COMP0087 25/26
Lecture 2: Word Embeddings

16/01/2026

How We Learn the Meaning of Words

Simca

How We Learn the Meaning of Words

- The mechanic said he hadn't seen a **Simca** in his garage since the 1970s.
- The **Simca** was parked next to a Renault at a vintage show.
- My grandmother learned to drive in a tiny blue **Simca**.

How We Learn the Meaning of Words

- The mechanic said he hadn't seen a **Simca** in his garage since the 1970s.
- The **Simca** was parked next to a Renault at a vintage show.
- My grandmother learned to drive in a tiny blue **Simca**.

How We Learn the Meaning of Words

- The mechanic said he hadn't seen a **Simca** in his garage since the 1970s.
- The **Simca** was parked next to a Renault at a vintage show.
- My grandmother learned to drive in a tiny blue **Simca**.

How We Learn the Meaning of Words

UCL

University College London

Machine view

UCL = [52, 3218]

University College London
=
[31272, 9304, 7295]

Tiktoken tokenizer demo: https://platform.openai.com/tokenizer

"You shall know a word by the company it keeps"

– J.R. Firth (1957)

How Machines Learn the Meaning of Words

- The king sat on the throne.
- The queen sat on the throne.
- The cat sat on the mat.
- The dog sat on the mat.

How Machines Learn the Meaning of Words

- king = (the, sat, on, the, throne, .)
- queen = (the, sat, on, the, throne, .)
- cat = (the, sat, on, the, mat, .)
- dog = (the, sat, on, the, mat, .)

How Machines Learn the Meaning of Words

- sim(queen, king) = 6/6 = 100%
- sim(queen, cat) = 5/6 = 83.3%

Can we get better estimation?

How Machines Learn the Meaning of Words

Increase the dataset diversity

queen = (the, sat, on, the, throne, ., crown, royal)

cat = (the, sat, on, the, mat, ., pet, fur)

- similarity(queen, cat) < 50%

From Human Intuition to Machine Representation

A computer doesn't "understand" context. It needs numbers.

Goal: Turn each word into a vector of numbers. The vectors capture semantic relationships.

A Naive Approach: One-Hot Encoding

Vocabulary: [cat, dog, king, queen, man, woman]

cat = [1, 0, 0, 0, 0, 0]

dog = [0, 1, 0, 0, 0, 0]

king = [0, 0, 1, 0, 0, 0]

Problem

king = [0, 0, 1, 0, 0, 0]

queen = [0, 0, 0, 1, 0, 0]

woman = [0, 0, 0, 0, 0, 1]

similarity(queen, woman) = similarity(king, woman)

Goal: Turn each word into a vector of numbers. The vectors capture ~~semantic~~ relationships.

Learn from human

"You shall know a word by the company it keeps"

Use context information

Context window size: 2

The queen sat on the throne, wearing a crown.

throne = (on, the, `,` , wearing)

Context window size:4

The queen sat on the throne, wearing a crown.

throne = (queen, sat, on, the, `,` , wearing, a, crown)

Use context information

The queen sat on the throne, wearing a crown.

the = (sat, on, throne, `,`)

The queen sat on the throne, wearing a crown.

on = (queen, sat, on, the, throne)

...

Co-occurrence Matrix

	Queen	Throne	Crown	Female	Dog	King	Woman	Man
Queen	-	8	7	5	1	9	4	2
Throne	8	-	3	1	0	8	1	1
Crown	7	3	-	1	0	7	1	1
Female	5	1	1	-	2	1	8	2
Dog	1	0	0	2	-	1	2	3
King	9	8	7	1	1	-	2	4
Woman	4	1	1	8	2	2	-	7
Man	2	1	1	2	3	4	7	-

Word/Vector Similarity

Queen =[1, 8, 7, 5, 1, 9, 4, 2] Woman = [4, 1, 1, 8, 2, 2, 1, 7]

Man = [2, 1, 1, 2, 3, 4, 7, 1]

Dot product between two vectors $\vec{u} \cdot \vec{v} = \sum_{i}^{n} u_i * v_i$

Similarity(queen, woman) = 97 Similarity(queen, man) = 96

Word/Vector Similarity

Queen =[1, 8, 7, 5, 1, 9, 4, 2] Woman = [4, 1, 1, 8, 2, 2, 1, 7]

Man = [2, 1, 1, 2, 3, 4, 7, 1]

$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{||\vec{u}|| \cdot ||\vec{v}||} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}}$$

Similarity(queen, woman) = 0.53 Similarity(queen, man) = 0.67

Dominant Common Words Problem

Issue: High-frequency words ("the," "a," "is," "of") dominate co-occurrence counts

Why it's problematic:

- Common words co-occur with nearly everything
- Obscures meaningful semantic relationships
- "doctor" and "banana" both appear near "the"

Sparsity of Co-occurrence Matrix

Vocabulary size: Real vocabularies have 100,000+ words

Matrix size: 100,000 × 100,000 = 10 billion entries

Sparsity: >99.9% of entries are zero

Computational inefficiency

The generalization problem

Issue: Co-occurrence statistics only capture what was directly observed

cat sits on mat

dog runs in park

If "cat" and "dog" never co-occur directly, model misses that both are animals/pets.

The generalization problem

Issue: Co-occurrence statistics only capture what was directly observed

cat sits on mat

dog runs in park

Similar words should have similar representations even without direct co-occurrence evidence

To summarise

Pros:

- Captures basic semantic relationships from raw text statistics
- Simple and explainable

Cons:

- Common words dominate meaningful signals
- Huge and sparse matrices
- Cannot generalize beyond directly observed co-occurrences

Real-World Example: Larger Scale Patterns

Will scaling laws also apply to counting-based methods?

Hypothesis: If we have good data, can we get reasonable performance?

Action plan: Develop a codebase to build vocabulary (try BPE maybe?), then count, then perform similarity analysis.

