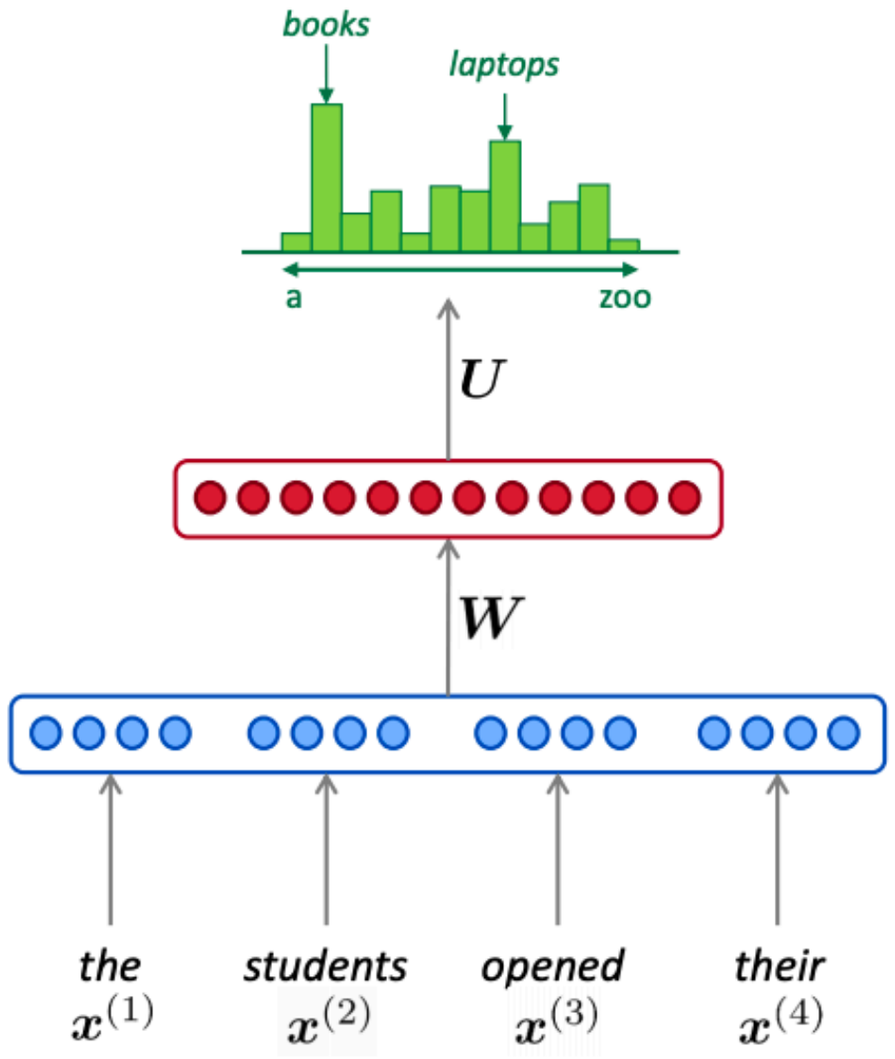
Google

what is the

what is the **weather**
what is the **meaning of life**
what is the **dark web**
what is the **xfl**
what is the **doomsday clock**
what is the **weather today**
what is the **keto diet**
what is the **american dream**
what is the **speed of light**
what is the **bill of rights**

Google Search I'm Feeling Lucky

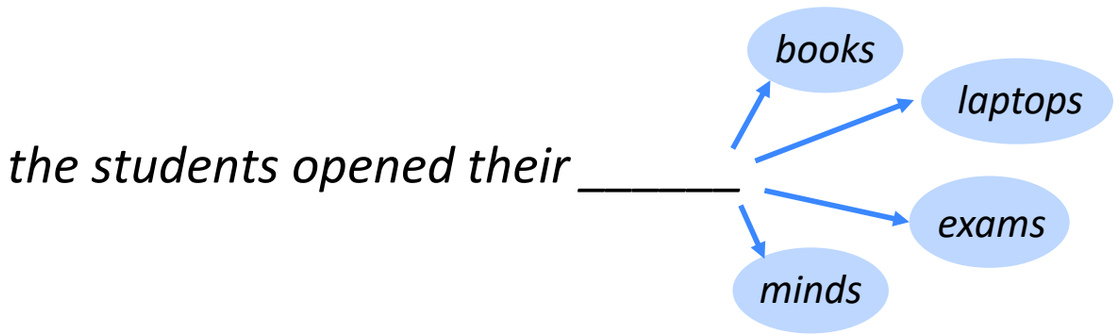
Assignment # 1

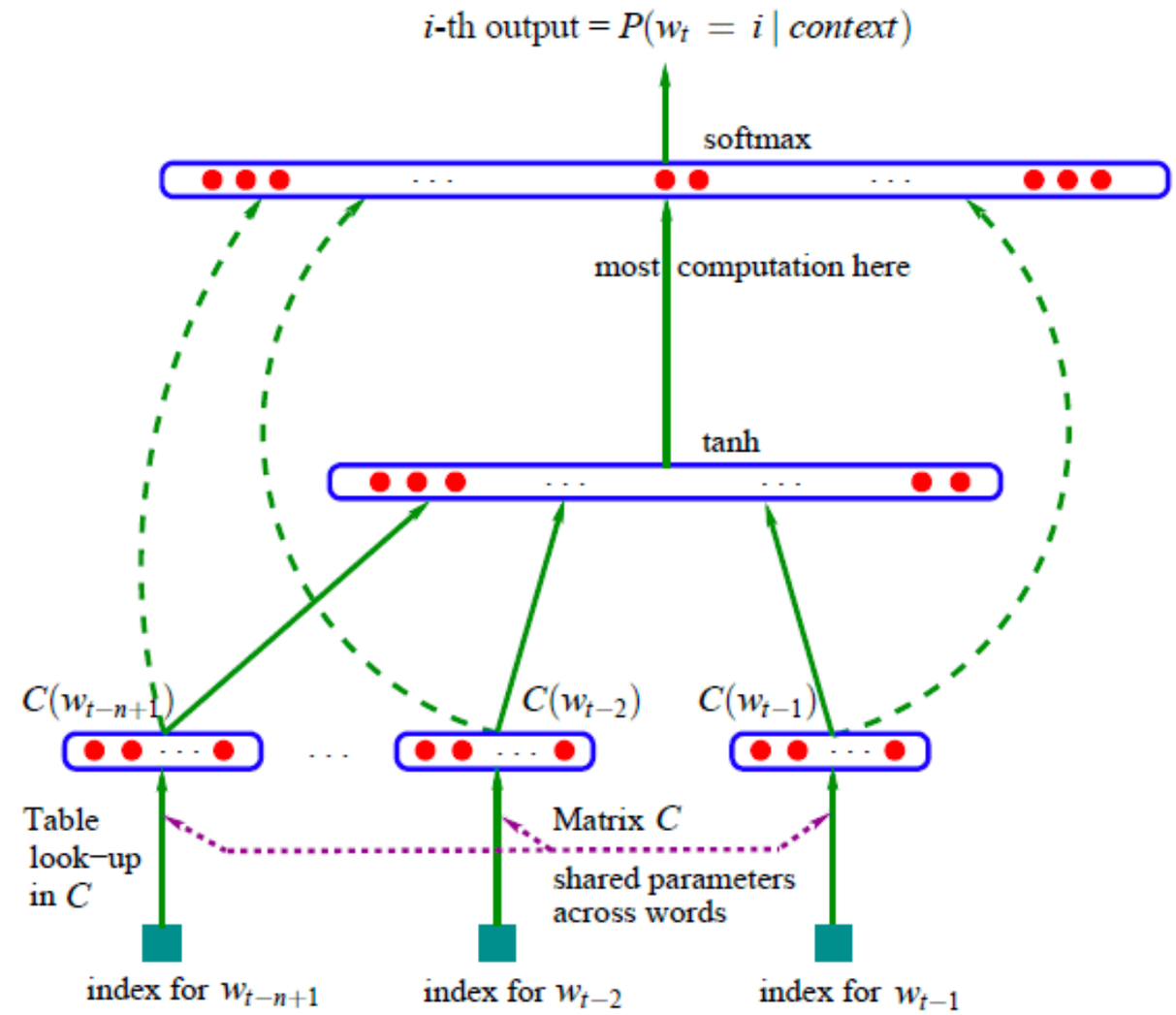


Bengio, 2003

Language Model

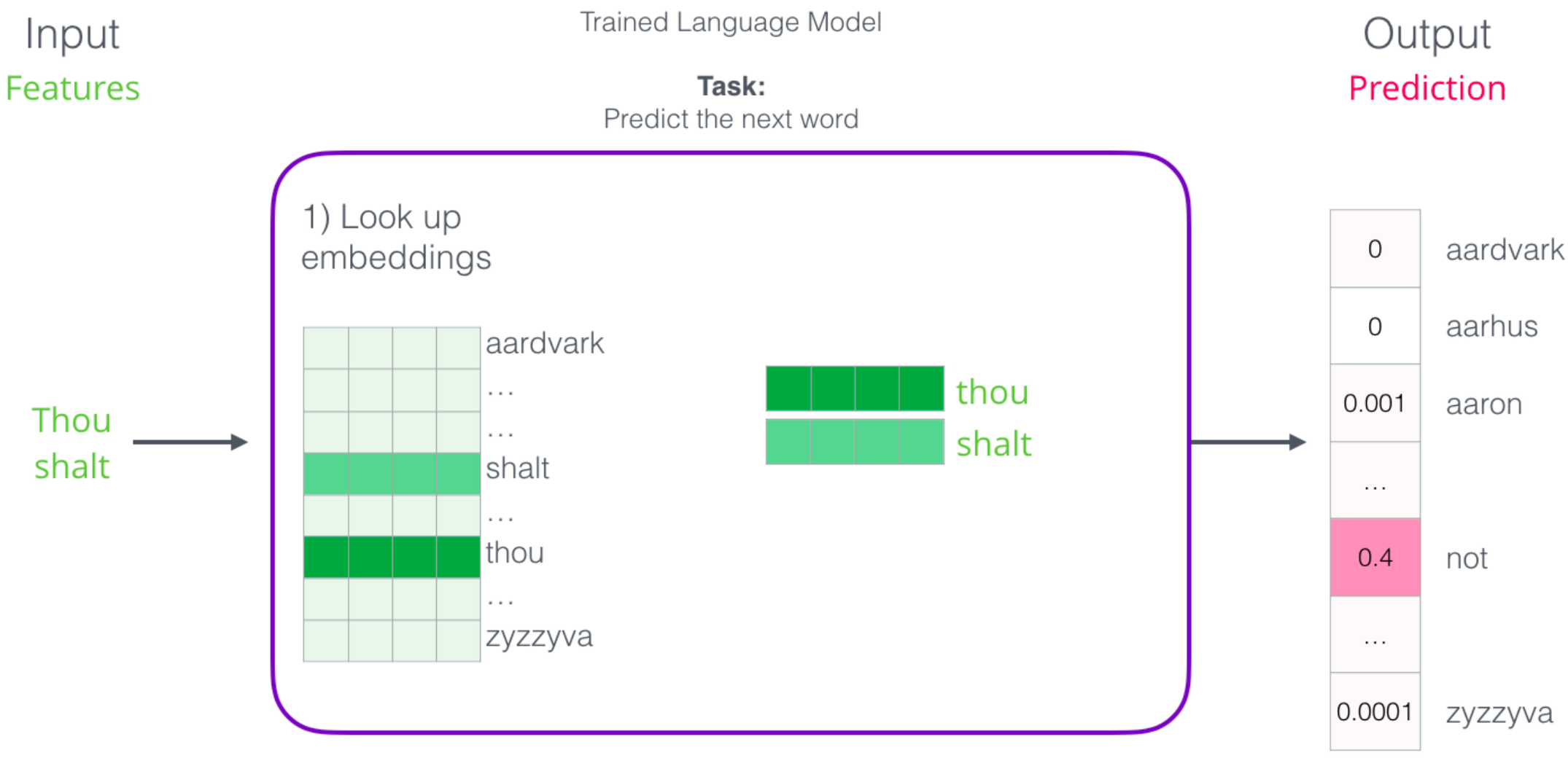
A language model takes a list of words (history), and attempts to predict the word that follows them





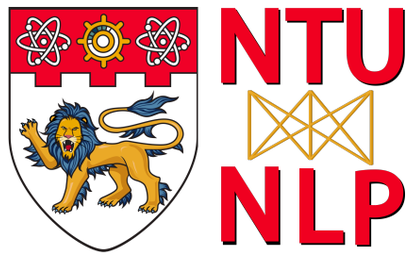
Source: Bengio, 2003

Language Model



Source:http://jalammar.github.io/

Lecture Plan



- Recap
  - Word meaning & Distributed representations
  - LM for word vector
- Word2Vec models & Negative sampling
- Incorporating subword information (FastText)
- Glove
- Word vector evaluation
- Cross-lingual word embeddings

Word2Vec

NTU NLP

Methods to efficiently create word embeddings

Idea:

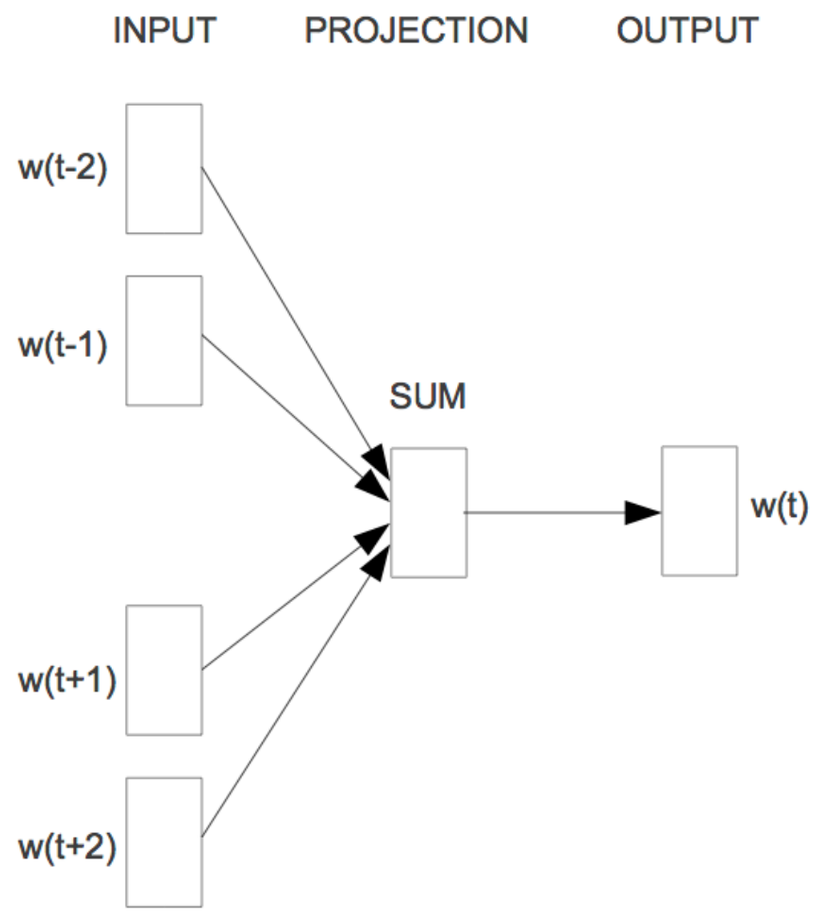
- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word c and context ("outside") words o
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

Source: Stanford 224N

Word2Vec - CBOW and Skipgram

Methods to efficiently create word embeddings

If our goal is just to learn word embeddings, instead of only looking words before the target word, we can also look at words after it.



This is called a **Continuous Bag of Words (CBOW)** architecture

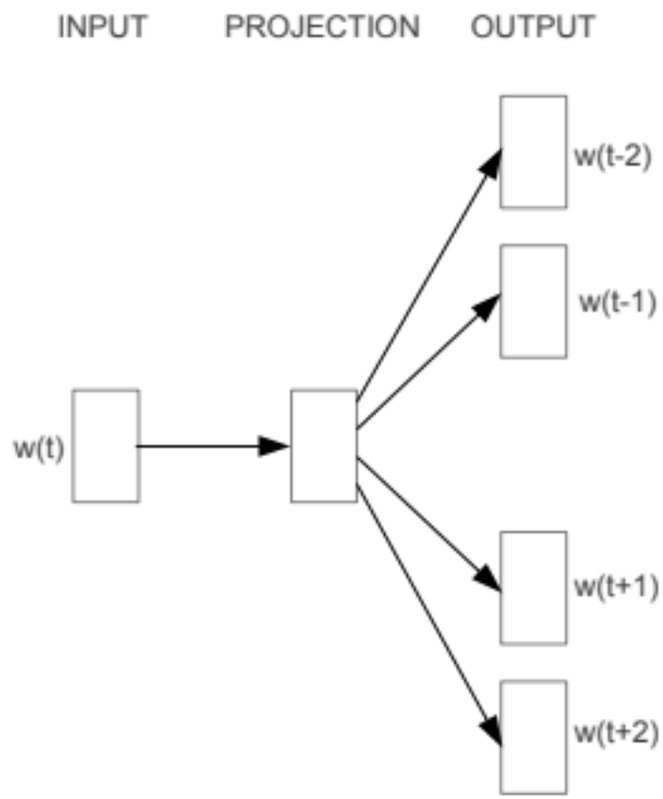
Source: Stanford 224N

Word2Vec - CBOW and Skipgram

NTU NLP

Methods to efficiently create word embeddings

Instead of guessing a word based on its context (the words before and after it), the other architecture tries to guess neighbouring words using the current word.

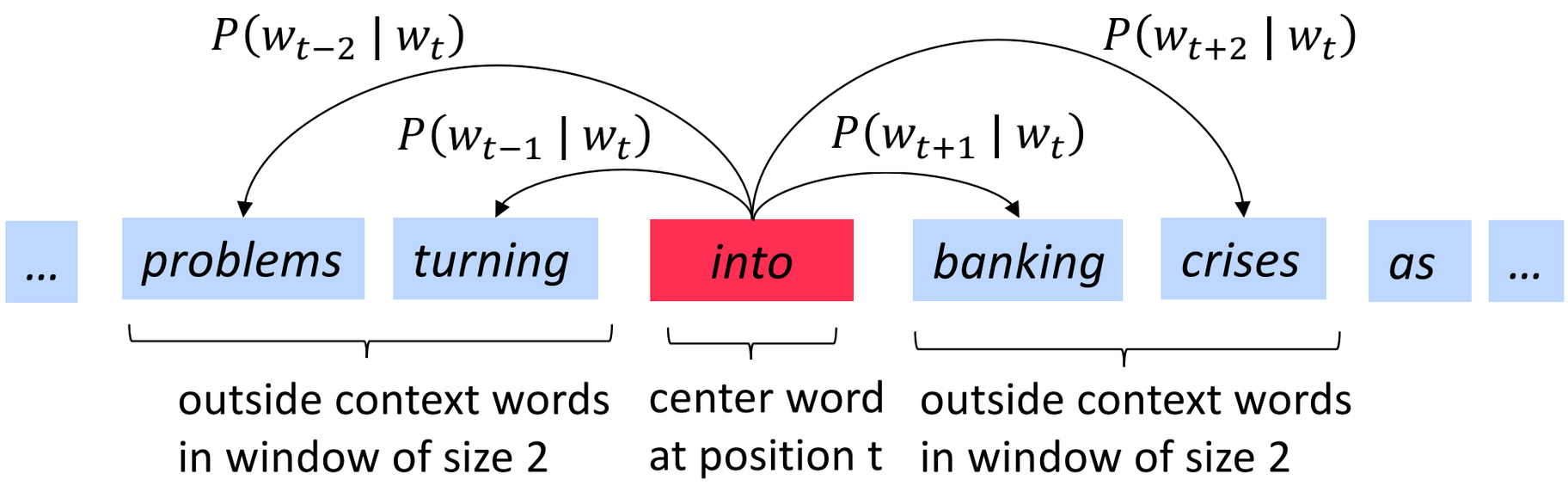


Skip-gram

This is called a Skipgram architecture

Source: Stanford 224N

Word2Vec - skipgram



Source: Stanford 224N

Word2Vec - skipgram

NTU NLP

Input-Output training pairs

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	…

input word	target word
not	thou
not	shalt
not	make
not	a

Source: http://jalammar.github.io/

Word2Vec – prediction

Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Dot product compares similarity of *o* and *c*.

