5100-B4-4F19: Information Retrieval

Term Embeddings for Information Retrieval

20 May 2019

Desmond Elliott

https://absalon.instructure.com/courses/31777/

Extensive credits to Mitra and Craswell (2019)

Housekeeping: Assignments

- ~~17 May: Submit peer review of Assignment 1~~
- 20 May: Assignment 2 released
- 27 May: Submit Assignment 2
- 31 May: Submit anonymous peer review of Assignment 2

Assignment 2

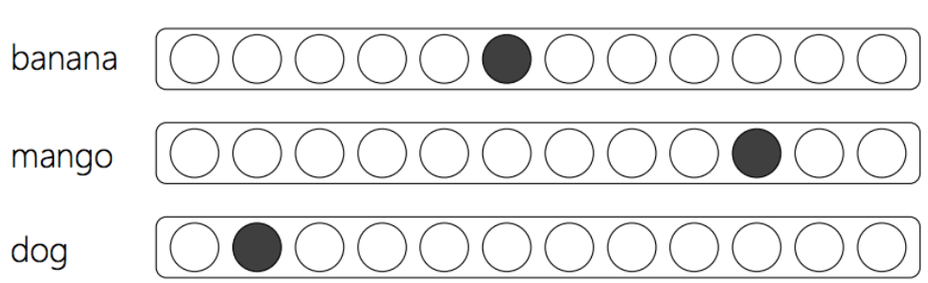
1. **Learn to evaluate existing distributional semantics models - extend ad-hoc text retrieval models with word embeddings.**
   (33% of the whole assignment grade)

2. **Learn to evaluate ad-hoc text retrieval models.**
   (33% of the whole assignment grade)

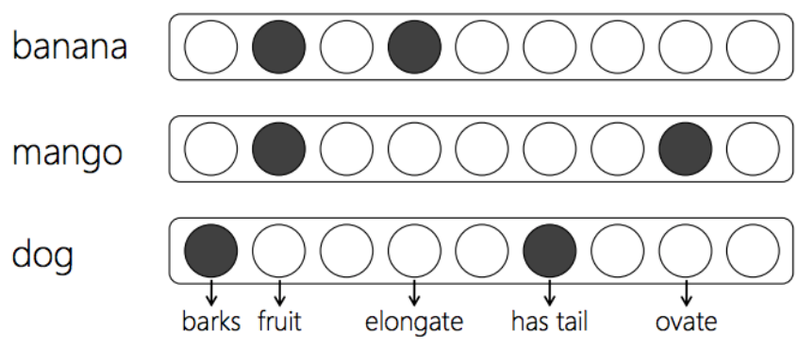
3. **Learn to present experimental results in a report**
   (34% of the whole assignment grade)

What are Term Embeddings?

- Term Embeddings *are* Word Embeddings
- Word embeddings are **dense vectors** that represent words
- Dense: they are not sparse vectors



Sparse $x \in \mathbb{R}^V$



Dense $x \in \mathbb{R}^{300}$

Credit: Figure 3.1 Mitra and Craswell (2018)

This Lecture

- Recap Lecture 2: Word Embeddings
- Estimating query-document relevance with embeddings
- Embeddings for Query Expansion

Word Embeddings (revisited)

- Word embeddings represent words as **dense feature vectors**

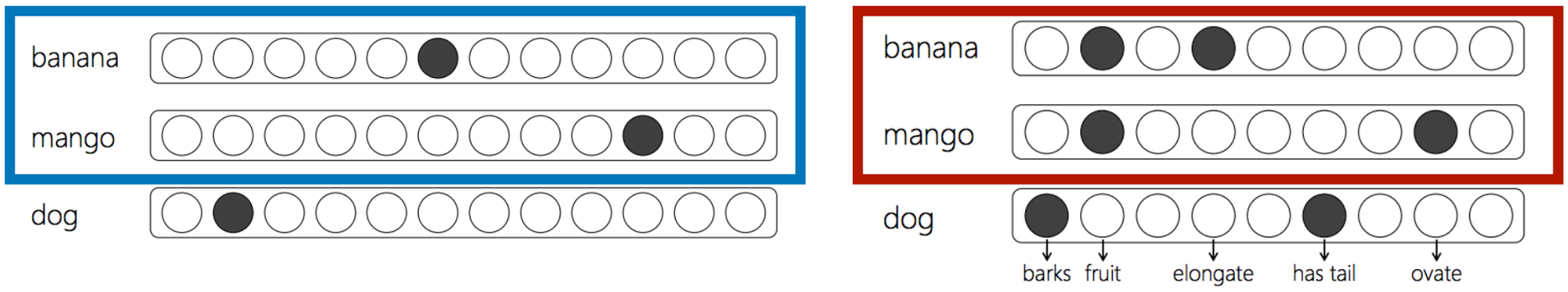
Sparse $\boldsymbol{x} \in \mathbb{R}^{V}$