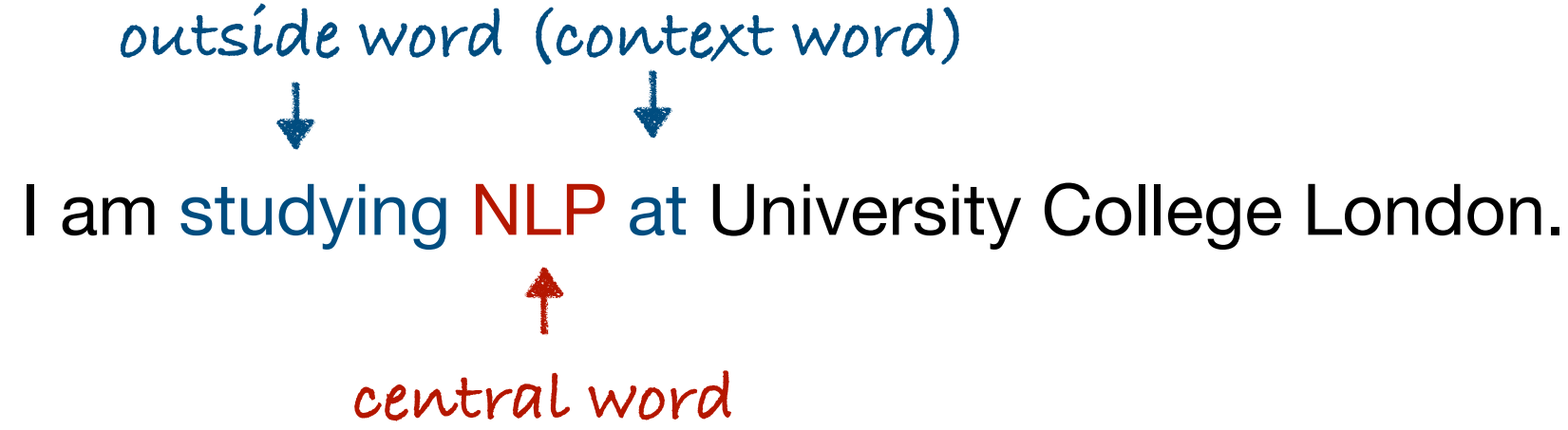
Starting point: Data? (fineweb-edu-subset), Compute budget? (10-minute laptop run?), Code? (AI: yes!), evaluation (later today), and a fancy curve (dataset size ~ evaluation score)

From Counting to Prediction

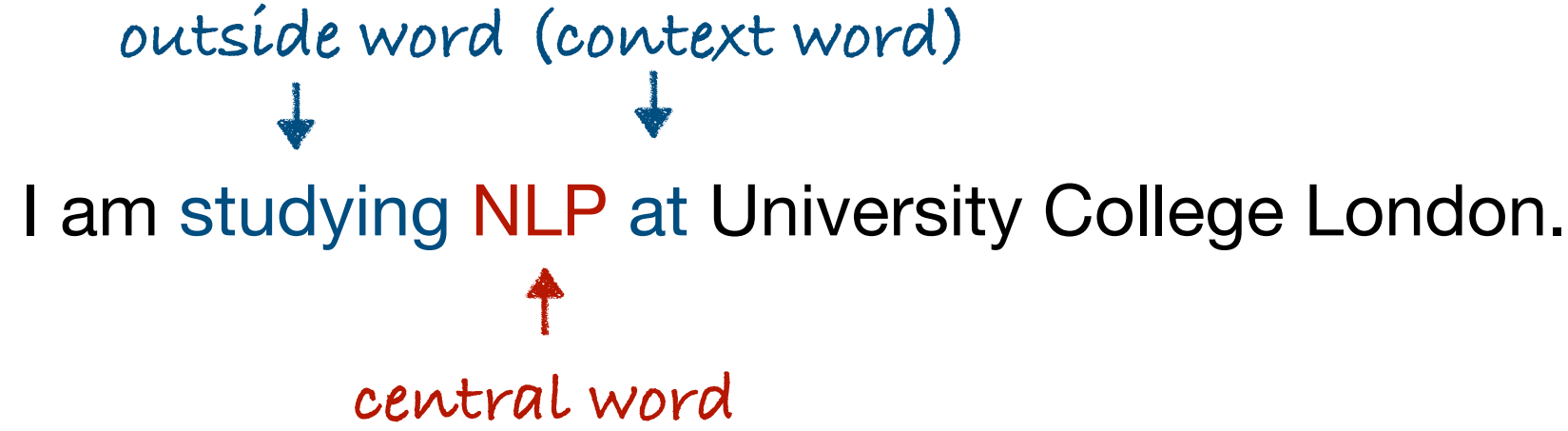
I am studying NLP at University College London.