$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$

Larger dot product = larger probability

Normalize over entire vocabulary to give probability distribution

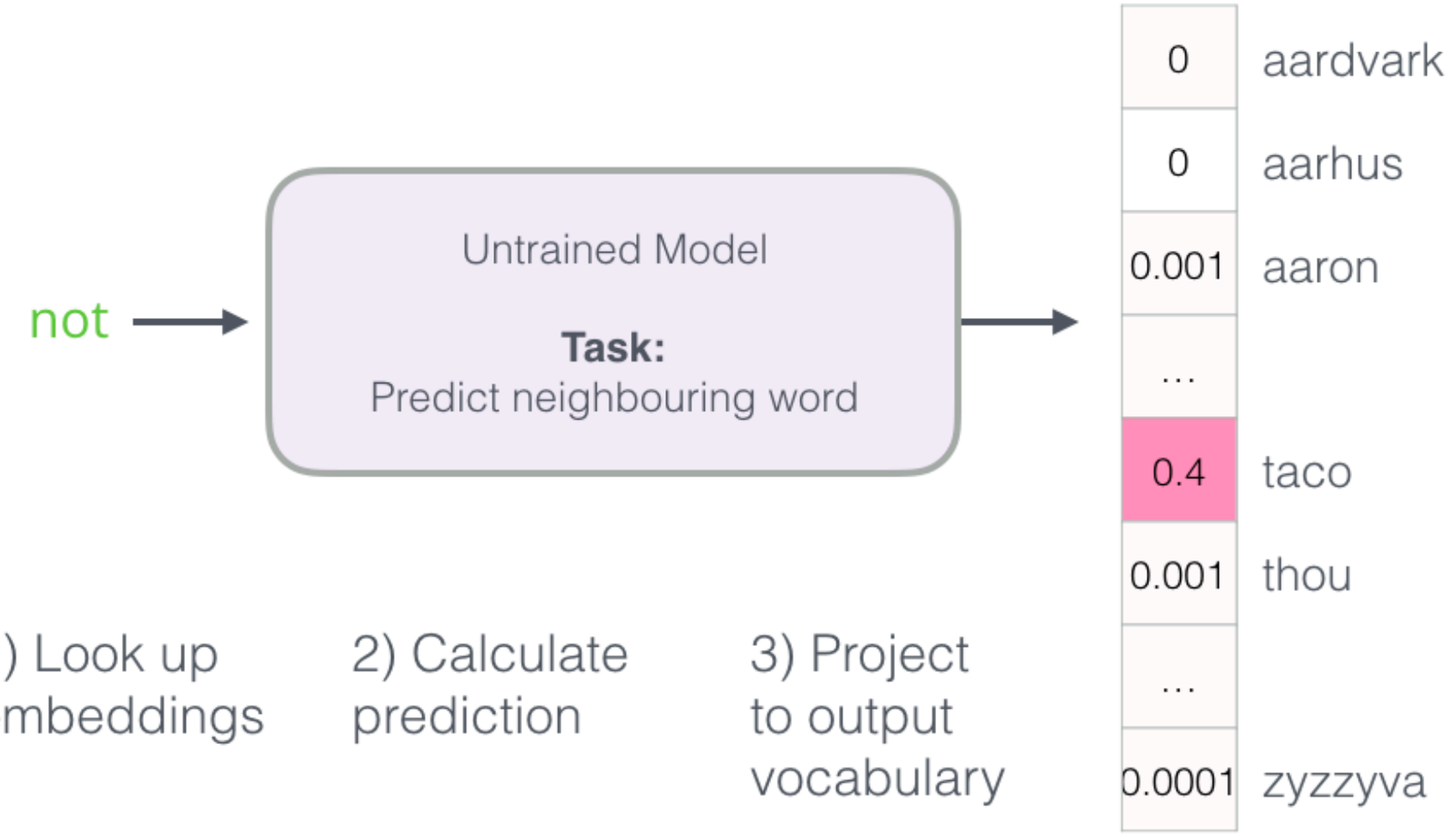
$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- "max" because amplifies probability of largest x_i
- "soft" because still assigns some probability to smaller x_i

Source: Stanford 224N

Word2Vec - prediction

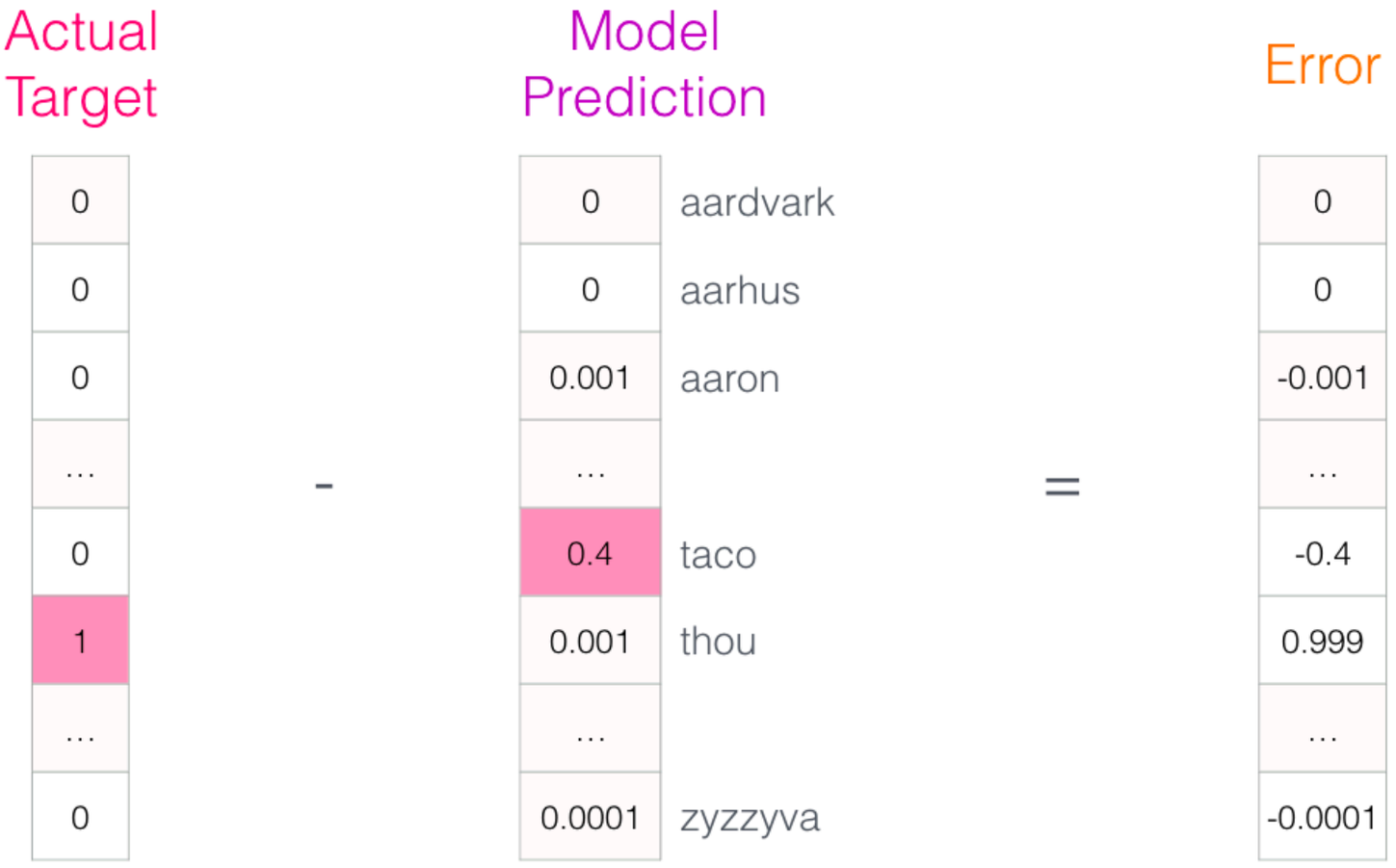
Thou shalt not make a machine in the likeness of a human mind



Word2Vec - loss/Gradient computation

Thou shalt not make a machine in the likeness of a human mind

$$\log p(o|c) = \log \frac{\exp({u_o}^T v_c)}{\sum_{w=1}^{V} \exp({u_w}^T v_c)}$$

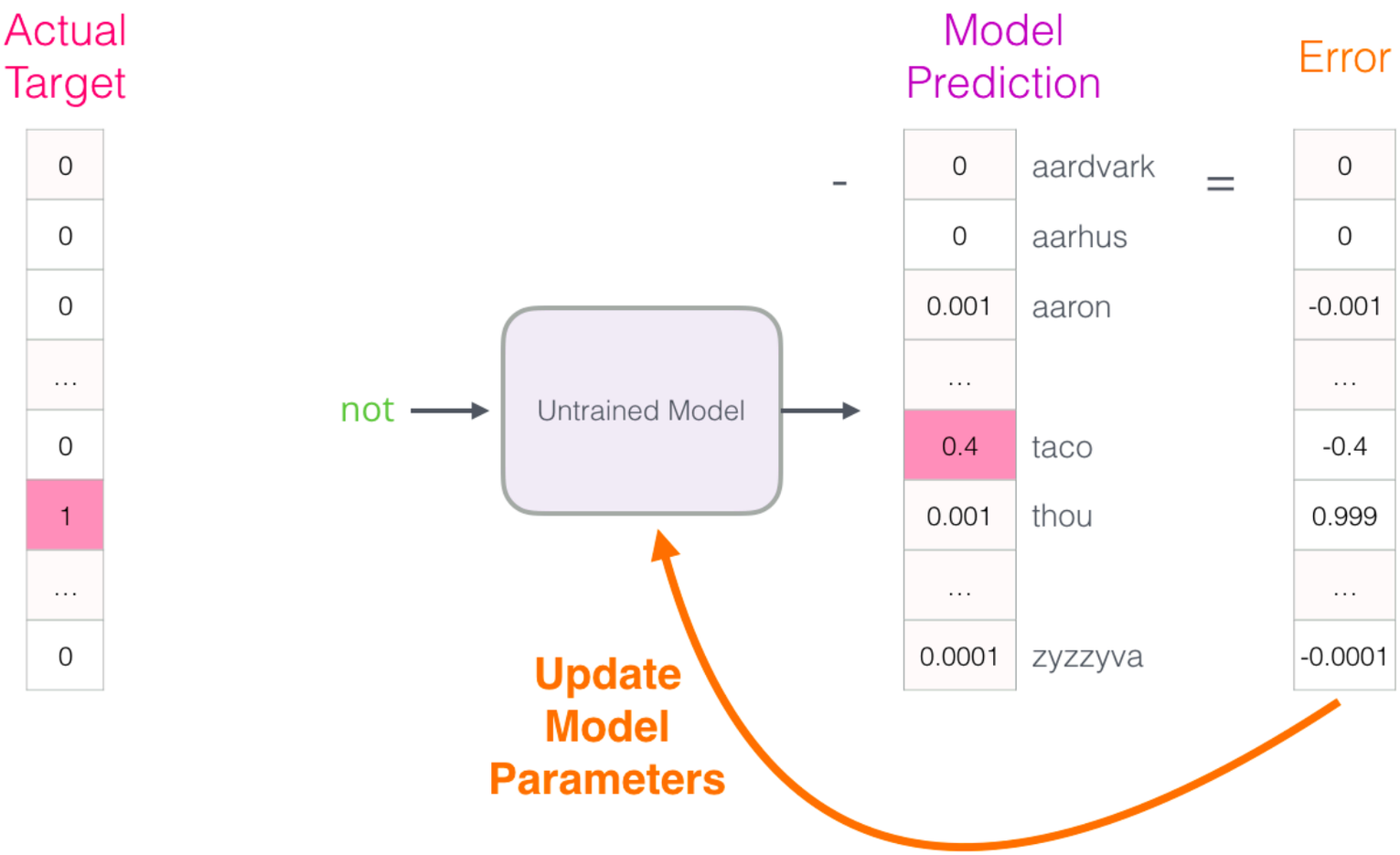


How far off was the model? We subtract the two vectors resulting in an error vector

Source: http://jalammar.github.io/

Word2Vec - training

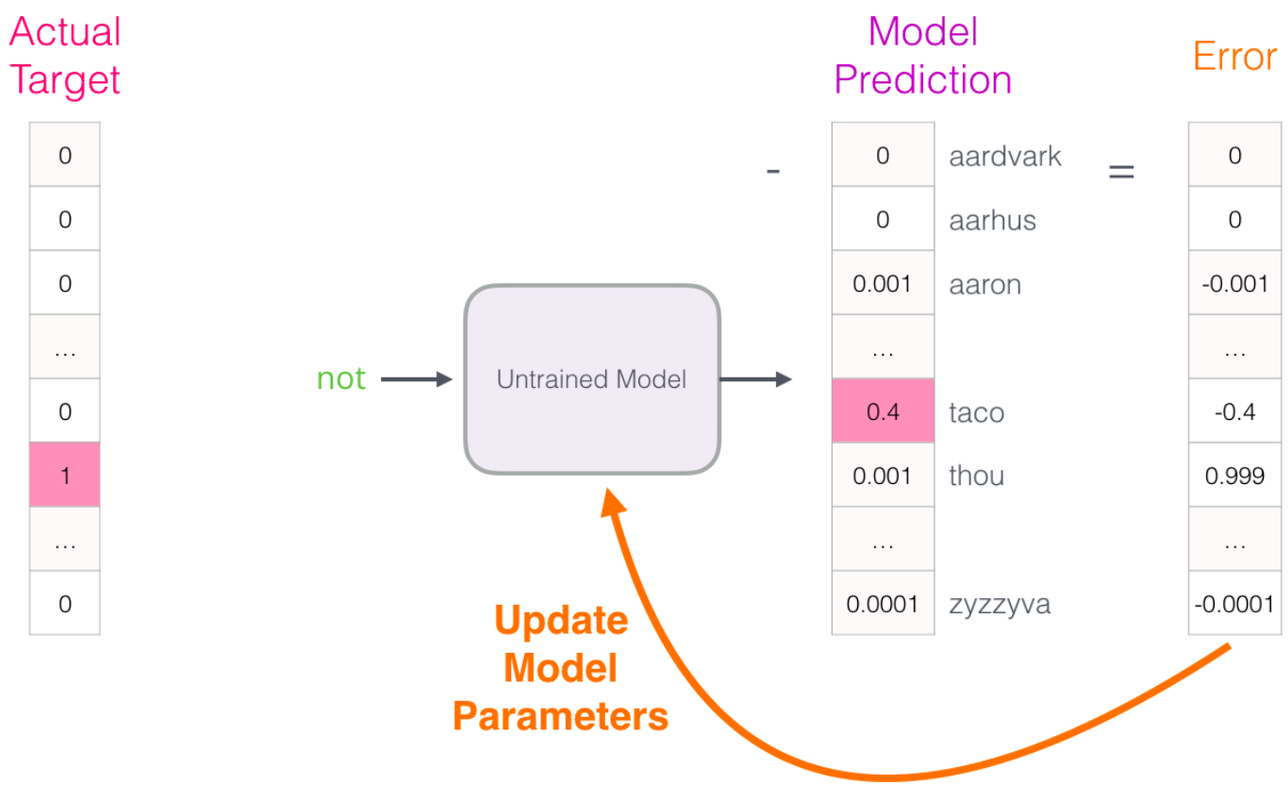
Thou shalt not make a machine in the likeness of a human mind



This error vector can now be used to update the model so the next time, it's a little more likely to guess thou when it gets not as input.

Source: http://jalammar.github.io/

Word2Vec - training



- We proceed to do the same process with the next sample in our dataset, and then the next, until we've covered all the samples in the dataset. That concludes one epoch of training. We do it over again for a number of epochs.
- Then we'd have our trained model and we can extract the embedding matrix from it and use it for any other application.

Source: http://jalammar.github.io/

Word2Vec - training

$$\log p(o|c) = \log \frac{\exp(u_o{}^T v_c)}{\sum_{w=1}^{V} \exp(u_w{}^T v_c)}$$

Actual Target		Model Prediction			Error
0		0	aardvark		0
0		0	aarhus		0
0		0.001	aaron		-0.001
…	-	…		=	…
0		0.4	taco		-0.4
1		0.001	thou		0.999
…		…			…
0		0.0001	zyzzyva		-0.0001

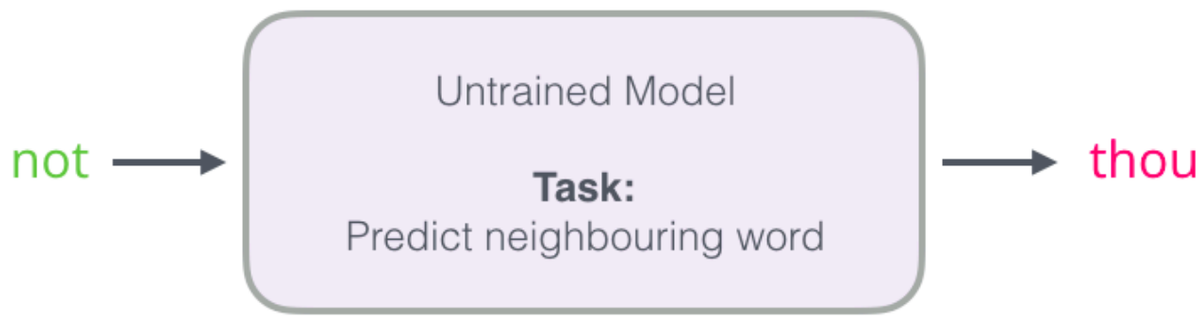
- One Issue: The last step is expensive (because of the denominator). We need to do it once for every training sample in our dataset (easily millions of times)

Source: http://jalammar.github.io/

Negative Sampling

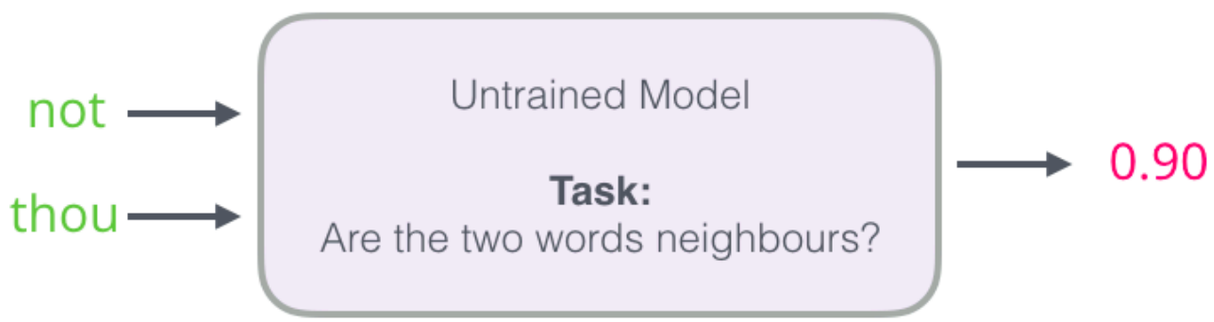
- One Issue: The last step is expensive (because of the denominator). We need to get rid of this!

Change Task from



$$\log p(o|c) = \log \frac{\exp({u_o}^T v_c)}{\sum_{w=1}^{V} \exp({u_w}^T v_c)}$$

To:

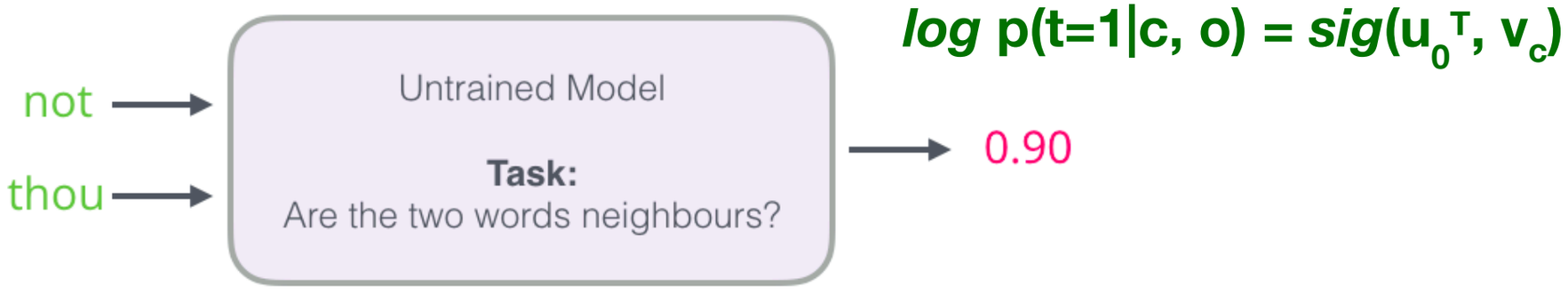


log p (t=1|c, o) = sig (u_0^T, v_c)

Source: http://jalammar.github.io/

Negative Sampling

To:



$$\log p(t=1|c, o) = sig(u_0^T, v_c)$$

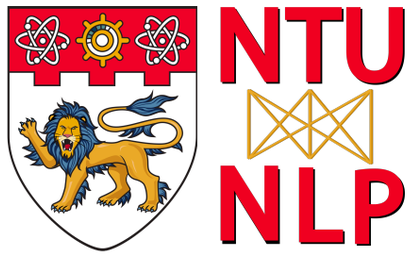
This simple switch changes the model to a logistic regression model (in the output) - thus it becomes much simpler and much faster to calculate.