central word

From Counting to Prediction

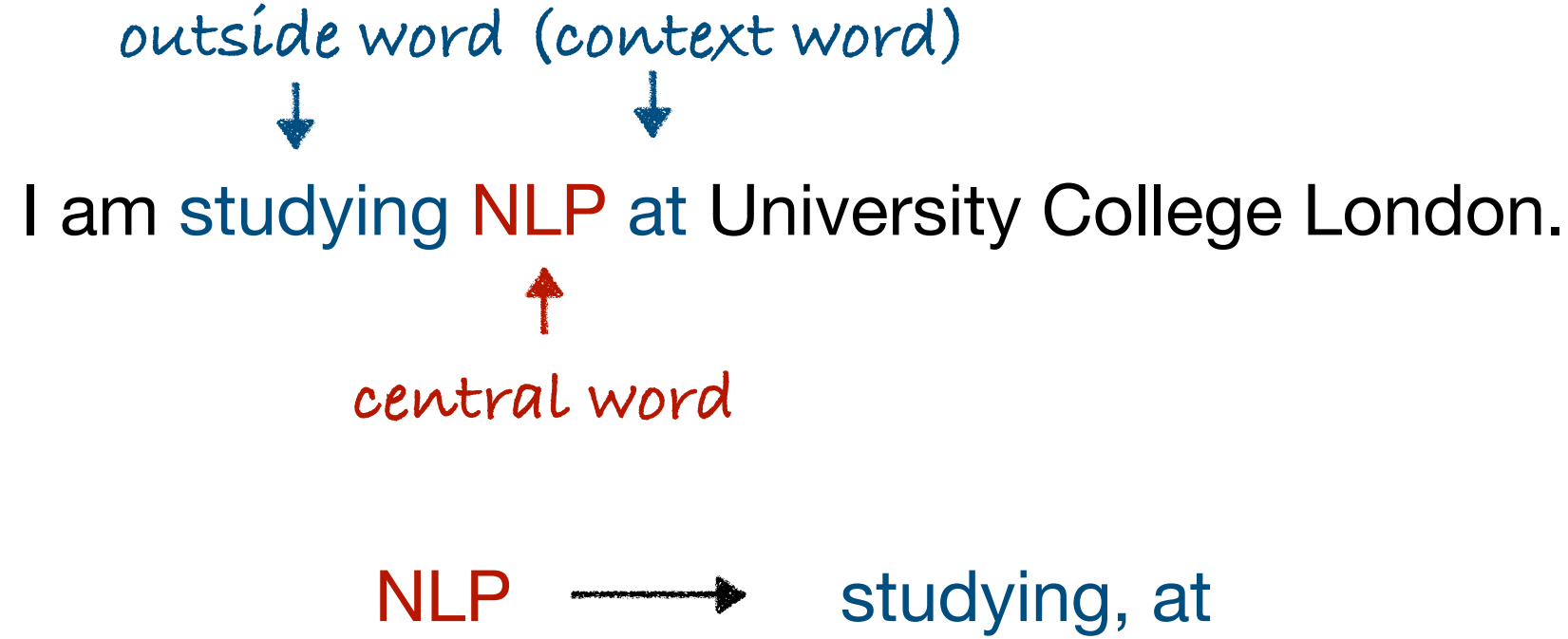


From Counting to Prediction

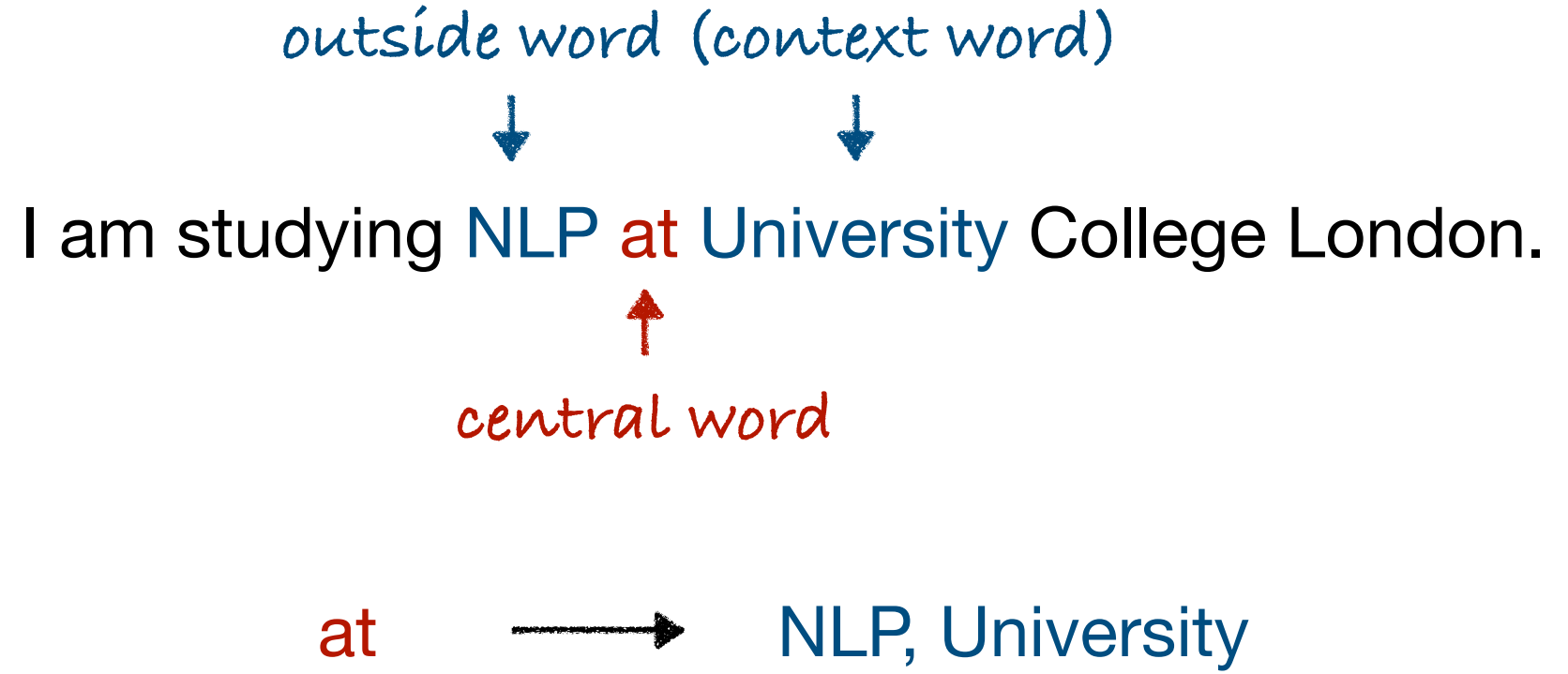


Meaning of NLP = relevant to context

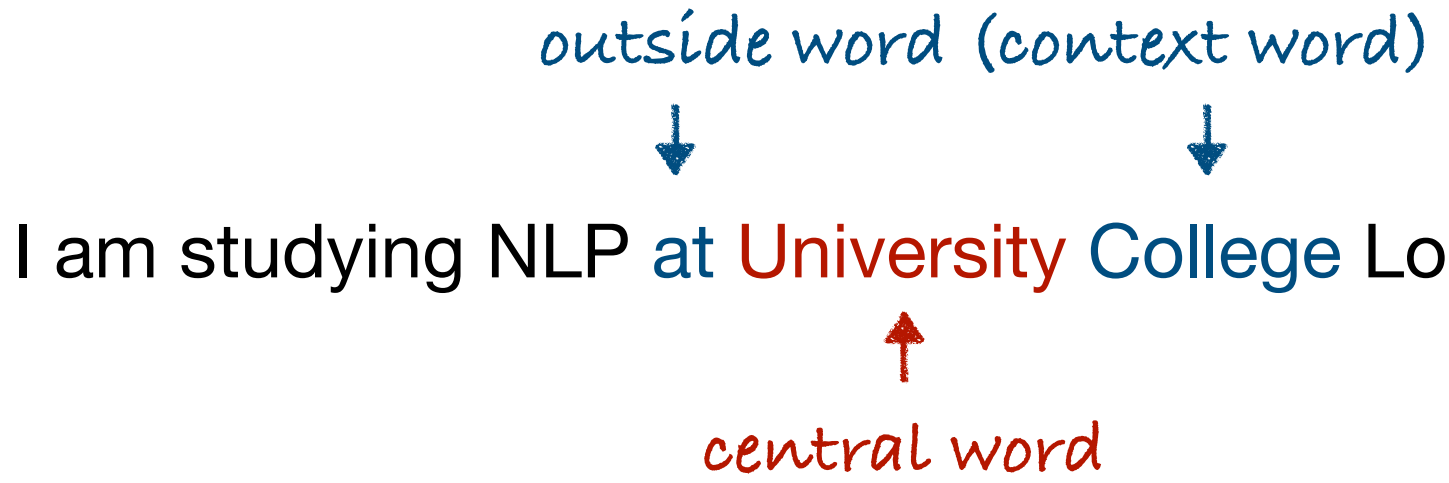
From Counting to Prediction



From Counting to Prediction

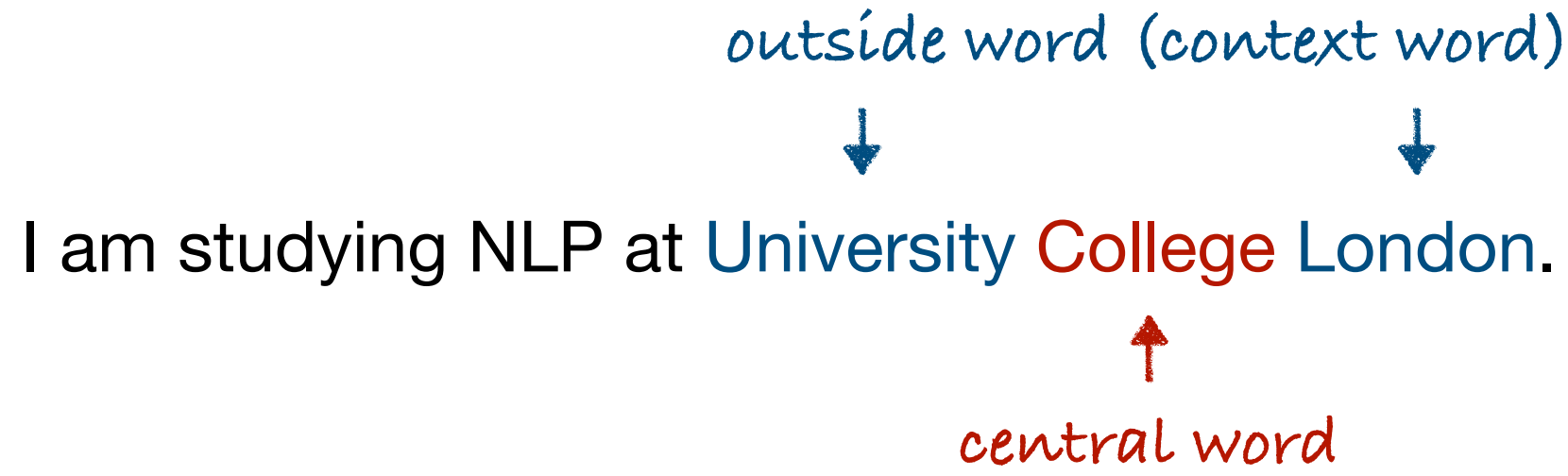


From Counting to Prediction



University → at, College

From Counting to Prediction



From Counting to Prediction

Our idea: Learn word vectors by teaching them to predict contexts.

Predicting context words

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$

I am studying NLP at University College London.

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

Learning Objective

For each position $1, \ldots, T$, in a text corpus, Word2Vec predicts context words within a m-sized window given the central word w_t

$$\text{Likelihood} = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t, \theta)$$

Learning Objective

For each position $1, \ldots, T$, in a text corpus, Word2Vec predicts context words within a m-sized window given the central word w_t

$$\text{Loss} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t, \theta)$$

Learning Objective

$$\text{Loss} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P(w_{t+j}\,|\,w_t,\theta)$$

I am studying NLP at University College London.

t=4

Learning Objective

$$\text{Loss} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log P(w_{t+j}\,|\,w_t, \theta)$$

I am studying NLP at University College London.

t=5

Learning Objective