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

Source: http://jalammar.github.io/

Negative Sampling



But there's one loophole! All of our examples are positive

input word	output word	target
not	thou	**1**
not		**0**
not		**0**
not	shalt	**1**
not	make	**1**

Negative examples

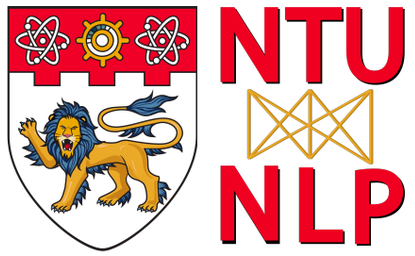
Introduce negative samples to our dataset - samples of words that are not neighbors. Our model needs to return 0 for those samples.

Randomly sample words from our vocabulary

Negative sampling is a version of Noise-contrastive estimation. We are contrasting the actual signal (positive examples of neighboring words) with noise (randomly selected words that are not neighbors)

Source: http://jalammar.github.io/

Skipgram with Negative Sampling



Skipgram

shalt	not	make	a	machine

input	output
make	shalt
make	not
make	a
make	machine

Negative Sampling

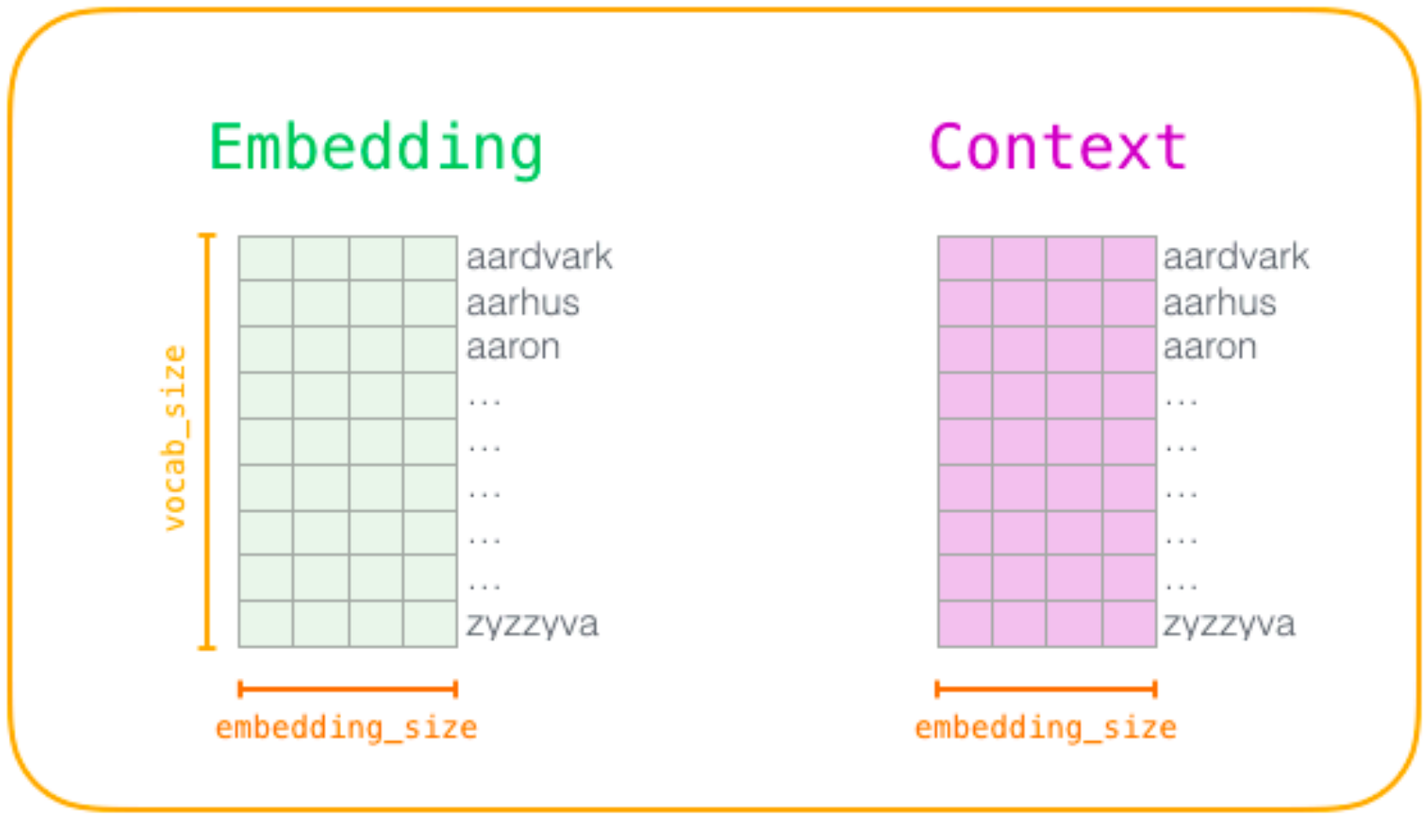
input word	output word	target
make	shalt	**1**
make	aaron	**0**
make	taco	**0**

$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_{j \sim P(w)} \left[\log \sigma\left(-u_j^T v_c\right)\right]$$

Maximize prob. that real outside word appears (co-occur), minimize prob. that random words appear around center word

Source: http://jalammar.github.io/

Skipgram Training



Create two matrices - an Embedding matrix and a Context matrix.

Initialize these matrices with random values

Source: http://jalammar.github.io/

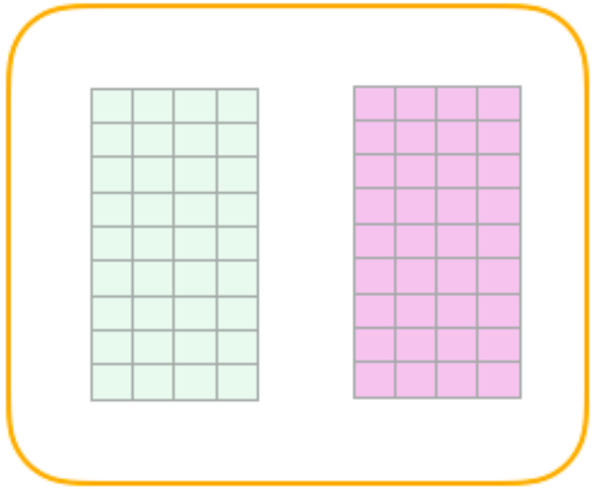
Skipgram Training

NTU NLP

dataset

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	mango	0
not	finglonger	0
not	make	1
not	plumbus	0
...	...	...

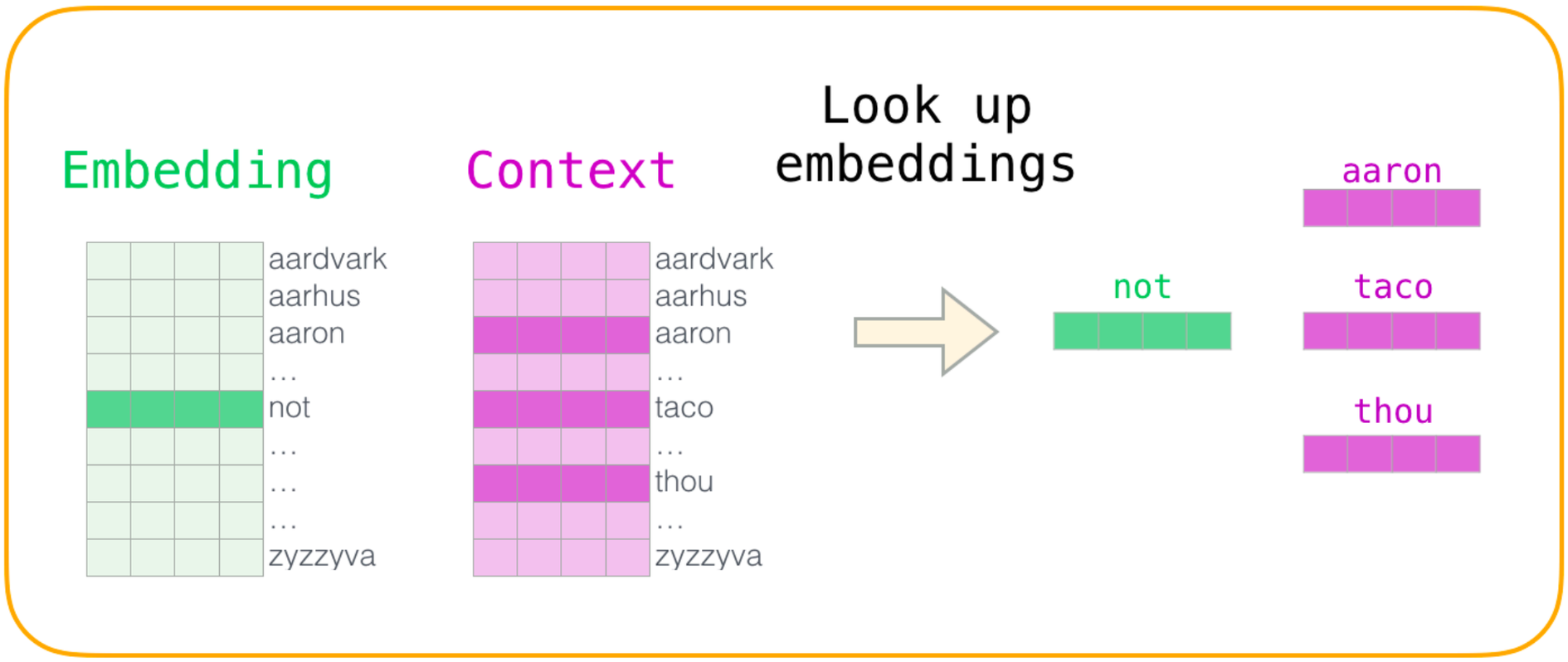
model



In each training step, we take one positive example and its associated negative examples.

Source: http://jalammar.github.io/

Skipgram Training

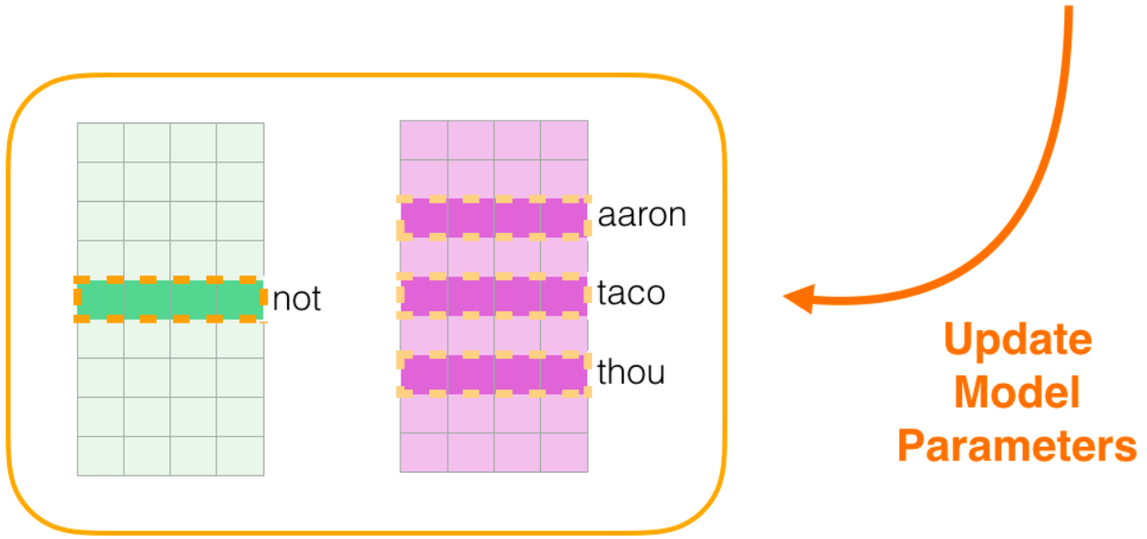


We proceed to look up their embeddings – for the input word, we look in the Embedding matrix. For the context words, we look in the Context matrix (even though both matrices have an embedding for every word in our vocabulary).

Source: http://jalammar.github.io/

Skipgram Training

input word	output word	target	input • output	sigmoid()	Error
not	thou	1	0.2	0.55	0.45
not	aaron	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68



We proceed to compute the predictions, the loss, gradients and model updates

Source: http://jalammar.github.io/

Skipgram Training

Two hyper-parameters:

Window Size and Number of Negative Samples

Window size: 5

Window size: 15

Task dependent
(default is 5)

Negative samples: 2

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0

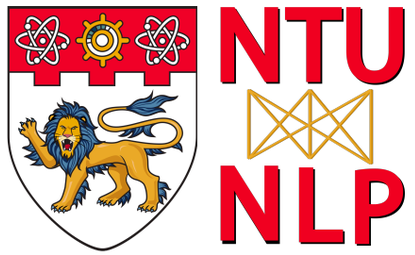
Negative samples: 5

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0
make	finglonger	0
make	plumbus	0
make	mango	0

5 - 20
5 is good enough

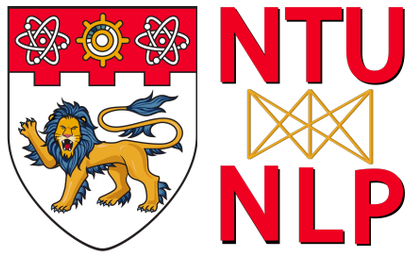
Source: http://jalammar.github.io/

Incorporating Subword Information



- Morphology: Words have internal structures and formation methods; e.g.,"dog", "dogs", "doggy", and "doghouse". Same root, "dog", but use different suffixes to change the meaning of the word.
- This relation can be extended to other words; e.g., "cat" and "cats". The relationship between "boy" and "boyfriend" is just like the relationship between "girl" and "girlfriend".
- Word2vec does not directly use morphology; "dog" and "dogs" are represented by two different vectors.
- FastText [Bojanowski et al., 2017] uses subword embedding in the skip-gram model of word2vec.

FastText



- Each central word is represented as a collection of subwords. E.g., for n = 3, "house" is represented as "<ho", "hou", "ous", "use", "se>", and "<house>".
- For each central word, we record the union of all its subwords with length of 3 to 6 and special subwords (e.g., "<house>"). Thus, the dictionary is the union of the collection of subwords of all words.

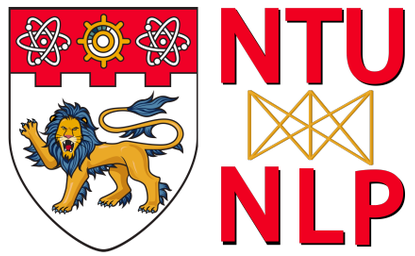
$$\sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top$$

G_w => union of all subwords of w

- Remaining is same as the skipgram model

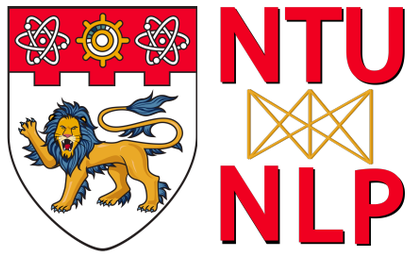
[Bojanowski et al., 2017]

Lecture Plan



- Recap
  - Word meaning & Distributed representations
  - LM for word vector
- Word2Vec models & Negative sampling
- Incorporating subword information (FastText)
- Glove
- Word vector evaluation
- Cross-lingual word embeddings

Capturing co-occurrence counts directly



- Word2Vec only takes local contexts into account
- It does not take advantage of global count statistics (explicitly).
- We can compute a word co-occurrence matrix X

- Similar to word2vec, use window around each word – captures both syntactic and semantic information

Sample corpus

"The cat sat on the mat"

- Window length 2 (more common: 5–10)
- Symmetric (irrelevant whether left or right context)
- Considers global statistics

Co-occurrence matrix

	the	cat	sat	on	mat
the	2	1	2	1	1
cat	1	1	1	1	0
sat	2	1	1	1	0
on	1	1	1	1	1
mat	1	0	0	1	1

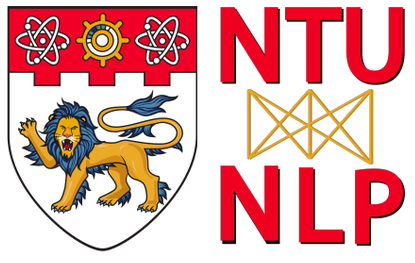
Problems with simple co-occurrence vectors

Co-occurrence matrix

	the	cat	sat	on	mat
the	2	1	2	1	1
cat	1	1	1	1	0
sat	2	1	1	1	0
on	1	1	1	1	1
mat	1	0	0	1	1

- Increase in size with vocabulary
- Very high dimensional: requires a lot of storage
- Classification models have sparsity issues (less robust)

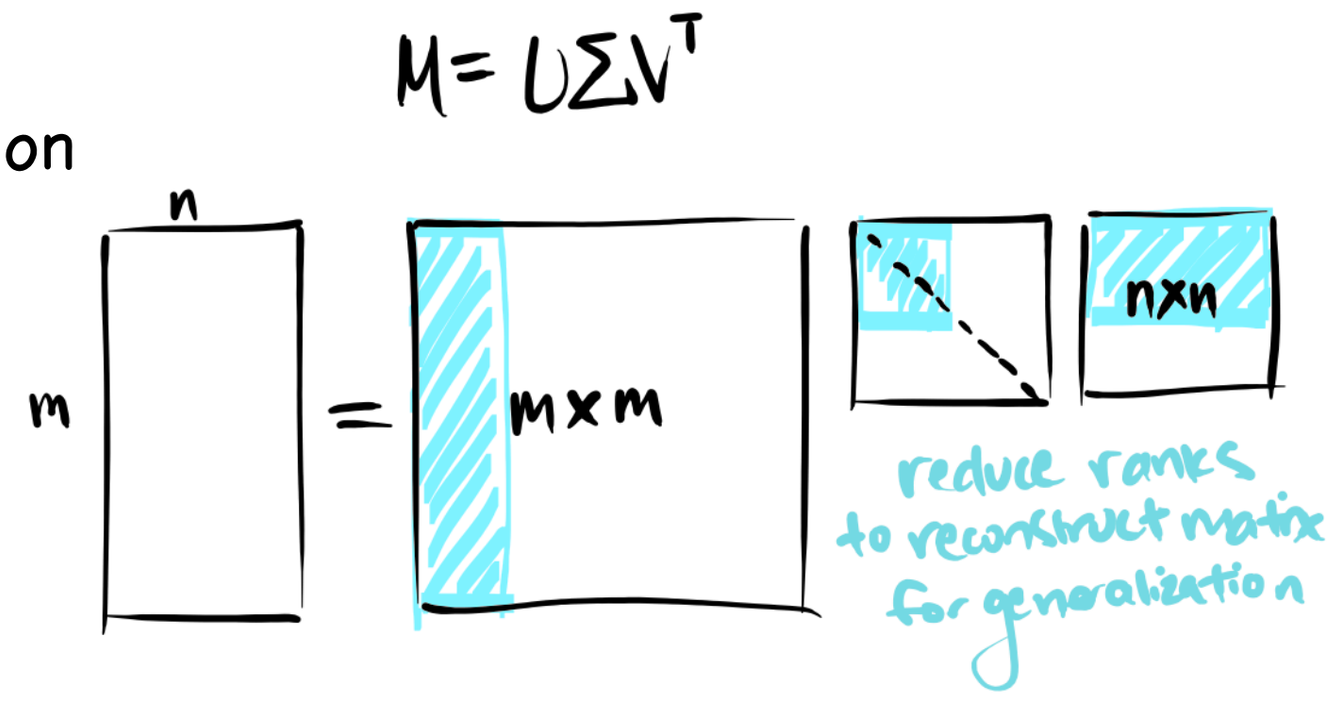
Solution – Dimension Reduction



- Idea: store "most" of the important information in a fixed, small number of dimensions: a dense vector
- Usually 25–1000 dimensions, similar to word2vec
- How to reduce the dimensionality?