$$\text{Loss} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P(w_{t+j}\,|\,w_t,\theta)$$

I am studying NLP at University College London.

t=6

Learning Objective

$$\text{Loss} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P(w_{t+j}\,|\,w_t,\theta)$$

I am studying NLP at University College London.

$$\sum_{-m\le j\le m, j\neq 0} logP(\ldots) = logP(w_2\,|\,w_4) + logP(w_3\,|\,w_4) + logP(w_5\,|\,w_4) + logP(w_6\,|\,w_4)$$

P(am|NLP) P(studying|NLP) P(at|NLP) P(University|NLP)

Learning Objective

I am studying NLP at University College London.

$$\text{Loss} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log P(w_{t+j}\,|\,w_t,\theta)$$

central word index

sliding context prediction

How to calculate probability?

I am studying NLP at University College London.

P(University|NLP)

central vector v

outside vector u

$$\text{score} = u^T v$$

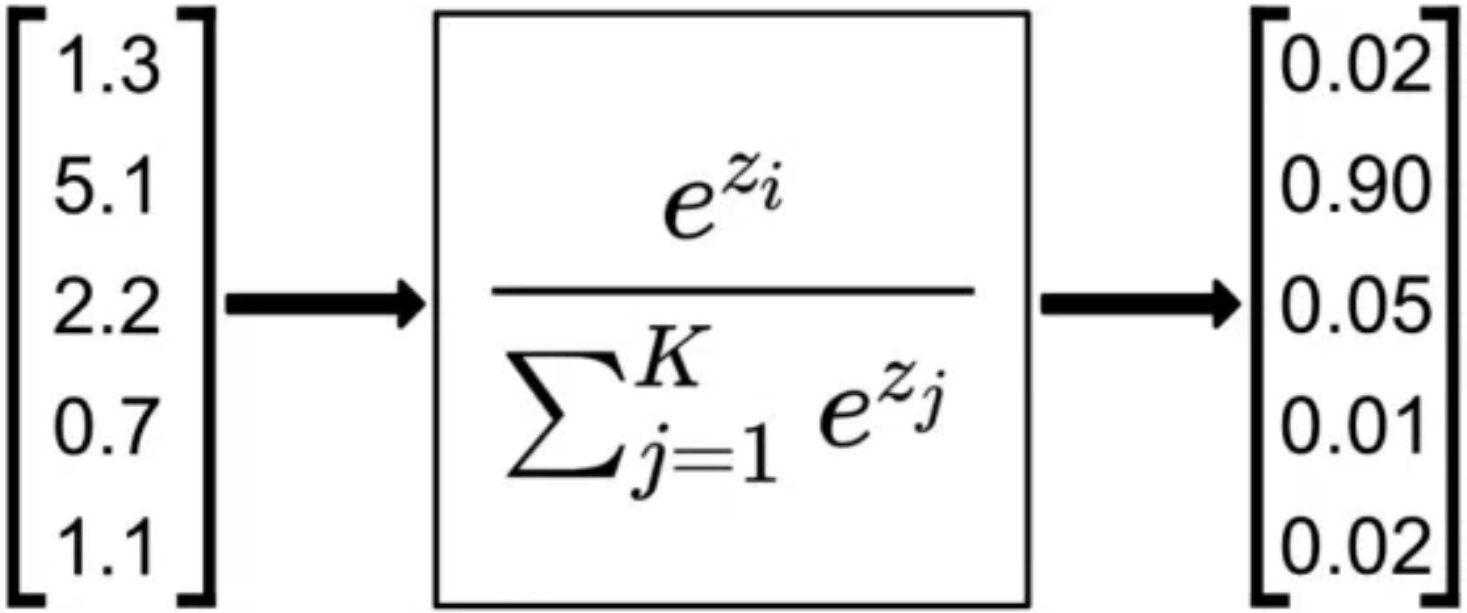
$u^T_{\text{University}}$ v_{NLP}

How to calculate probability?

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

- The softmax function maps arbitrary values x_i to a probability distribution p_i
- “max” because amplifies probability of largest x_i, “soft” because still assigns some probability to smaller x_i

How to calculate probability?



How to calculate probability?

Full Vocabulary ={a, …, at, …, University, …}

I am studying NLP at University College London.

$u^T_a \; v_{NLP}$ … $u^T_{at} \; v_{NLP}$ … $u^T_{am} \; v_{NLP}$

$$P(\text{outside}|\text{central}) = \frac{e^{u_o^T v_c}}{\sum_{w \in V} e^{u_w^T v_c}}$$

How to optimize probability?

I am studying NLP at University College London.

$J_{t,j}$ = -logP(University|NLP) =

$$-u_{\text{University}}^{T} v_{\text{NLP}} + \log \sum_{w \in V} e^{u_w^T v_{\text{NLP}}}$$

$$v_{\text{NLP}} := v_{\text{NLP}} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{\text{NLP}}} \qquad u_{\text{w}} := u_{\text{w}} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_{\text{w}}}, \forall w \in V$$

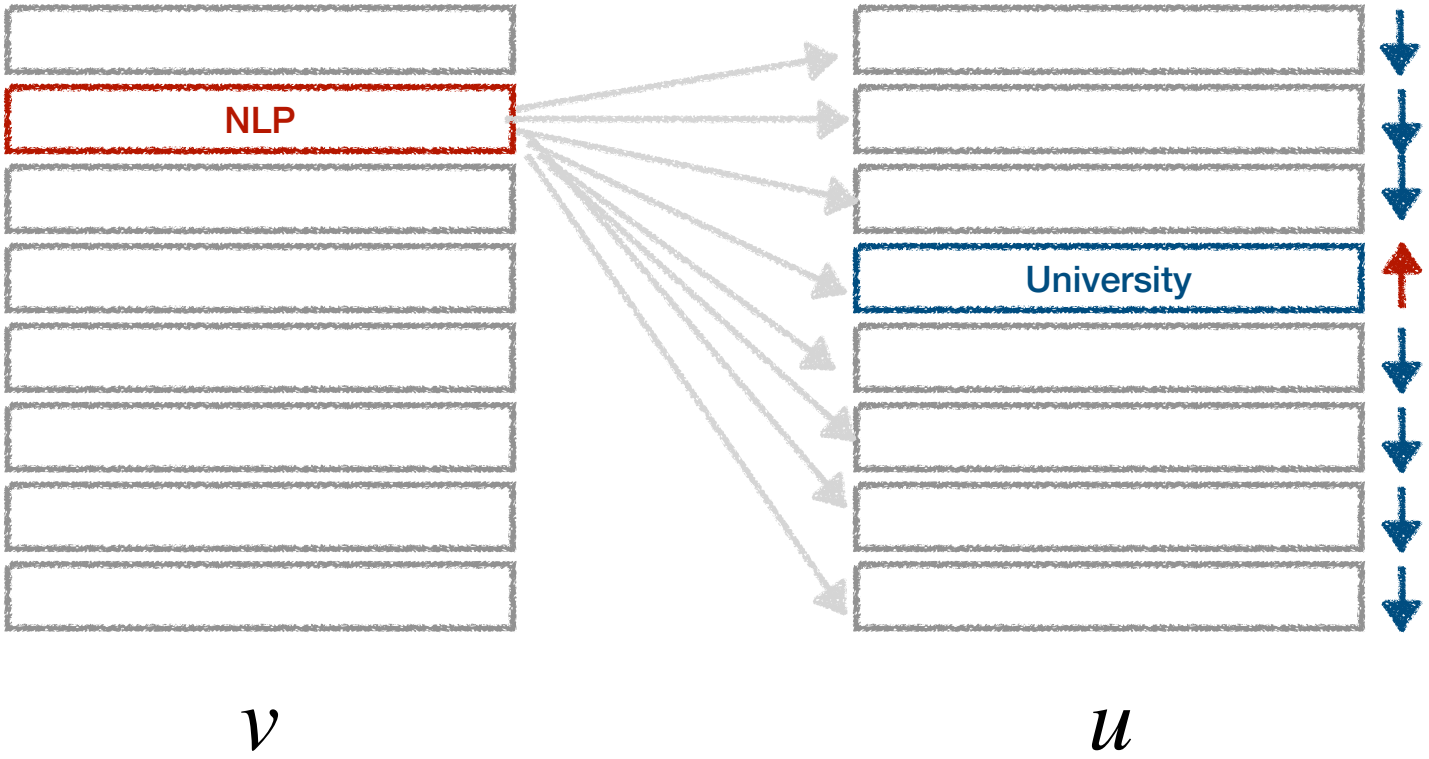
A simplified view

I am studying NLP at University College London.

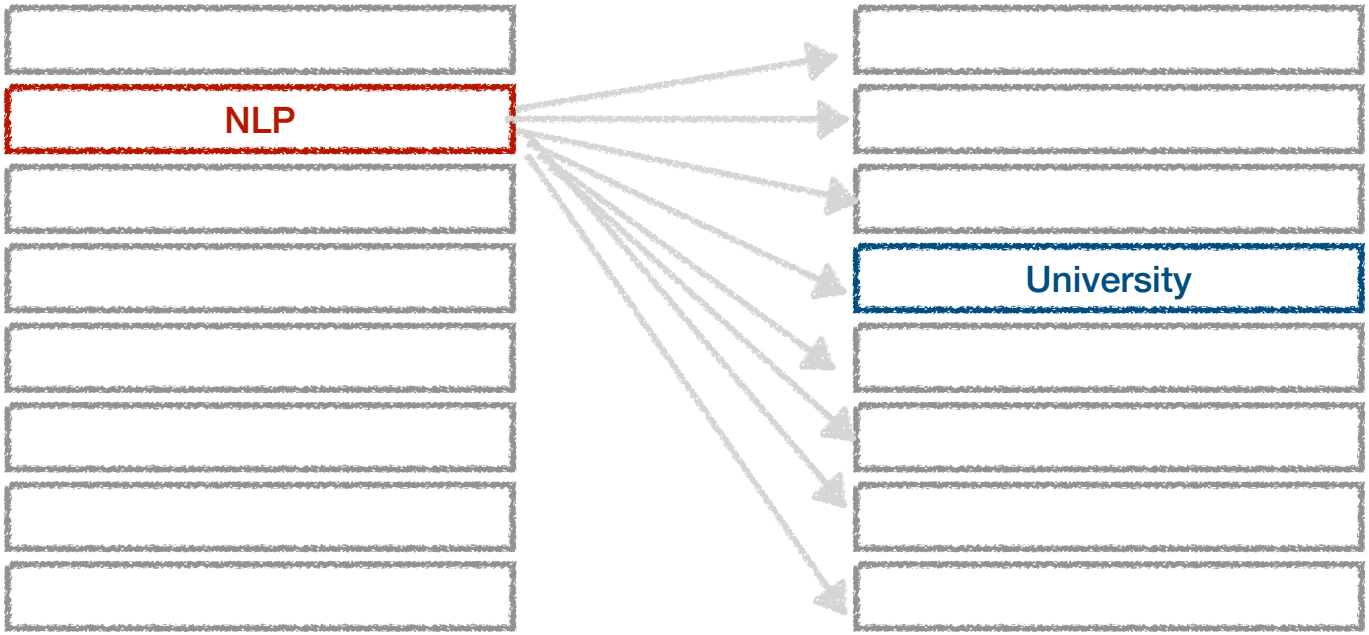
$$-u^T_{\text{University}} v_{\text{NLP}} + \log \sum_{w \in V} e^{u_w^T v_{\text{NLP}}}$$