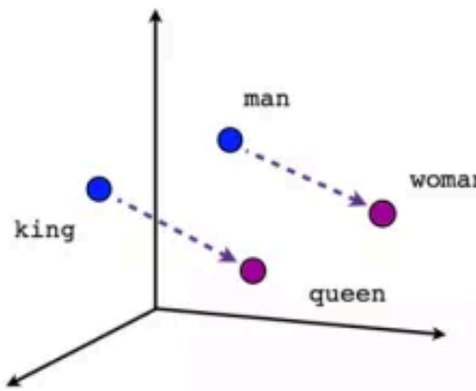
Singular Value Decomposition (aka. Latent Semantic Analysis)

- Factorize M into $U\Sigma V^T$, where U and V are orthonormal
- Gives the best rank k approximation to M, in terms of least squares
- U = LSA vectors

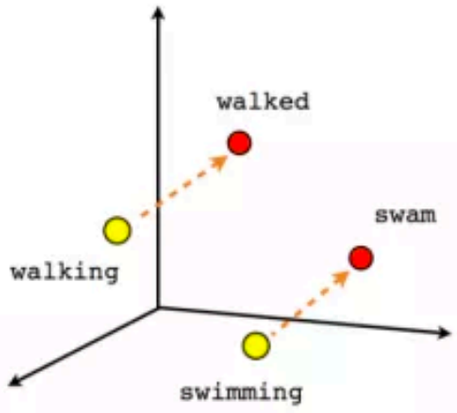


Problems with LSA vectors

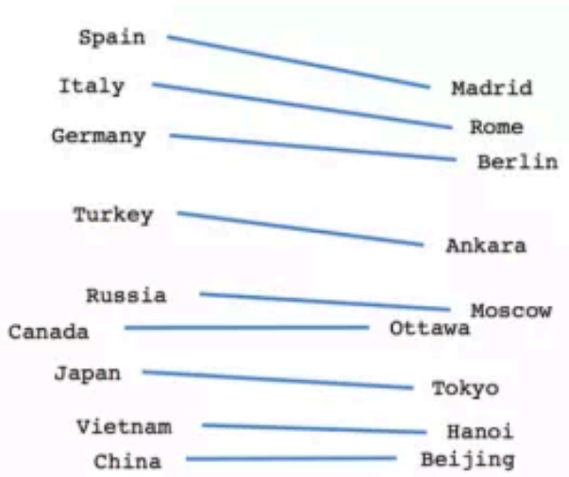
Although LSA considers global statistics, word analogies were difficult to show with LSA (without hacks)



Male-Female



Verb tense

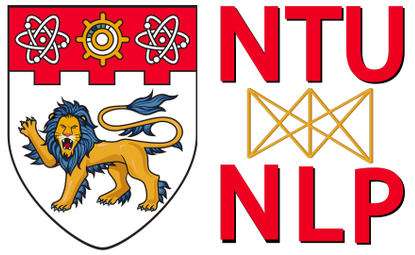


Country-Capital

Vector **dimensions should capture meaning (semantics) and syntax.**

Some dimensions might capture male vs. female (semantics), some might capture noun vs. verb, present tense vs. past tense (syntax), plural or singular (syntax), country-capital (semantics), do-doer, etc..

Count based vs. direct prediction



- LSA, HAL (Lund & Burgess),
- COALS, Hellinger-PCA (Rohde et al, Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- Skip-gram/CBOW (Mikolov et al)
- NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

GloVe aims to take the best of both worlds: take **global** information into account while learning **dimensions of meaning (and syntax)**.

Source: 224n (Stanford)

Glove

NTU NLP

Main Idea: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

[Pennington, Socher, and Manning, EMNLP 2014]

Source: 224n (Stanford)

Glove

Main Idea: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9 x 10^{-4}	6.6 x 10^{-5}	3.0 x 10^{-3}	1.7 x 10^{-5}
$P(x\|\text{steam})$	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5 x 10^{-2}	1.36	0.96

$$p(i|j) = \frac{\#\text{ of times word } j \text{ appears with word } i}{\#\text{ of times word } j \text{ appears}} = \frac{n(i,j)}{n(j)}$$

[Pennington, Socher, and Manning, EMNLP 2014]

Source: 224n (Stanford)

Glove

Question: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

Answer: Log-bilinear model:

$$w_i \cdot w_j = \log P(i|j)$$

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9 x 10^{-4}	6.6 x 10^{-5}	3.0 x 10^{-3}	1.7 x 10^{-5}
$P(x\|\text{steam})$	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5 x 10^{-2}	1.36	0.96

With vector differences

$$w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$$

[Pennington, Socher, and Manning, EMNLP 2014]

Source: 224n (Stanford)

Glove

$$(\mathbf{w}_a - \mathbf{w}_b)^T \mathbf{w}_x = \log \frac{p(x|a)}{p(x|b)} = \log \frac{n(a,x)/n(a)}{n(b,x)/n(b)}$$

$$\mathbf{w}_a{}^T \mathbf{w}_x - \mathbf{w}_b{}^T \mathbf{w}_x + c_a - c_b = \log n(a,x) - \log n(a) - \log n(b,x) + \log n(b)$$

Add a bias term for each word to capture the fact that some words occur more often than others

We can convert this equation into an equation over a single entry in the co-occurrence matrix.

$$\begin{aligned} \mathbf{w}_a{}^T \mathbf{w}_x + c_a &= \log n(a,x) - \log n(a) \\ \mathbf{w}_a{}^T \mathbf{w}_x + c_a + \hat{c}_x &= \log n(a,x) \end{aligned}$$

$$\begin{aligned} \mathbf{w}_b{}^T \mathbf{w}_x + c_b &= \log n(b,x) - \log n(b) \\ \mathbf{w}_b{}^T \mathbf{w}_x + c_b + \hat{c}_x &= \log n(b,x) \end{aligned}$$

[Pennington, Socher, and Manning, EMNLP 2014]

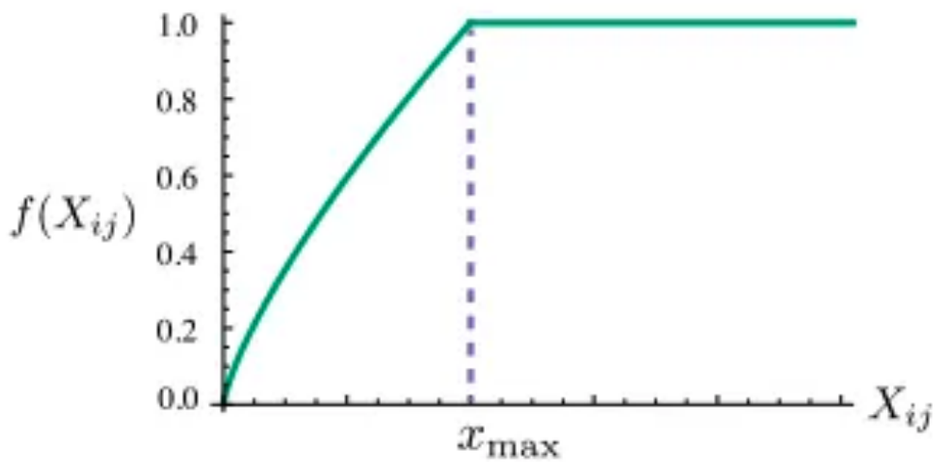
Source: 224n (Stanford)

GloVe

$$\mathbf{w}_a{}^T \mathbf{w}_x + c_a + \hat{c}_x \quad = \quad \log n(a, x)$$

One issue: it weights all co-occurrences equally. Co-occurrences that are **infrequent will tend to be noisy and unreliable**, so we want to weight frequent co-occurrences more heavily. But, we also don't want co-occurrences like "it is" dominating the loss, so we don't want to weight too heavily based on frequency.

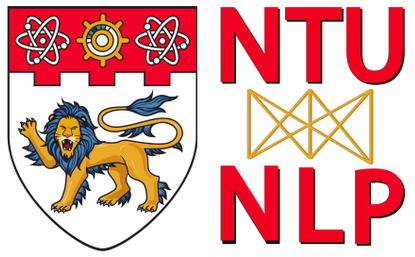
$$J(\theta) = \sum_{i,j} f(i,j)(\mathbf{w}_i{}^T \mathbf{w}_j + c_i + \hat{c}_j - \log n(i,j))^2$$



$$\text{weight}(x) = \min(1, (x/x_{max})^{\frac{3}{4}})$$

[Pennington, Socher, and Manning, EMNLP 2014]

Relation between W2V & Glove



$$J(\theta) = \sum_{i,j} f(i,j)(\mathbf{w}_i{}^T \mathbf{w}_j + c_i + \hat{c}_j - \log n(i,j))^2$$

Word2Vec: probability of seeing word j in the context of word i

$$Q_{ij} = \text{softmax}(\text{dot}(w_i, \tilde{w}_j))$$

The overall loss is equivalent to $\sum_{i,j} X_{ij} \log Q_{ij}$

Group over the context word i

$$\sum_i X_i \sum_j \frac{X_{ij}}{X_i} \log Q_{ij} = \sum_i X_i \sum_j P_{ij} \log Q_{ij}$$

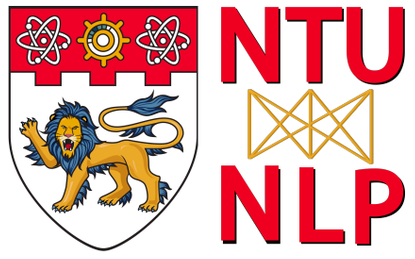
frequency (word i) weighted sum over the cross-entropy

The weighting factor comes from the fact that we stream across all the data equally

the weighting factor can be an arbitrary function of frequency

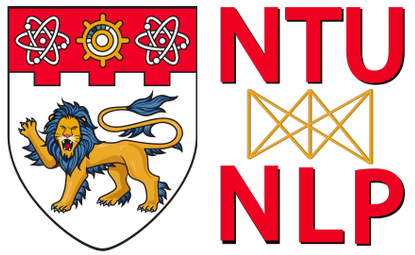
$$\sum_{ij} \text{weight}(X_{ij}) P_{ij} \log Q_{ij}$$ **CE vs. log squared distance**

Lecture Plan



- Recap
  - Word meaning & Distributed representations
  - LM for word vector
- Word2Vec models
- Negative sampling
- Incorporating subword information
- Glove
- Word vector evaluation
- Cross-lingual word embeddings

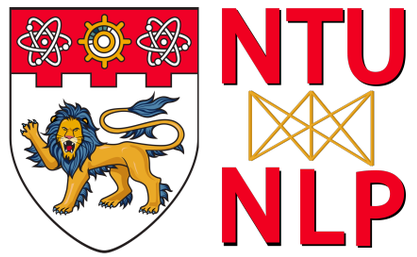
Evaluating Word Vectors



General evaluation types in NLP: Intrinsic and extrinsic

- Intrinsic evaluation:
  - Evaluation on the actual task or some intermediate (related) task
  - Helps to understand the system/model
  - Fast to check
  - Not clear if really helps end applications unless direct correlation to the application is established
- Extrinsic evaluation:
  - Evaluation on a real application (e.g., MT, Summarization, Dialogue)
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy (factor out other aspects), then a Win!

Intrinsic Evaluation of Word Vectors



- Word Vector Analogies:

man:woman :: king: ?

man is to woman as king is to queen

a is to a^* as b is to b^*

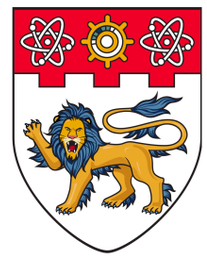
$$a - a^* = b - b^*$$

$$b - a + a^* = b^*$$

king - man + woman = queen

Word vectors should capture "relational similarities"

Intrinsic Evaluation of Word Vectors



NTU NLP

- Word Vector Analogies:

$$\underset{b}{\text{Tokyo}} - \underset{a}{\text{Japan}} + \underset{a^*}{\text{France}} = \underset{b^*}{\text{Paris}}$$

$$\underset{b}{\text{best}} - \underset{a}{\text{good}} + \underset{a^*}{\text{strong}} = \underset{b^*}{\text{strongest}}$$

We wish to find:

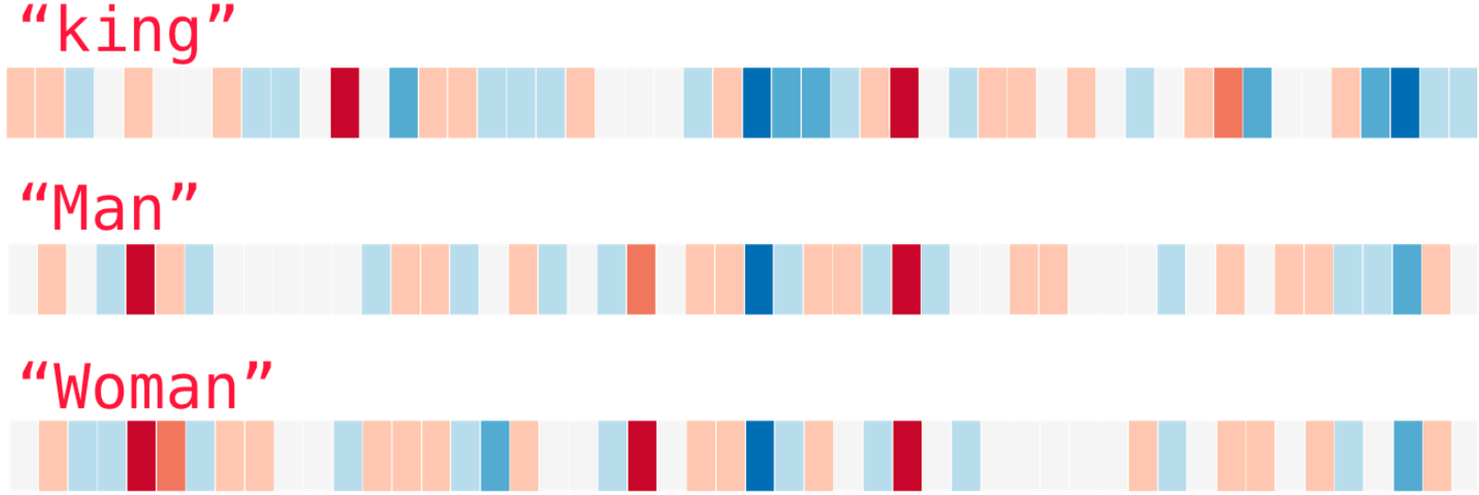
$$\arg\max_{b^*}(\cos(b^*, b - a + a^*))$$

$$=$$

$$\arg\max_{b^*}(\cos(b^*, b) - \cos(b^*, a) + \cos(b^*, a^*))$$

vector arithmetic = similarity arithmetic

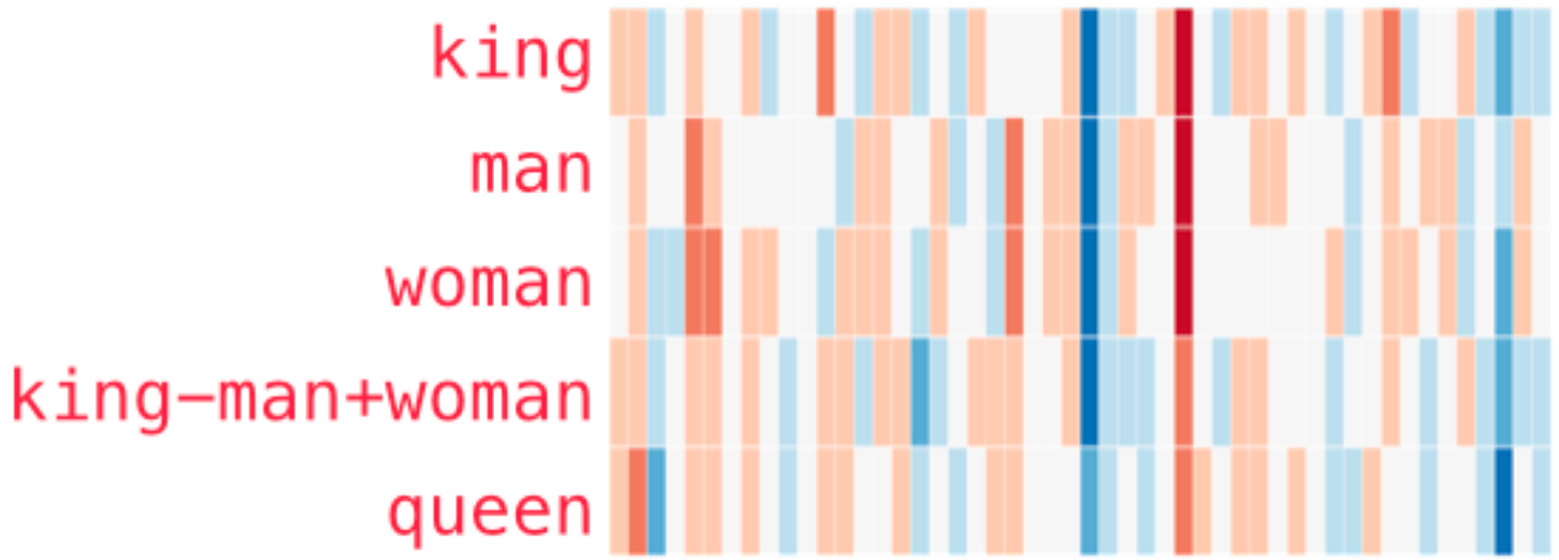
Word2Vec vectors



Source:http://jalammar.github.io/

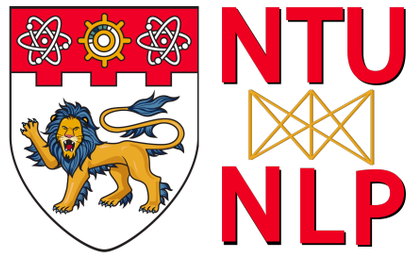
Word2Vec vectors

king − man + woman ~= queen



Source: http://jalammar.github.io/

Intrinsic Evaluation of Word Vectors



- Word Analogy Dataset:

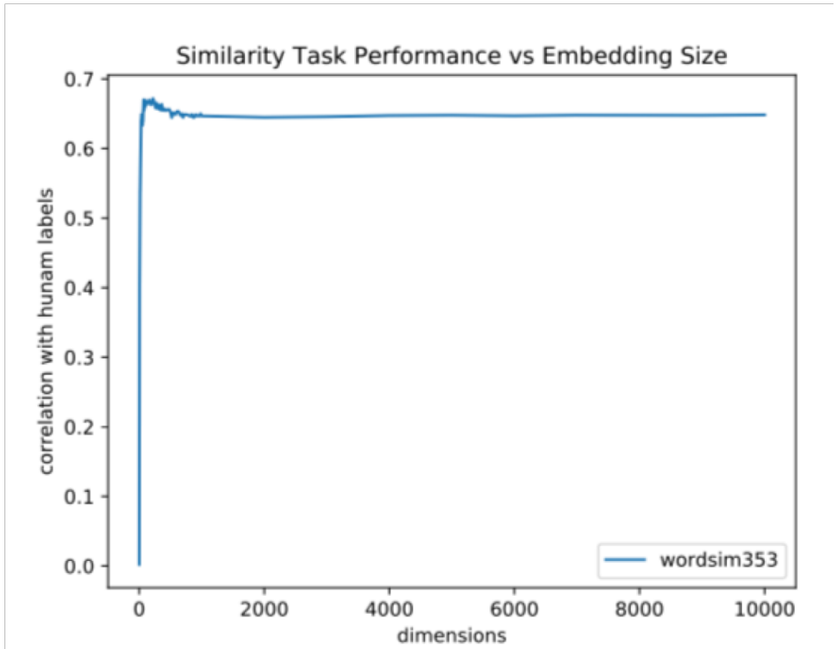
http://code.google.com/p/word2vec/source/browse/trunk/questions- words.txt

: city-in-state
Chicago Illinois Houston Texas
Chicago Illinois Philadelphia Pennsylvania
Chicago Illinois Phoenix Arizona
Chicago Illinois Dallas Texas
Chicago Illinois Jacksonville Florida
Chicago Illinois Indianapolis Indiana
Chicago Illinois Austin Texas
Chicago Illinois Detroit Michigan
Chicago Illinois Memphis Tennessee
Chicago Illinois Boston Massachusetts

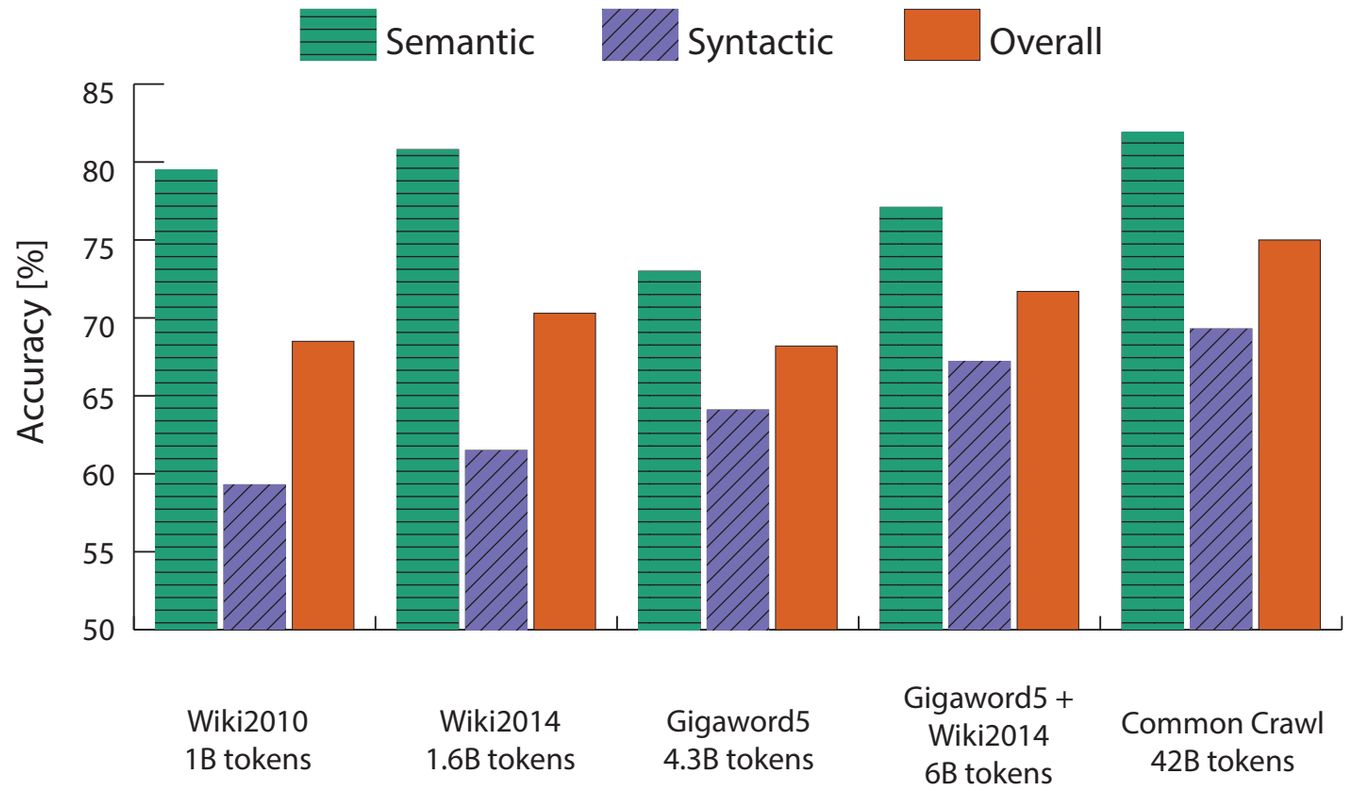
: gram4-superlative
bad worst big biggest
bad worst bright brightest
bad worst cold coldest
bad worst cool coolest
bad worst dark darkest
bad worst easy easiest
bad worst fast fastest
bad worst good best
bad worst great greatest

Source: 224n (Stanford)

Hyperparameters and Datasets



Good dimension is ~300

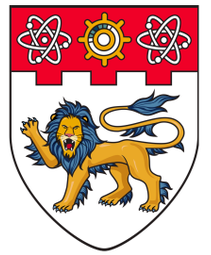


More data helps, Wikipedia is better than news text!

https://papers.nips.cc/paper/7368-on-the-dimensionality-of-word-embedding.pdf

Source: 224n (Stanford)

Another Intrinsic Evaluation



NTU NLP

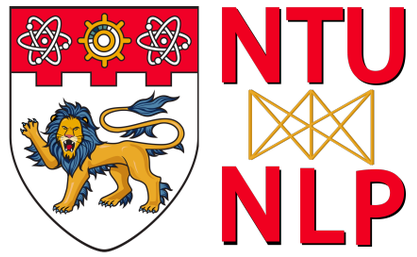
- Word vector distances and their correlation with human judgments

- Example dataset: WordSim353
  http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Source: 224n (Stanford)

Extrinsic Evaluation



- Word vectors are crucial in almost all intermediate and end tasks.

Intermediate tasks:

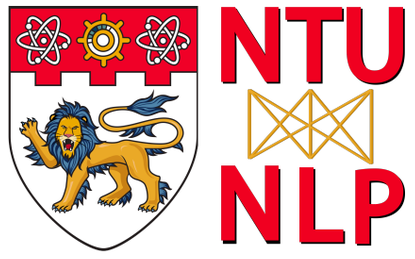
- Named Entity Recognition (Person, City, Org)
- Parsing (syntactic, semantic, discourse)
- Cross-lingual embeddings
- ….

End Applications:

- Machine Translation
- Sentiment Analysis
- Summarization
- Dialogue systems
- ….

Source: 224n (Stanford)

Lecture Plan



- Recap
  - Word meaning & Distributed representations
  - LM for word vector
- Word2Vec models
- Negative sampling
- Incorporating subword information
- Glove
- Word vector evaluation
- Cross-lingual word embeddings

NLP for Other Languages

NTU NLP

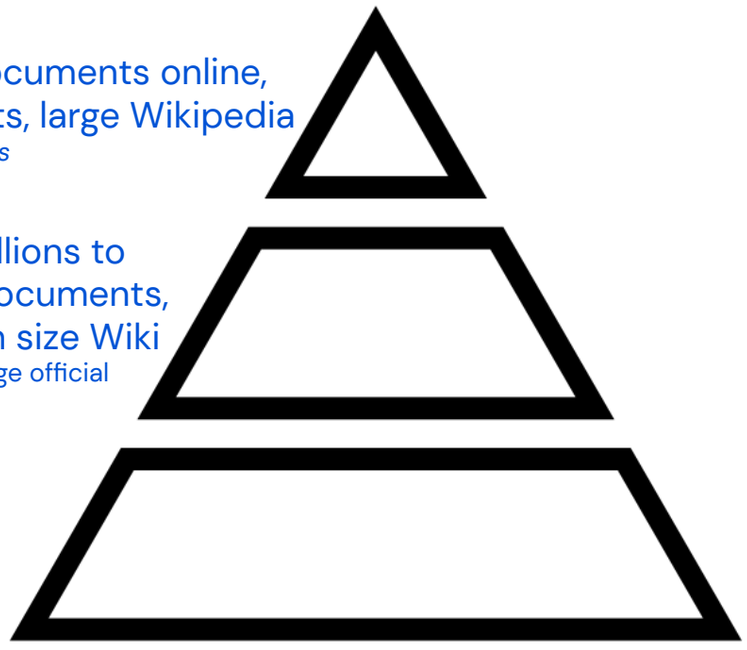
NLP not just for English but for the rest of the world's 6,500 languages

source: (Ruder, 2019)

NLP for Other Languages

NTU NLP

NLP not just for English but for the rest of the world's 6,500 languages



High-resource
100s of millions of documents online, large labelled datasets, large Wikipedia
English, major world languages

Medium-resource
Few labelled data, millions to 100,000s of online documents, parallel data, medium size Wiki
Most European languages, large official languages

Low-resource
No labelled data, few data online, small or no Wikipedia
Most languages in the world

NLP research has mostly focused on these

We need to apply NLP to these

Old paradigm:

- Language-specific NLP
- **Language-specific** feature computation and preprocessing

-	*p.currentSense + p.lemma*
-	*p.currentSense + p.pos*
-	*p.currentSense + a.pos*
-	p_{-1}*.FEAT1*
-	*p.FEAT2*
-	p_1*.FEAT3*
-	*p.semrm.semdprel*
-	*p.lm.dprel*
-	*p.form + p.children.dprel.bag*
	$p.lemma_n$ $(n = -1, 0)$
-	$p.lemma + p.lemma_1$
-	$p.pos_{-1} + p.pos$
-	$p.pos_1$
-	*p.pos + p.children.dprel.bag*

source: (Ruder, 2019)

NLP for Other Languages

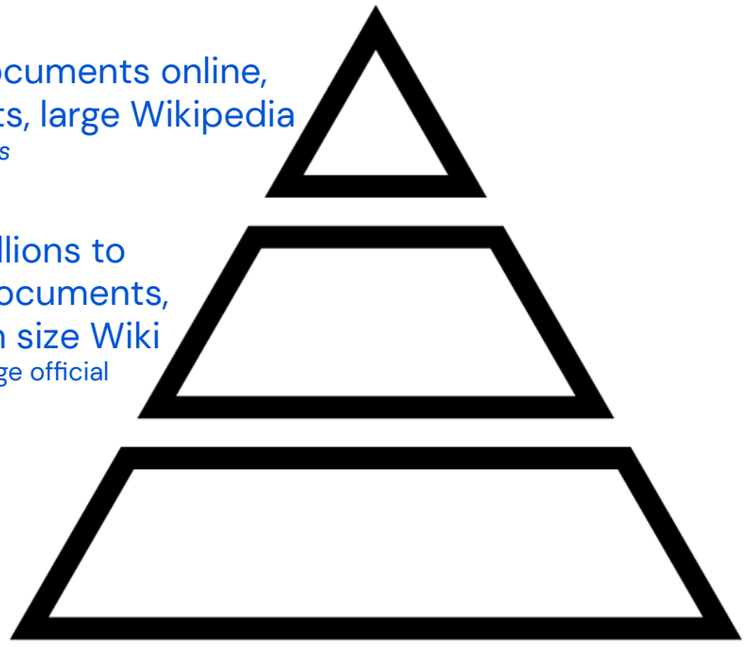
NTU NLP

NLP not just for English but for the rest of the world's 6,500 languages

High-resource
100s of millions of documents online, large labelled datasets, large Wikipedia
English, major world languages

Medium-resource
Few labelled data, millions to 100,000s of online documents, parallel data, medium size Wiki
Most European languages, large official languages

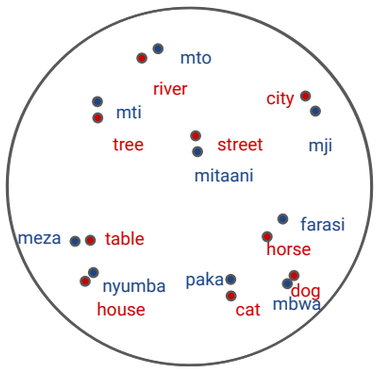
Low-resource
No labelled data, few data online, small or no Wikipedia
Most languages in the world



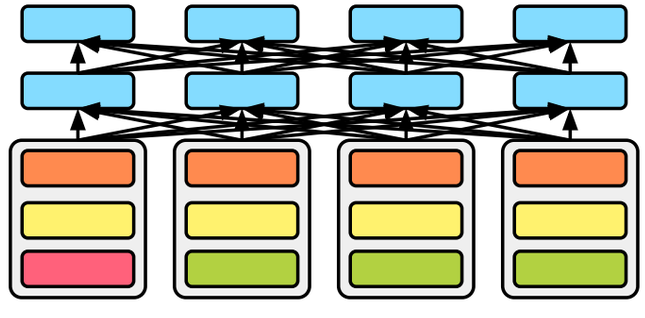
NLP research has mostly focused on these

We need to apply NLP to these

Cross-lingual representations can be learned at different levels: from the word level to the sentence level

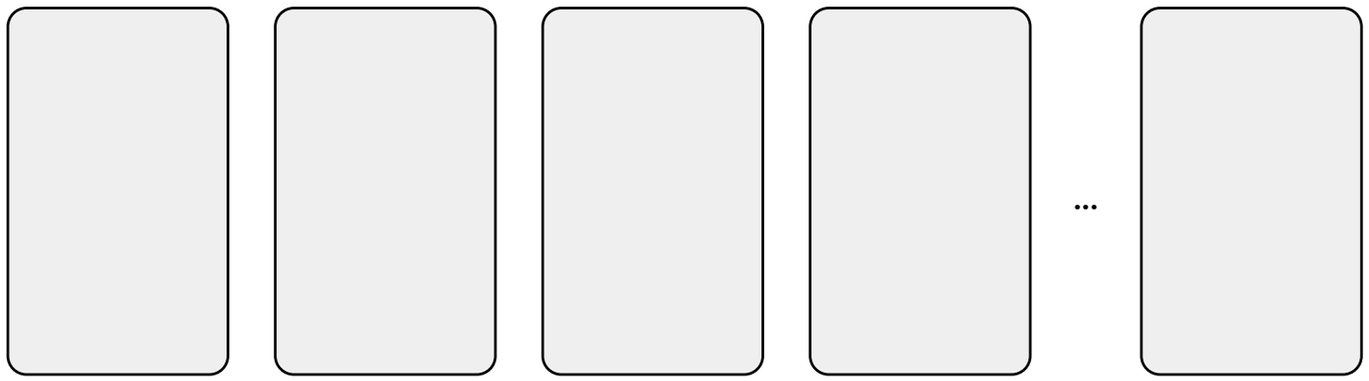


New paradigm: Cross-lingual representation learning

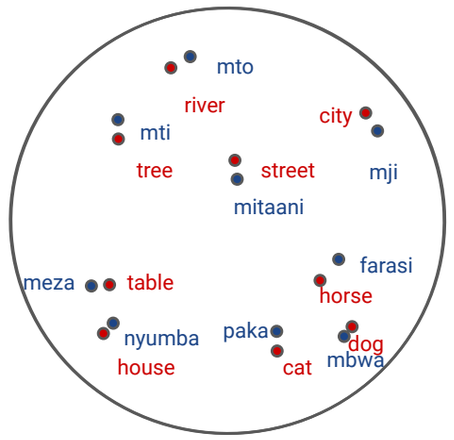


source: (Ruder, 2019)

Cross-lingual Learning Paradigm



- Step 1: Learn cross-lingual representations

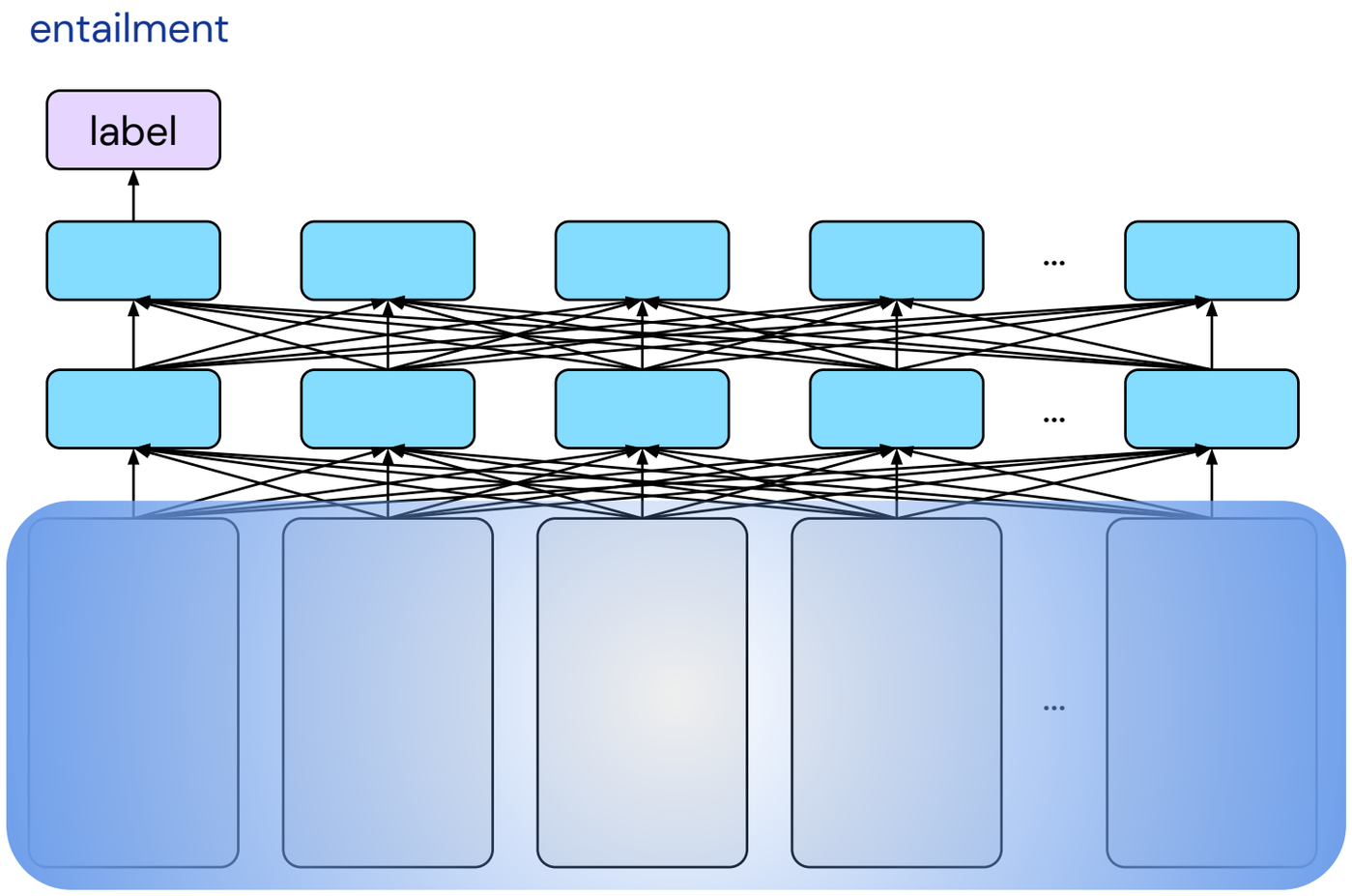


Ongoing work at NTU-NLP

Invited talk next week

source: (Ruder, 2019)

Cross-lingual Learning Paradigm

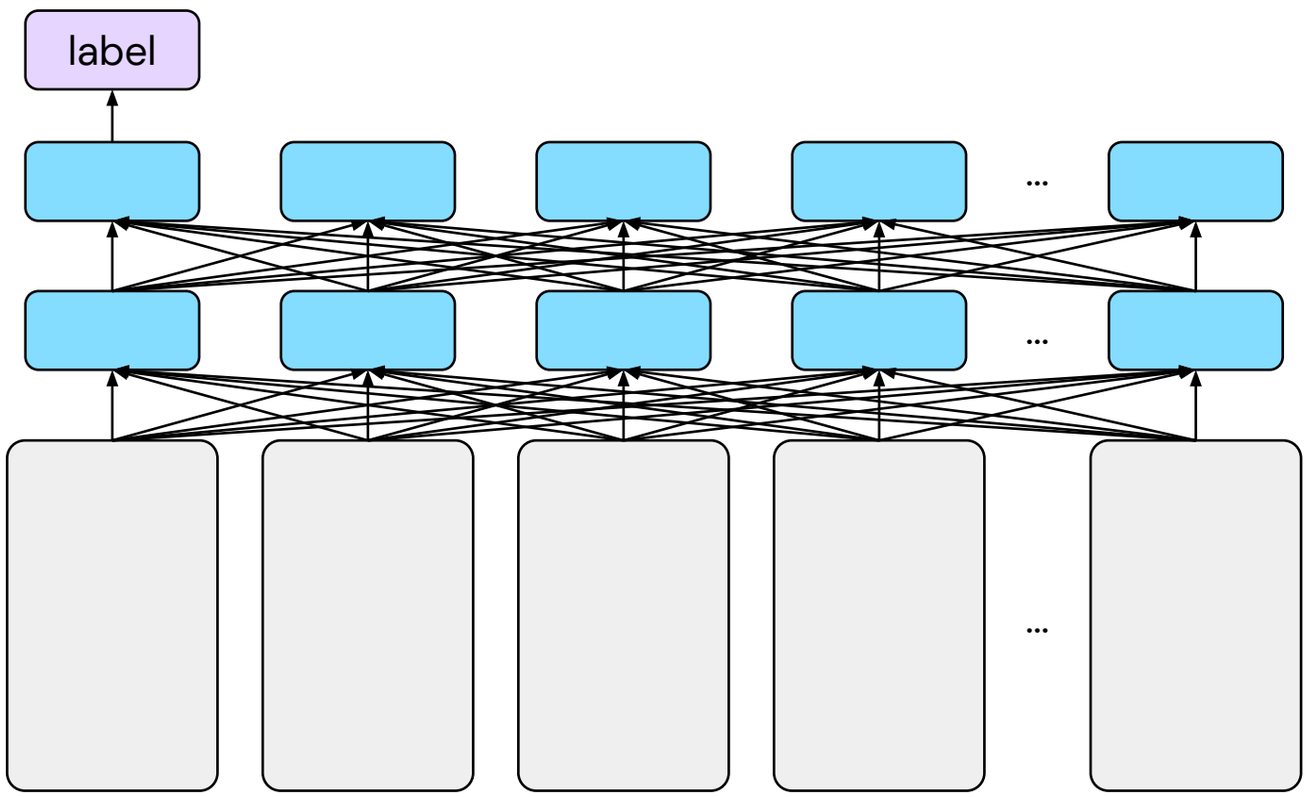


- Step 1: Learn cross-lingual representations
- Step 2: Fine-tune model on labelled data of a high-resource language (mostly English)

  cross-lingual parameters are often kept fixed

source: (Ruder, 2019)