$\uparrow u^T_{\text{University}} v_{\text{NLP}}$ $\qquad$ $u_w^T v_{\text{NLP}} \downarrow$

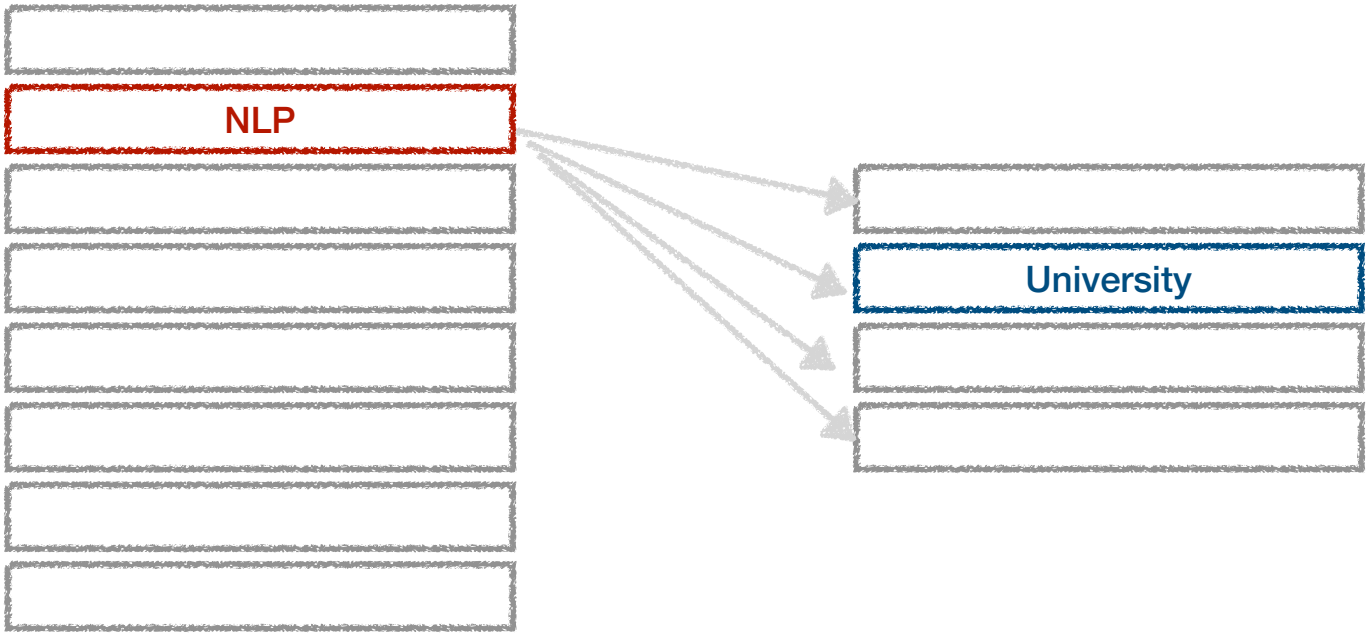
A simplified view



Computation View



Computation over subset?



Negative sampling as an approximation

I am studying NLP at University College London.

(University, NLP) - positive example

(mass, NLP) - negative example

(lamp, NLP) - negative example

(Room, NLP) - negative example

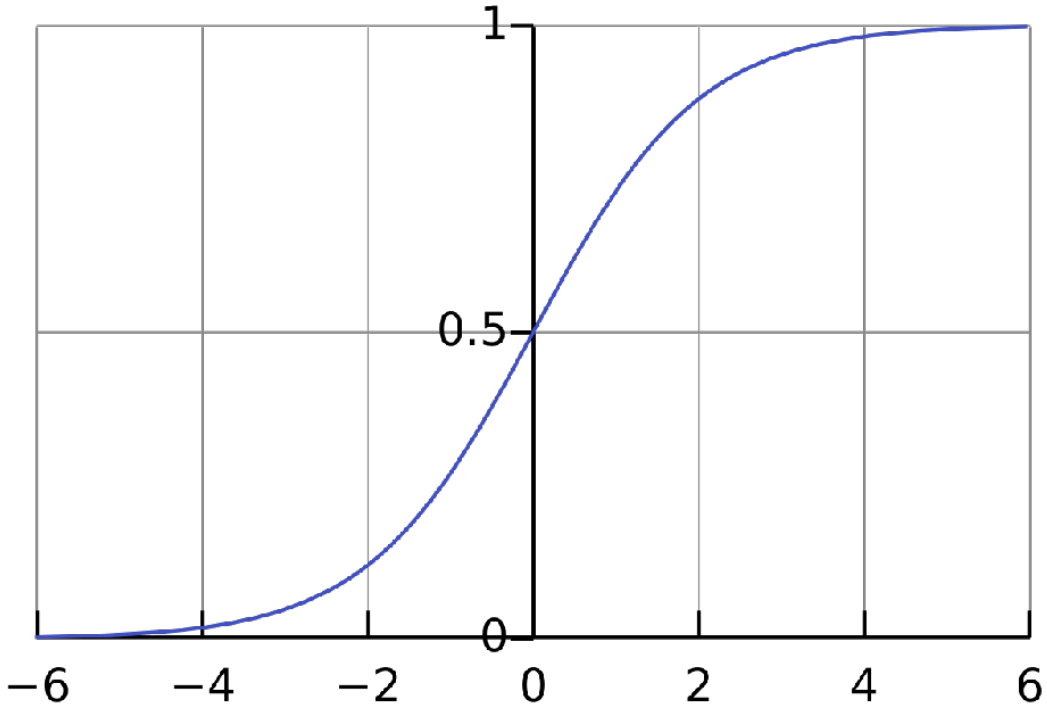
Task: pairs classification

I am studying NLP at University College London.

Central word (v)	Outside word (u)	Label	u^T v
NLP	University	1	0.7
NLP	at	1	0.2
NLP	lamp	0	-1.11

Task: pairs classification

I am studying NLP at University College London.



$$y = \frac{1}{1 + \exp(-x)}$$

Task: pairs classification

I am studying NLP at University College London.