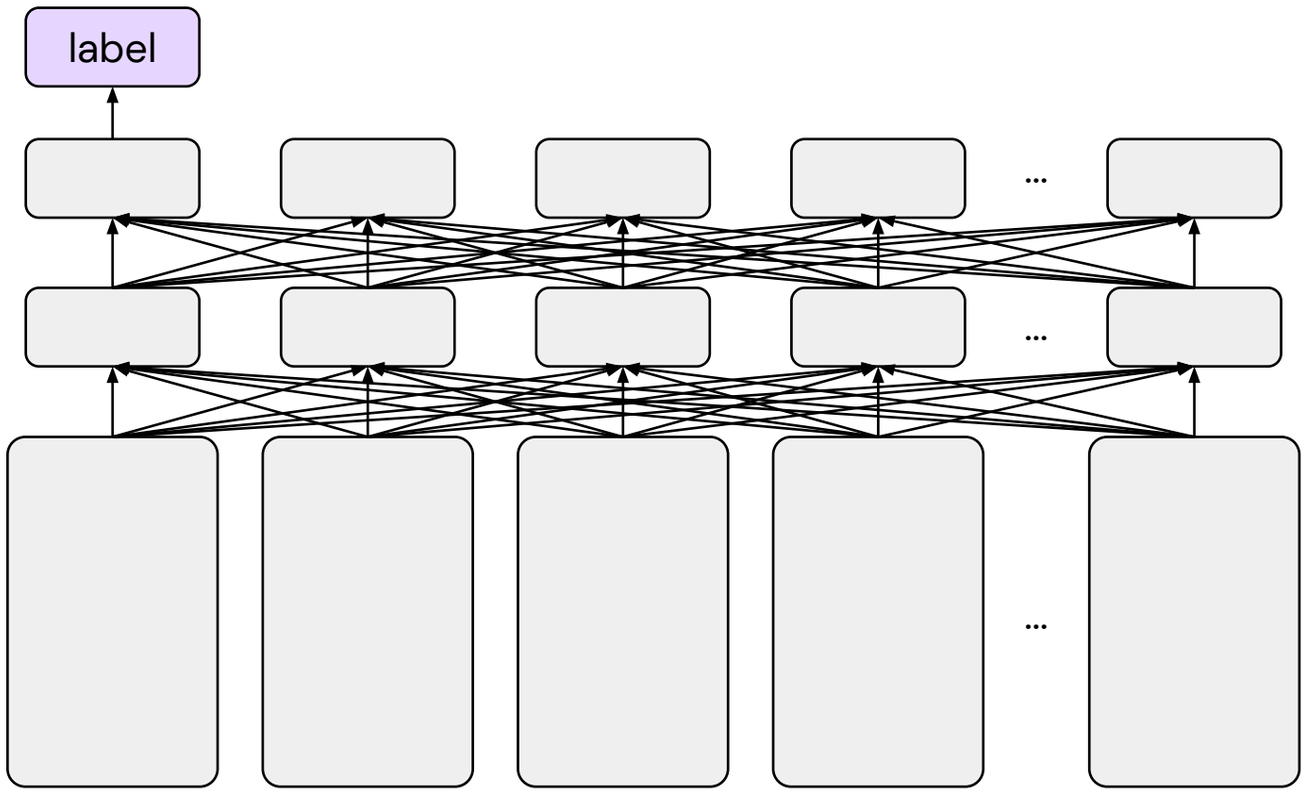
Cross-lingual Learning Paradigm



- Step 1: Learn cross-lingual representations
- Step 2: Fine-tune model on labelled data of a high-resource language
- Step 3: Zero-shot transfer to a low-resource language

source: (Ruder, 2019)

Cross-lingual Learning Paradigm



- Cross-lingual representations can be learned at different levels: from the word level to the sentence level

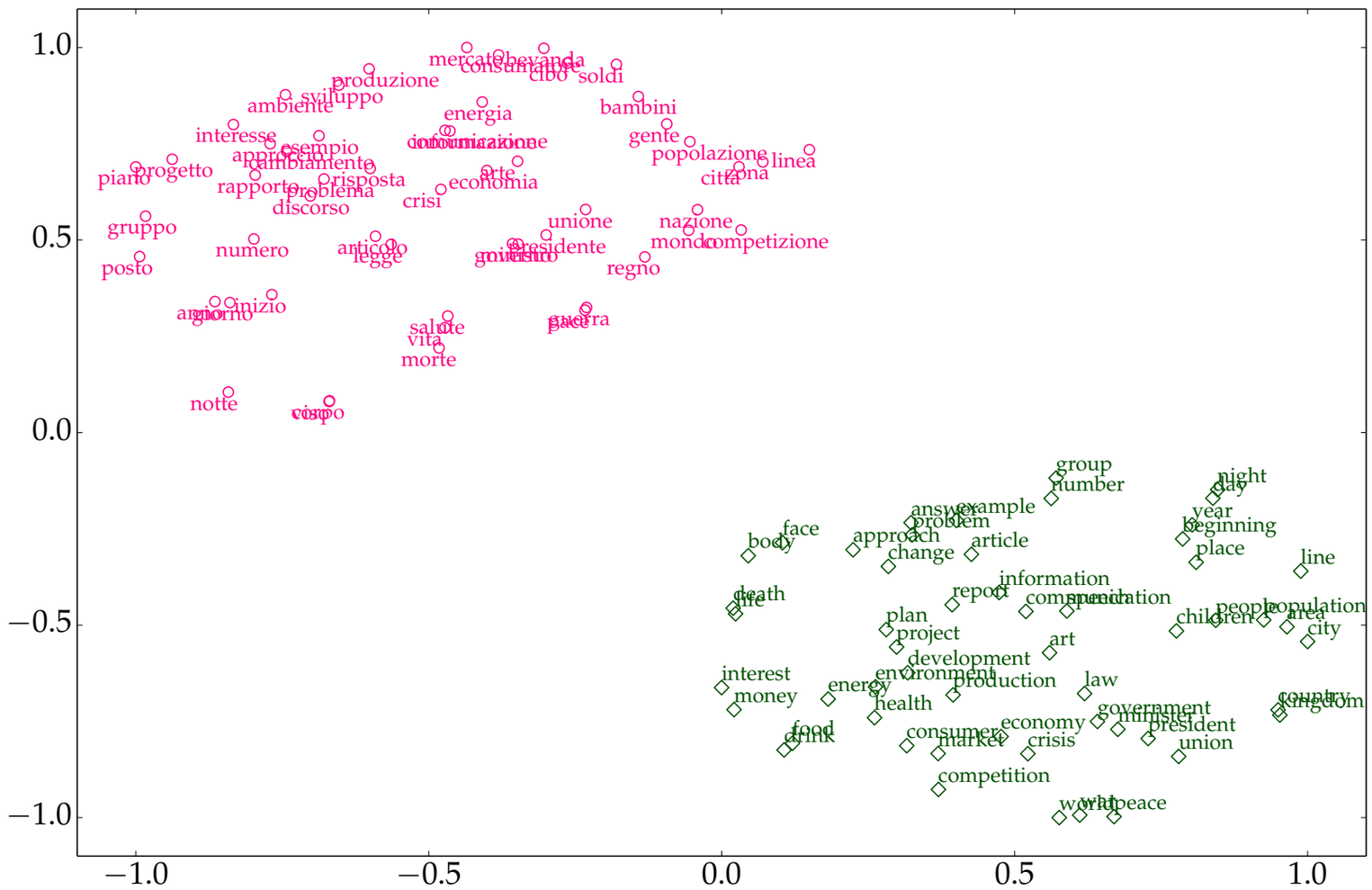
E.g., multi-lingual BERT

Ongoing work at NTU-NLP

source: (Ruder, 2019)

Cross-lingual Word Embeddings

Cross-lingual representations of words in a joint embedding space

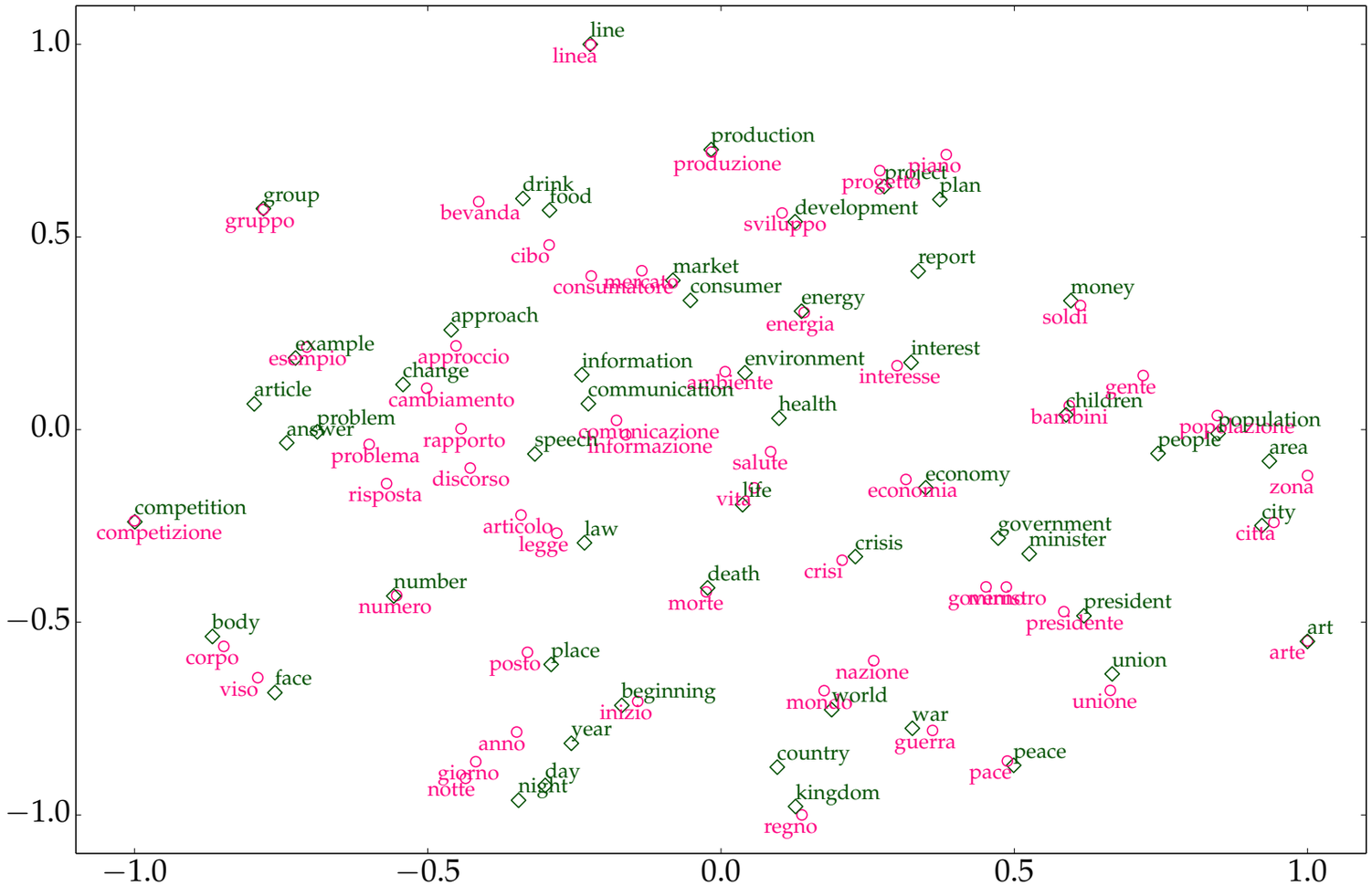


Two monolingual word embeddings (Ruder, 2019)

Cross-lingual Word Embeddings

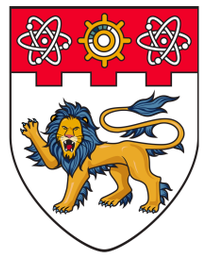
NTU NLP

Cross-lingual representations of words in a joint embedding space



Cross-lingual word embeddings (Ruder et al, 2019)

Cross-lingual Word Embeddings



NTU NLP

Why do we need cross-lingual embeddings?

- Compare word vectors (meaning of words) across languages.
  - Key to bilingual lexicon induction and machine translation
- Transfer of knowledge (semantic & syntactic) between languages.
  - Cross-lingual NER, QA, Sentiment Analysis

Cross-lingual Word Embeddings

What kind of data is used to learn cross-lingual embeddings?

	Parallel	Comparable
Word	Dictionaries	Images
Sentence	Translations	Captions
Document	-	Wikipedia

(a) Word, par. (b) Word, comp. (c) Sentence, par. (d) Sentence, comp. (e) Doc., comp.

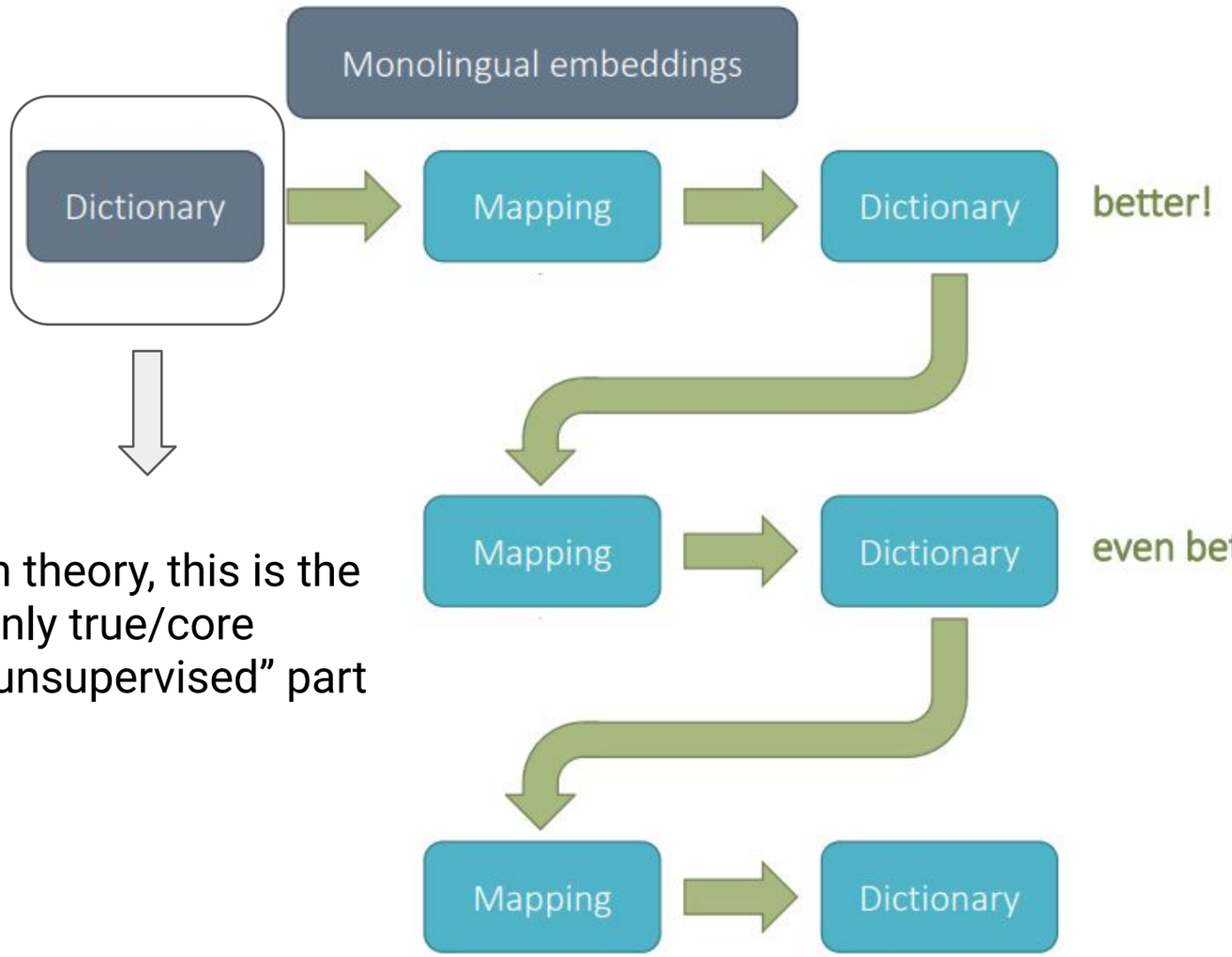
source: (Ruder, 2019)

Cross-lingual Embedding Learning Methods

	Parallel	Comparable
Word —Mapping	Mikolov et al. (2013b) Faruqui and Dyer (2014) Lazaridou et al. (2015) Dinu et al. (2015) Xing et al. (2015) Lu et al. (2015) Vulić and Korhonen (2016) Ammar et al. (2016b) Zhang et al. (2016b, 2017ab) Artexte et al. (2016, 2017, 2018ab) Smith et al. (2017) Hauer et al. (2017) Mrkšić et al. (2017b) Conneau et al. (2018a) Joulin et al. (2018) Alvarez-Melis and Jaakkola (2018) Ruder et al. (2018) Glavaš et al. (2019)	Bergsma and Van Durme (2011) Kiela et al. (2015) Vulić et al. (2016)
Word —Pseudo-bilingual	Xiao and Guo (2014) Gouws and Søgaard (2015) Duong et al. (2016) Adams et al. (2017)	Duong et al. (2015)
Word —Joint	Klementiev et al. (2012) Kočiský et al. (2014)	
Sentence —Matrix factorization	Zou et al. (2013) Shi et al. (2015) Gardner et al. (2015) Guo et al. (2015) Vyas and Carpuat (2016)	
Sentence —Compositional	Hermann and Blunsom (2013, 2014) Soyer et al. (2015)	
Sentence —Autoencoder	Lauly et al. (2013) Chandar et al. (2014)	
Sentence —Skip-gram	Gouws et al. (2015) Luong et al. (2015) Coulmance et al. (2015) Pham et al. (2015)	
Sentence —Other	Levy et al. (2017) Rajendran et al. (2016)	Calixto et al. (2017) Gella et al. (2017)
Document		Vulić and Moens (2013a, 2014, 2016) Søgaard et al. (2015) Mogadala and Rettinger (2016)

Check also more recent work including ours (see suggested readings)

Supervised/Unsupervised Mapping based methods

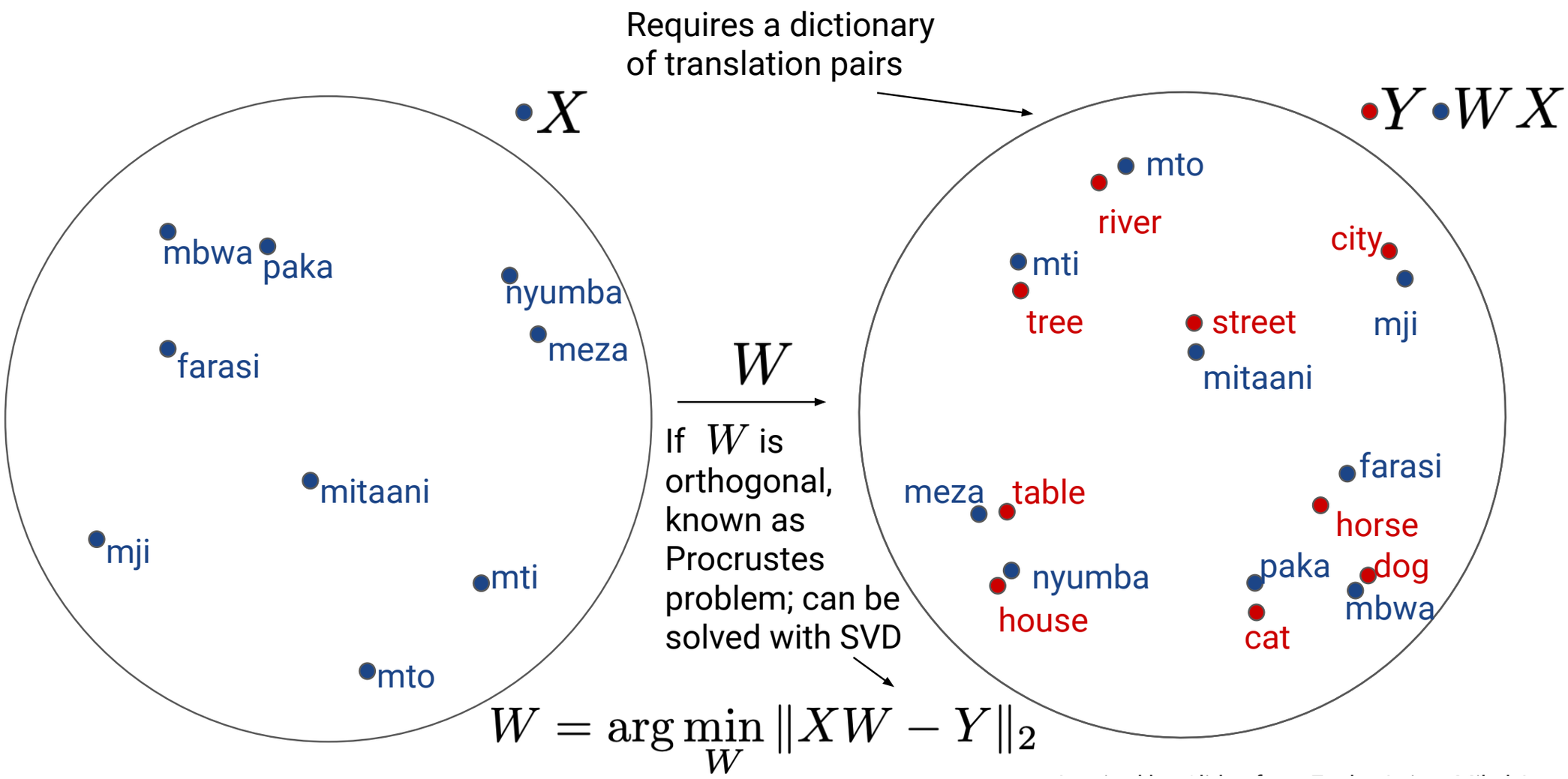


In theory, this is the only true/core “unsupervised” part

- Most current approaches use self-learning
- Particularly useful with limited supervision
- The seed dictionary improves over time
- Initial seed dictionary is either given (supervised) or learned (unsupervised)
- Many ways to learn seed dictionary
- One common approach: adversarial mapping

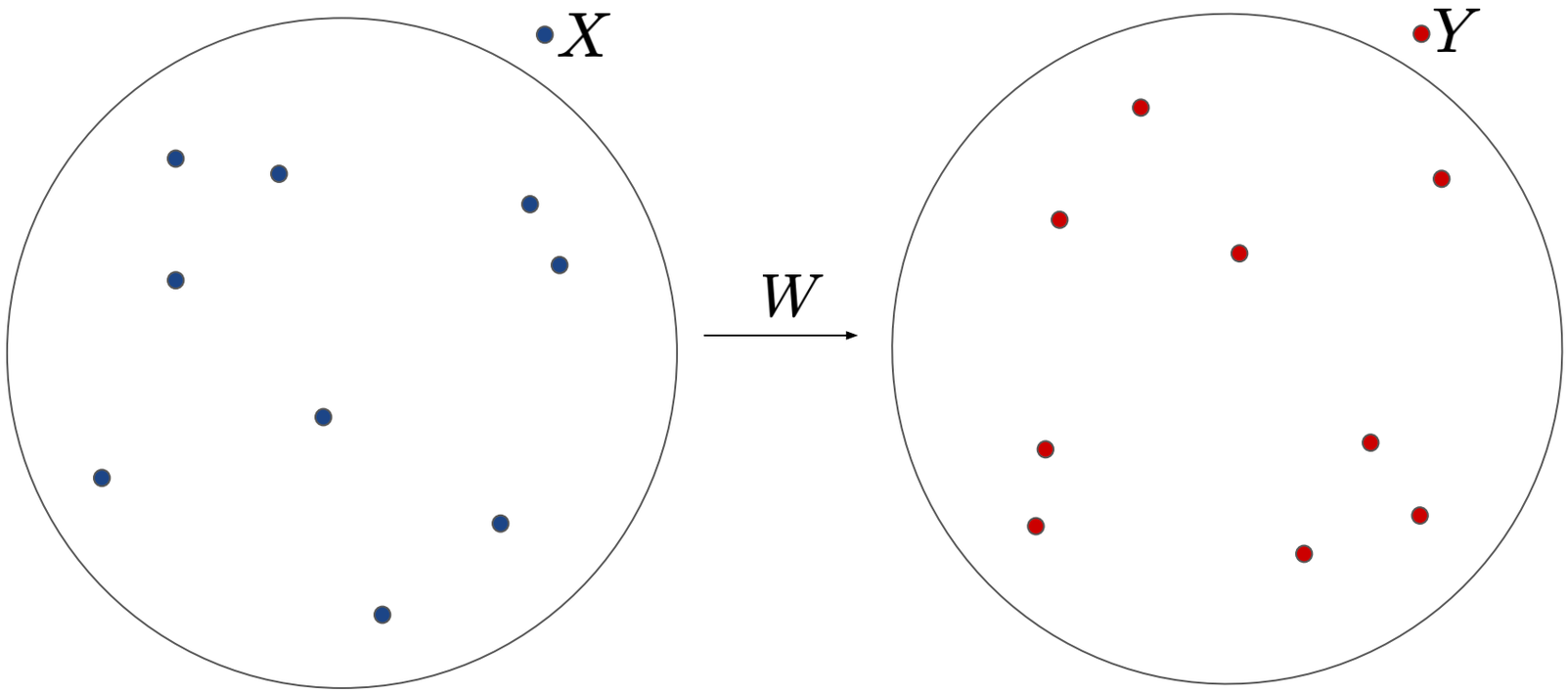
source: (Ruder, 2019)

Supervised Mapping based methods



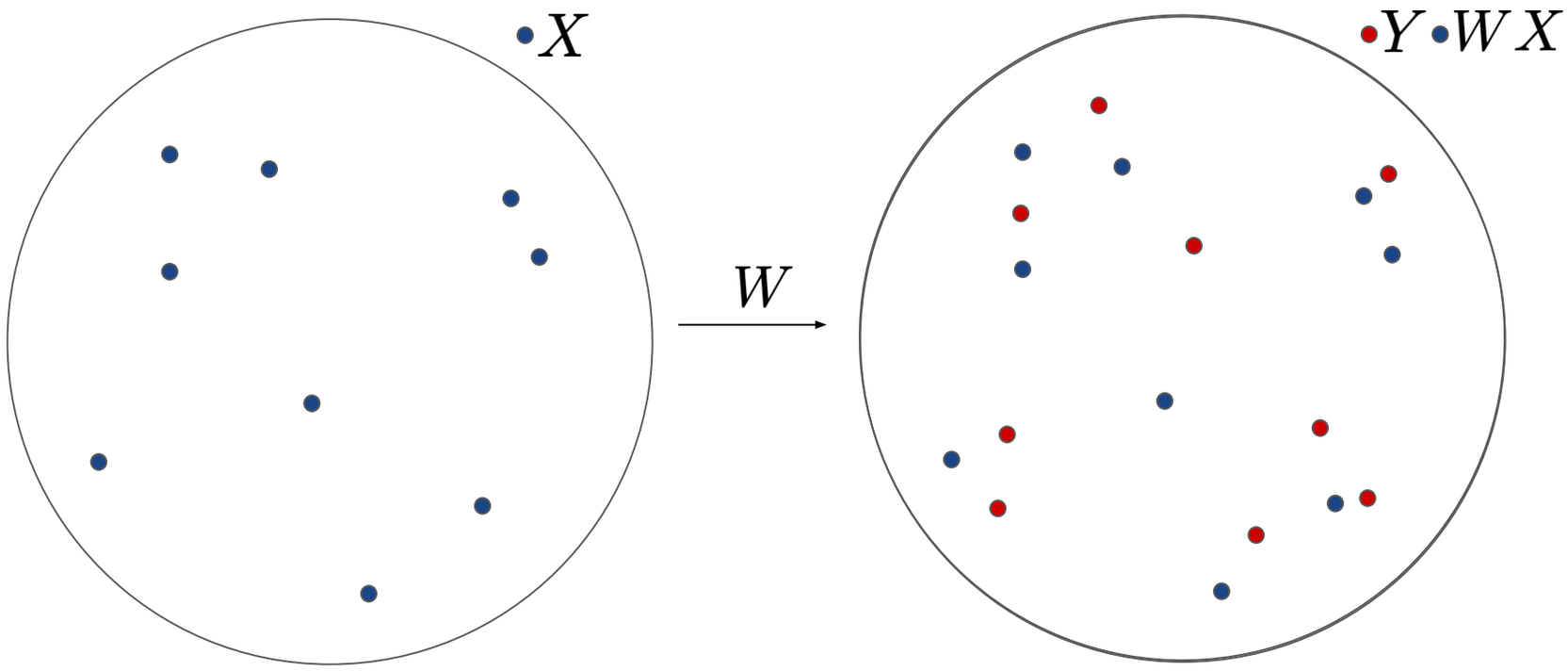
source: (Ruder, 2019)

Unsupervised Mapping



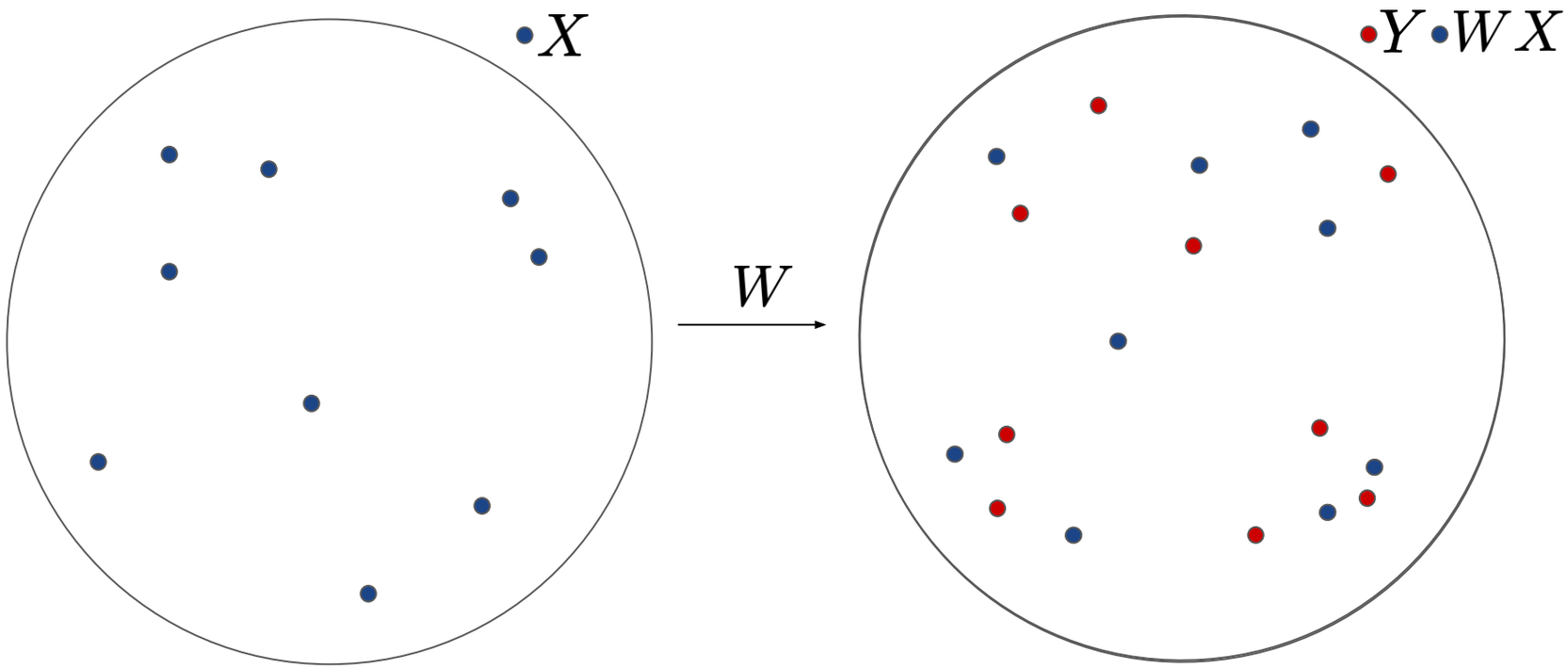
source: (Ruder, 2019)

Unsupervised Mapping



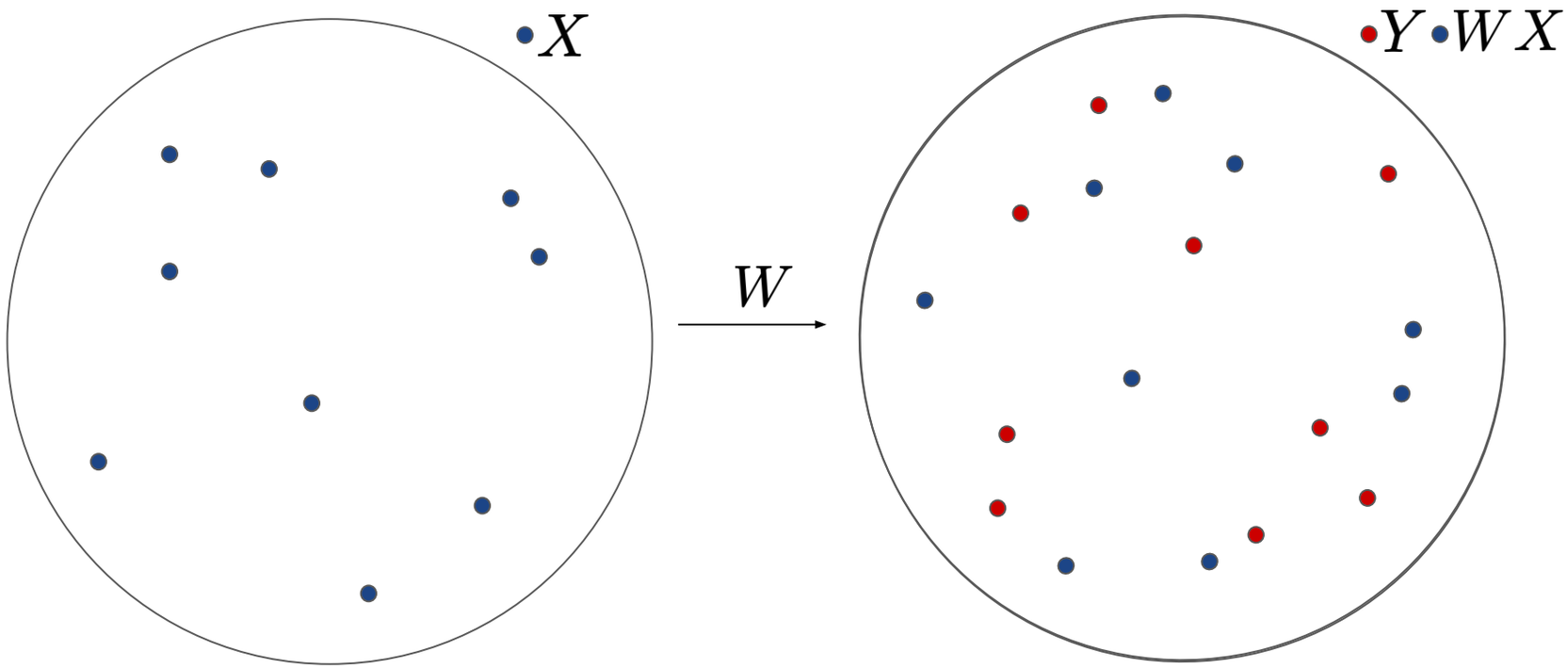
source: (Ruder, 2019)

Unsupervised Mapping



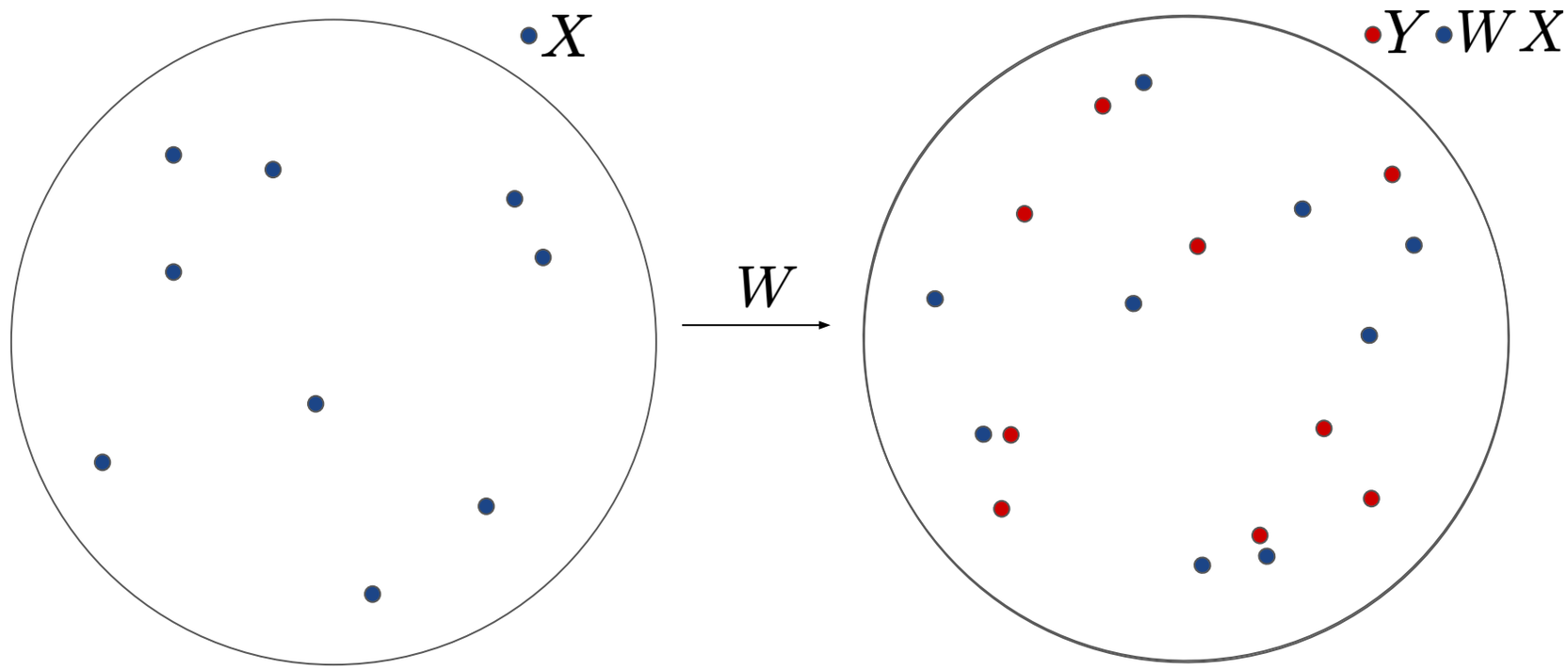
source: (Ruder, 2019)

Unsupervised Mapping



source: (Ruder, 2019)

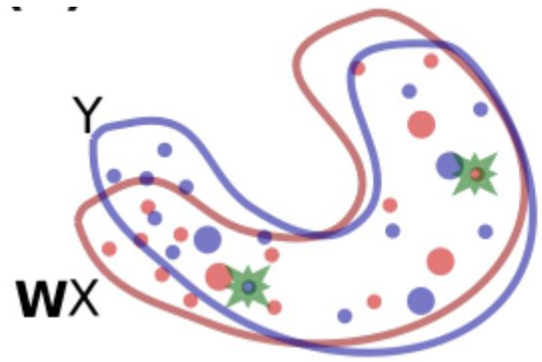
Unsupervised Mapping



source: (Ruder, 2019)

Adversarial Mapping

- **Generator**: the projection matrix W
- **Discriminator**: differentiate between the “fake” projected source samples WX and the “true” target samples Y.



an embedding in X

an embedding in Y

$$\mathcal{L}_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}\left(\text{source}=1\middle|Wx_i\right) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}\left(\text{source}=0\middle|y_i\right)$$