Central word (v)	Outside word (u)	Label	u^T v	Sigmoid()
NLP	University	1	0.74	0.68
NLP	at	1	0.2	0.55
NLP	lamp	0	-1.11	0.25

Updated objective function

I am studying NLP at University College London.

$$P(y = 1 \mid x) = \sigma(x) \qquad P(y = 0 \mid x) = 1 - \sigma(x) = \sigma(-x)$$

Updated objective function

I am studying NLP at University College London.

positive pair negative pairs

$$J_{t,j}(\theta) = -\log \sigma(u_{\text{University}}^{T} v_{\text{NLP}}) - \sum_{w \in \{w_{i1},\ldots,w_{iK}\}} \log \sigma(-u_{w}^{T} v_{\text{NLP}})$$

Updated objective function

I am studying NLP at University College London.

positive pair negative pairs

$$J_{t,j}(\theta) = -\log \sigma(u_{\text{University}}^{T} v_{\text{NLP}}) - \sum_{w \in \{w_{i1},\ldots,w_{iK}\}} \log\left(1 - \sigma(u_{w}^{T} v_{\text{NLP}})\right)$$

Updated objective function

I am studying NLP at University College London.

positive pair negative pairs

$$J_{t,j}(\theta) = -\log \sigma(u^T_{\text{University}} v_{\text{NLP}}) - \sum_{w \in \{w_{i1},\ldots,w_{iK}\}} \sigma(-u^T_w v_{\text{NLP}})$$

$\uparrow u^T_{\text{University}} v_{\text{NLP}}$ $u^T_w v_{\text{NLP}} \downarrow$

Speedup analysis

Full softmax version: |V| times product

$|V| = 10{,}000$

Negative sampling version: K times product

$K = 10$

100x speed up for this part

Dominant Common Words Problem, again

Issue: High-frequency words ("the," "a," "is," "of") dominate sampling process.

Why it's problematic:

- Model learns little from these easy negatives.

Solution: negative sampling trick

$$P(w) \sim Unigram(w)^{0.75}$$

Unigram: Count the frequency of all words in a corpus.

Solution: negative sampling trick

$$P(w) \sim Unigram(w)^{0.75}$$

the: 1,000,000

UCL: 1,000

the$^{0.75}$: 31,622

UCL$^{0.75}$: 177

Hyperparameters

- Number of negative examples: 15–20 for smaller datasets, 2–5 for larger datasets.
- Embedding dimension: 50-300

Context window size

- Common practice: 5 - 10
- Small window: functional and syntactic similarity
- Large window: topical similarities

SkipGram versus CBOW

SkipGram

I am studying NLP at University College London.

Continuous Bag of Word (CBOW)

I am studying NLP at University College London.

SkipGram versus CBOW

I am studying NLP at University College London.

I am studying NLP at University College London.

Which one is better for low frequency word modelling?

Visualisation of Word Embedding

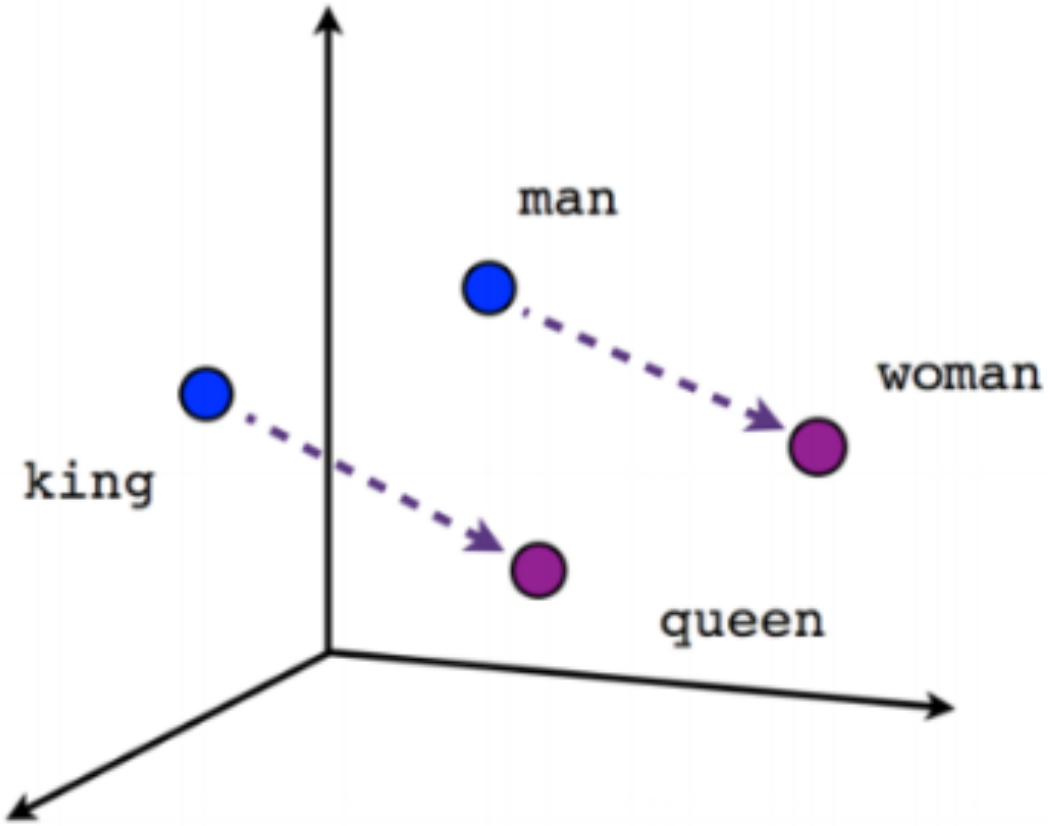
```
King = [ 0.50451 , 0.68607 , -0.59517 ,
-0.022801, … , 0.78321 , -0.91241 ,
-1.6106 , -0.64426 , -0.51042 ]
```

“king”