Maximize probability of predicting correct source

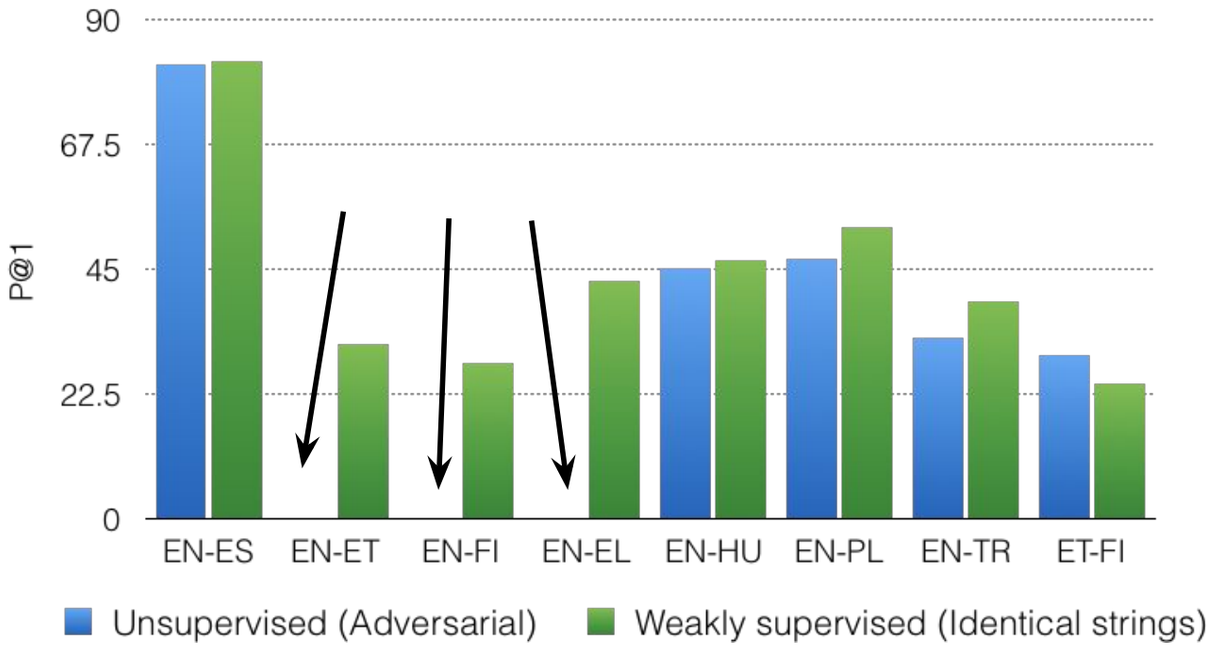
$$\mathcal{L}_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}\left(\text{source}=0\middle|Wx_i\right) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}\left(\text{source}=1\middle|y_i\right)$$

Maximize probability of “fooling” discriminator

For every input sample, generator and discriminator are trained successively with SGD to minimize their losses.

source: (Ruder, 2019)

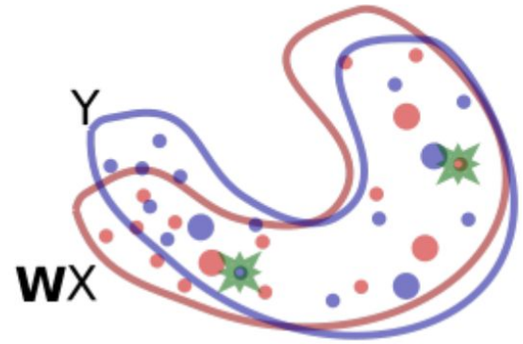
Adversarial Mapping



Søgaard et al. (2018)

- More specifically, monolingual embedding spaces should be ***approximately isomorphic***, i.e. same number of vertices, connected the same way
- **Does not strictly hold** even for related languages

- **Fail to map** between distant languages
  → Why?
- Remember the unsupervised alignment step:

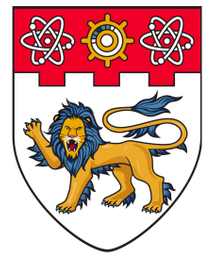


- Embedding spaces need to be **similar** for this to work

Our recent work (ACL, 2020) is independent of this assumption

source: (Ruder, 2019)

Evaluation of Cross Lingual Vectors



NTU NLP

Word Translation /Dictionary Induction (standard/intrinsic task):

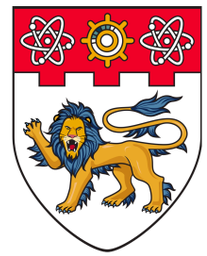
	En-Ms		En-Fi		En-Et		En-Tr		En-El		En-Fa		En-He		En-Ta		En-Bn		En-Hi	
	→	←	→	←	→	←	→	←	→	←	→	←	→	←	→	←	→	←	→	←
GH Distance	0.49		0.54		0.68		0.41		0.46		0.39		0.45		0.29		0.49		0.56	
Unsupervised Baselines																				
Artetxe et al. (2018b)	49.0	49.7	49.8	63.5	33.7	51.2	52.7	63.5	47.6	63.4	33.4	40.7	43.8	57.5	0.0	0.0	18.4	23.9	39.7	48.0
Conneau et al. (2018)	46.2	0.0	38.4	0.0	19.4	0.0	46.4	0.0	39.5	0.0	30.5	0.0	36.8	53.1	0.0	0.0	0.0	0.0	0.0	0.0
Supervision With **"1K Unique"** Seed Dictionary																				
Supervised Baselines																				
Artetxe et al. (2017)	36.5	41.0	40.8	56.0	21.3	39.0	39.5	56.5	34.5	56.2	24.1	35.7	30.2	51.7	5.4	12.7	6.2	19.9	22.6	38.8
Artetxe et al. (2018a)	35.3	34.0	30.8	40.8	21.6	32.6	33.7	43.3	32.0	46.4	22.8	27.6	32.27	39.1	7.3	11.9	11.3	15.7	26.2	30.7
Conneau et al. (2018)	46.2	44.7	46.0	58.4	29.3	40.0	44.8	58.5	42.1	56.5	31.6	38.4	38.3	52.4	11.7	16.0	14.3	19.7	32.5	42.3
Joulin et al. (2018)	31.4	30.7	30.4	41.4	20.1	26.0	30.7	36.5	28.8	43.6	18.7	23.1	33.5	34.3	6.0	10.1	7.6	11.3	20.7	25.7
LNMAP	**50.1**	**50.6**	**54.3**	**64.3**	**39.1**	**51.1**	**52.9**	**64.0**	**48.3**	**62.1**	**35.3**	**41.3**	**45.3**	**54.9**	**18.3**	**24.3**	**19.7**	**30.3**	**36.9**	**49.3**
Supervision With **"5K Unique"** Seed Dictionary																				
Supervised Baselines																				
Artetxe et al. (2017)	36.5	42.0	40.8	57.0	22.4	39.6	39.6	56.7	37.2	56.4	26.0	35.3	31.6	51.9	6.2	13.4	8.2	21.3	23.2	38.3
Artetxe et al. (2018a)	54.6	52.5	48.8	65.2	38.2	54.8	52.0	65.1	47.5	64.6	38.4	42.4	47.4	57.4	18.4	25.79	21.9	31.8	40.3	49.5
Conneau et al. (2018)	46.4	45.7	46.0	59.2	31.0	41.7	45.9	60.1	43.1	56.8	31.6	37.7	38.4	53.4	14.27	19.1	15.0	22.6	32.9	42.8
Joulin et al. (2018)	50.0	49.3	53.0	66.1	39.8	52.0	**54.0**	61.7	47.6	63.4	**39.6**	42.2	**53.0**	56.3	16.0	24.25	21.3	27.0	38.3	47.5
LNMAP	**50.2**	**54.6**	**54.7**	**69.2**	**41.8**	**57.2**	52.0	**66.6**	**49.7**	**65.5**	36.8	**44.5**	47.3	**58.7**	**20.7**	**30.7**	**22.2**	**36.2**	**39.1**	**51.6**
Supervision With **"Whole"** ("5K Unique" Source Words) Seed Dictionary																				
Supervised Baselines																				
Artetxe et al. (2017)	37.0	41.6	40.8	57.0	22.7	39.5	38.8	56.9	37.5	57.2	25.4	36.3	32.2	52.1	5.9	14.1	7.7	21.7	22.4	38.3
Artetxe et al. (2018a)	55.2	51.7	48.9	64.6	37.4	54.0	52.2	63.7	48.2	65.0	39.0	42.6	47.6	58.0	19.6	25.2	21.1	30.6	**40.4**	50.0
Conneau et al. (2018)	46.3	44.8	46.4	59.0	30.9	42.0	45.8	59.0	44.4	57.4	31.8	38.8	39.0	53.4	15.1	18.4	15.5	22.4	32.9	44.4
Joulin et al. (2018)	**51.4**	49.1	**55.6**	65.8	40.0	50.2	**53.8**	61.7	**49.1**	62.8	**40.5**	42.4	**52.2**	57.9	17.7	24.0	20.2	26.9	38.2	47.1
LNMAP	50.4	**56.0**	55.1	**71.5**	**41.9**	**57.5**	52.6	**66.3**	**49.1**	**67.1**	36.5	**43.7**	47.4	**61.0**	**19.9**	**32.0**	**22.5**	**36.6**	38.7	**52.7**

Table 1: Translation accuracy (P@1) on **low-resource** languages on **MUSE dataset** using fastText embeddings.

MUSE: https://github.com/facebookresearch/MUSE

source: (Mohiuddin et al, 2020)

Evaluation of Cross Lingual Vectors



NTU NLP

Extrinsic tasks: Cross-lingual Natural Language Inference (XNLI)

Supervised	Dict	EN–DE	EN–FR	EN–TR	EN–RU	Avg
PROC	1K	0.561	0.504	0.534	0.544	0.536
PROC	5K	0.607	0.534	0.568	0.585	**0.574**
PROC-B	1K	0.613	0.543	0.568	0.593	**0.579**
PROC-B	3K	0.615	0.532	0.573	0.599	**0.580**
DLV	5K	0.614	0.556	0.536	0.579	**0.571**
RCSLS	1K	0.376	0.357	0.387	0.378	0.374
RCSLS	5K	0.390	0.363	0.387	0.399	0.385
Unsupervised						
VECMAP		0.604	0.613	0.534	0.574	**0.581**
MUSE		0.611	0.536	0.359*	0.363*	0.467
ICP		0.580	0.510	0.400*	0.572	0.516
GWA		0.427*	0.383*	0.359*	0.376*	0.386

Other Extrinsic tasks:

- Cross-lingual Document Classification
- Cross-lingual Information Retrieval
- Cross-lingual Paraphrase Identification
- Cross-lingual Question Answering

source: (Gravas et al, 2019)

Dealing with Multiple Senses

- Most words have more than one meaning (Polysemy)
  - Especially common words
  - Especially words that have existed for a long time

	Domain	%		Domain	%
noun	biology	29.3	**verb**	factotum	67.0
	factotum	20.7		psychology	3.5
	art	6.2		sport	2.9
	sport	3.1		art	2.5
	medicine	3.1		biology	2.5
	other	37.6		*other*	21.6
adjective	factotum	67.8	**adverb**	factotum	81.4
	biology	6.5		psychology	7.5
	art	3.2		art	1.8
	psychology	2.7		physics	1.1
	physics	1.9		economy	1.1
	other	17.9		other	7.1

McCarthy et al, 2007

One vector for all senses?

Or

One vector for each sense?

Dealing with Multiple Senses

One vector for all senses?

Different senses of a word reside in a linear superposition (weighted sum) in standard word embeddings like word2vec/glove

$$v_{tie} \approx \alpha_1 v_{tie1} + \alpha_2 v_{tie2} + \alpha_3 v_{tie3} + \cdots$$

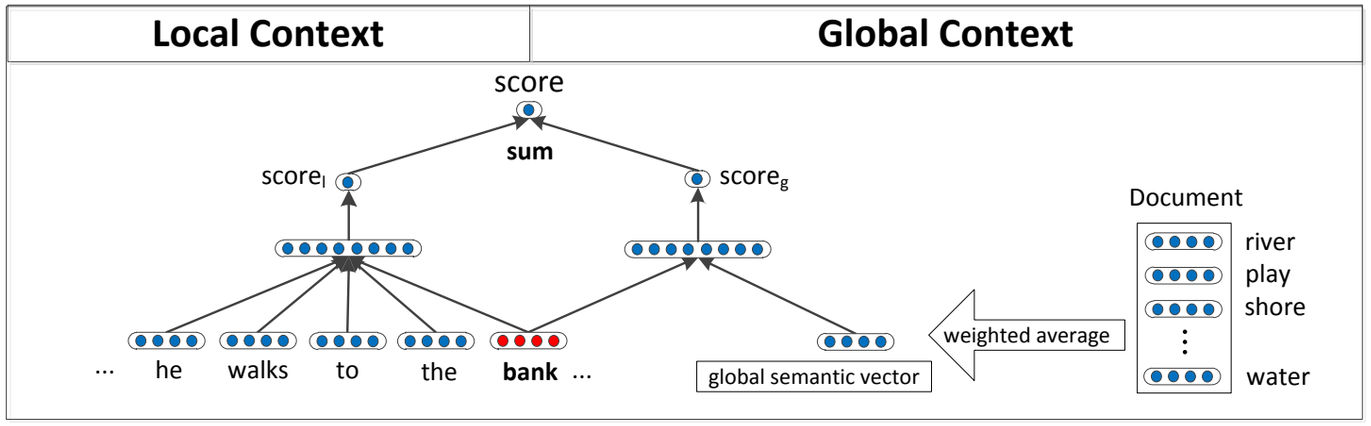
Sparse coding can recover vectors that approximately capture the senses.

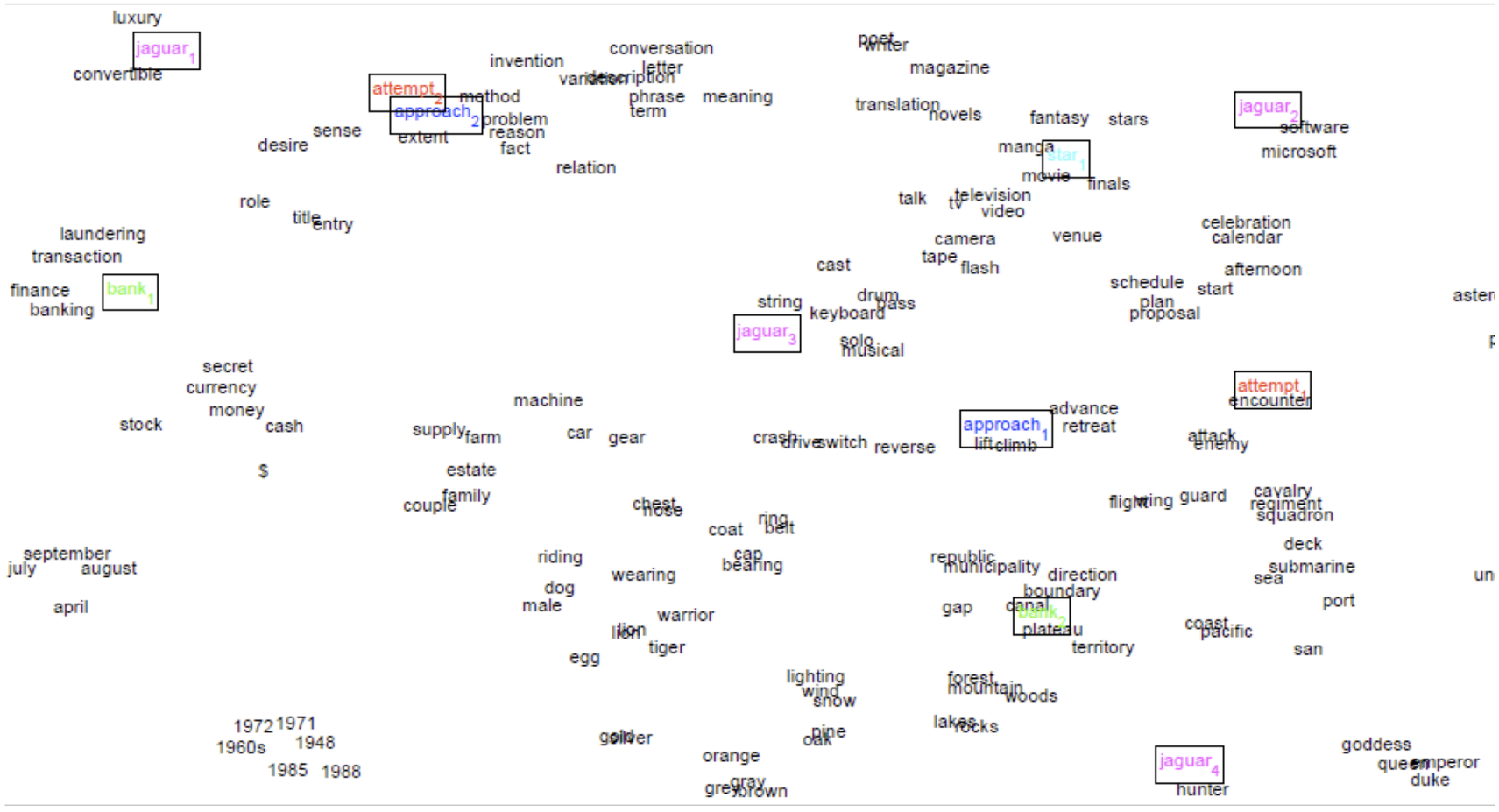
tie					spring				
trousers	season	scoreline	wires	operatic	beginning	dampers	flower	creek	humid
blouse	teams	goalless	cables	soprano	until	brakes	flowers	brook	winters
waistcoat	winning	equaliser	wiring	mezzo	months	suspension	flowering	river	summers
skirt	league	clinching	electrical	contralto	earlier	absorbers	fragrant	fork	ppen
sleeved	finished	scoreless	wire	baritone	year	wheels	lilies	piney	warm
pants	championship	replay	cable	coloratura	last	damper	flowered	elk	temperatures

Arora et al, 2018

Dealing with Multiple Senses

One vector for each sense





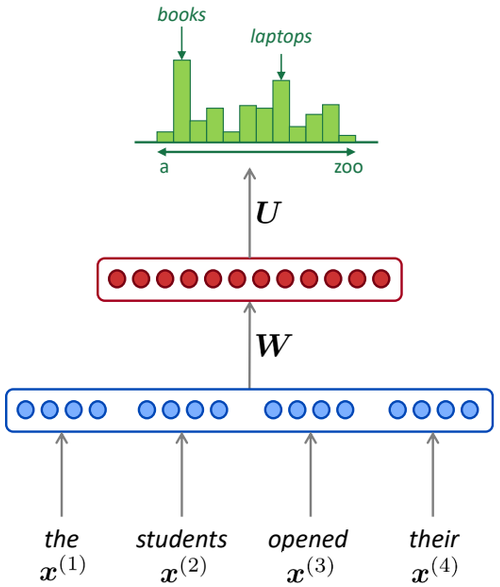
Huang et al. 2012

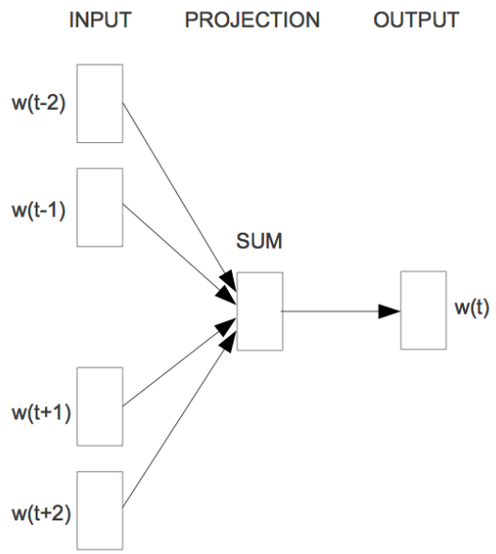
Recent: Contextual representations (ELMo, BERT)

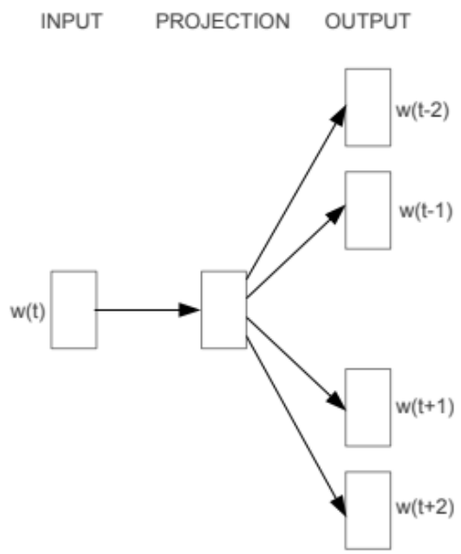
Summary

NTU NLP

Word2Vec







Negative Sampling

Skipgram

shalt	not	make	a	machine

input	output
make	shalt
make	not
make	a
make	machine

Negative Sampling

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0

Word embedding evaluation

$$a - a^* = b - b^*$$

X-lingual embeddings