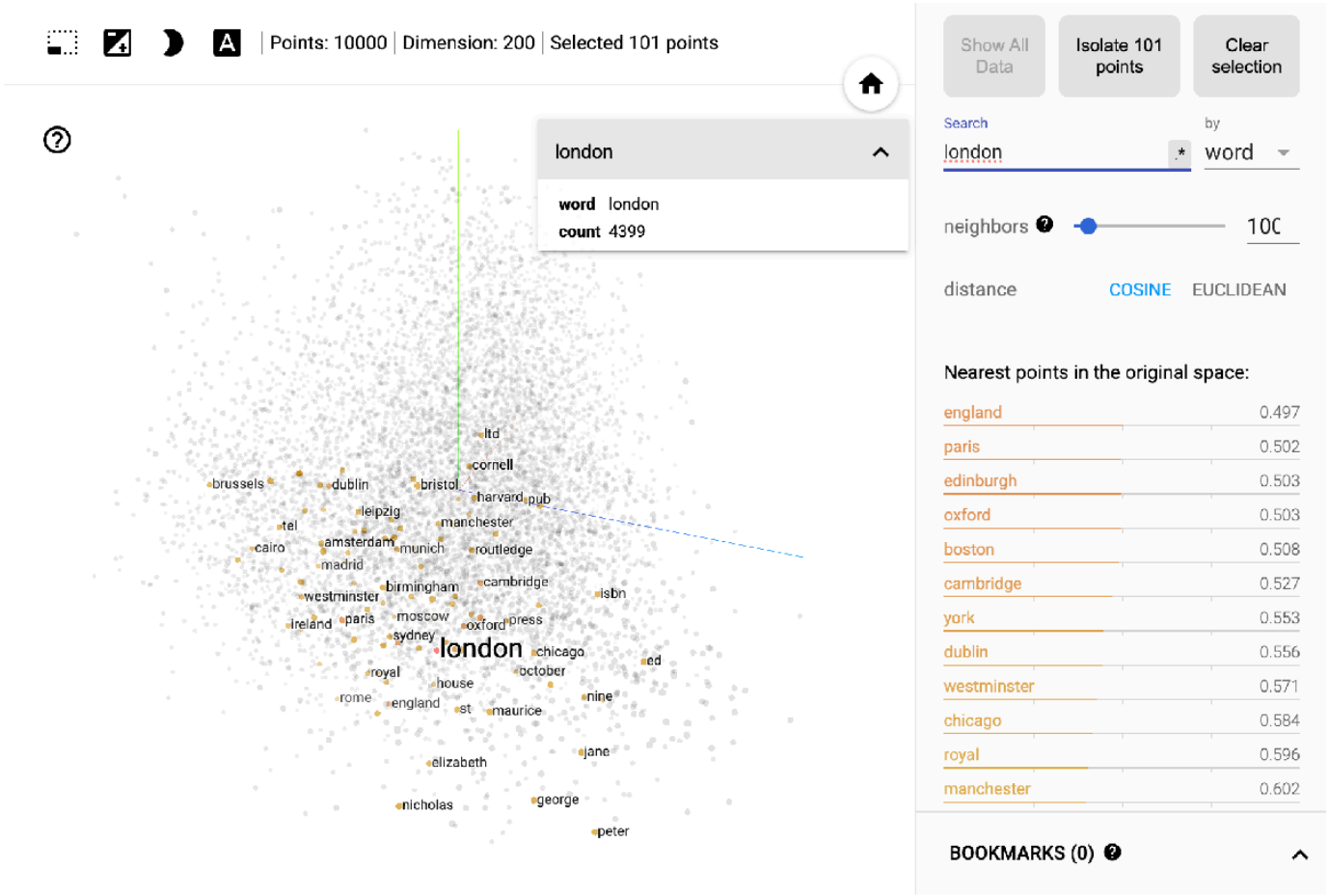
“Man”

“Woman”

Word Embedding Evaluation



Visualisation of Word Embedding

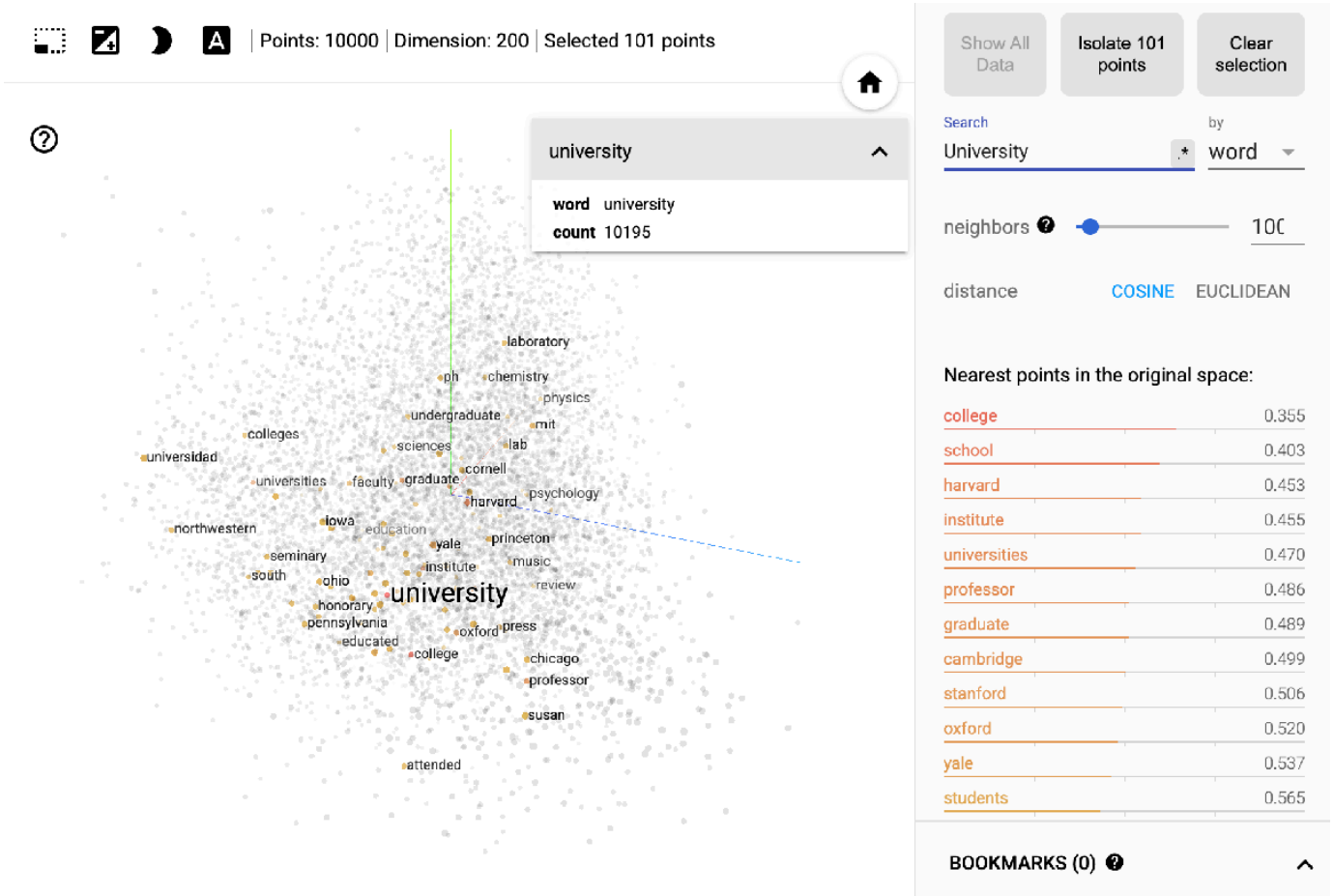


https://projector.tensorflow.org/

Visualisation of Word Embedding

https://projector.tensorflow.org/

Visualisation of Word Embedding



https://projector.tensorflow.org/

Word Embedding Evaluation

Word Similarity Benchmarks

word pair		score
vulgarism	profanity	9.62
subdividing	separate	8.67
friendships	brotherhood	7.5
exceedance	probability	5.0
assigned	allow	3.5
marginalize	interact	2.5
misleading	beat	1.25
radiators	beginning	0

Compute correlation between model predictions and human judgments.

Word Embedding Evaluation

Word Analogy Benchmarks

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Learning from data

How to beat word2vec?

A trick: find better data

Evaluation Leakage

- Why you should create your own dataset for evaluation, sometimes.
- Check king-queen example on public web distribution.

Thank you!