Dense $\boldsymbol{x} \in \mathbb{R}^{300}$

- Why is this useful?
  - Easier to determine if a pair of words mean the same thing in dense vector space than sparse vector space

Word Similarity in Dense Vector Space

- Consider the words ‘banana’ and ‘mango’



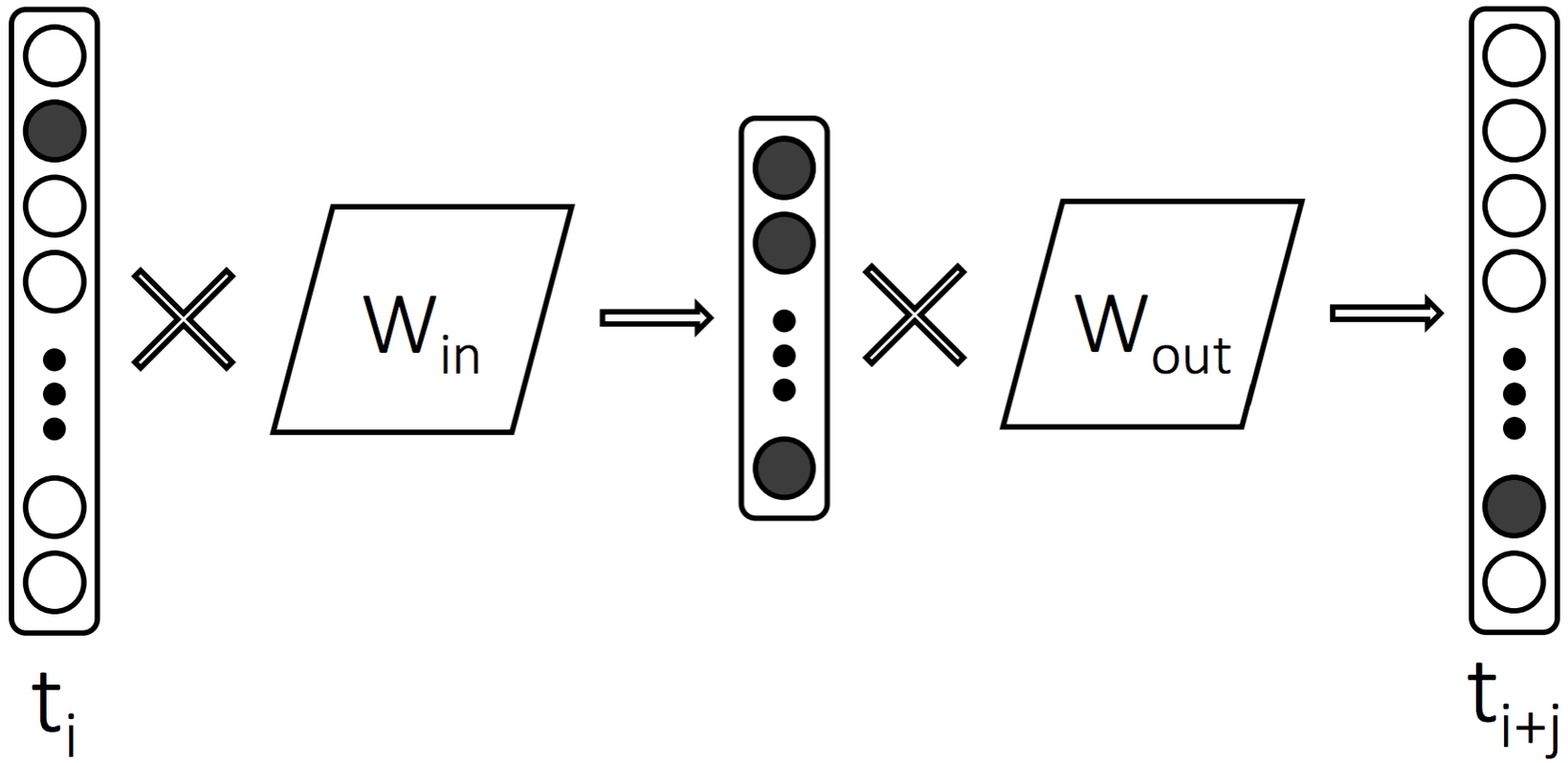
Sparse $x \in \mathbb{R}^V$ Dense $x \in \mathbb{R}^{300}$

- In sparse vector space, their cosine similarity is 0.
- In dense vector space, their cosine similarity is **0.5**

Types of Embeddings

- Type-based
  - Embedding of a term are **based on the type** of the term instead of the context in which they are used.
  - Skip-gram
  - CBOW
  - GLoVE
  - FastText
- Contextual
  - Embedding of a term are **based on the context** in which the term is used.
  - CoVE
  - ELMo 
  - BERT 
  - LISA

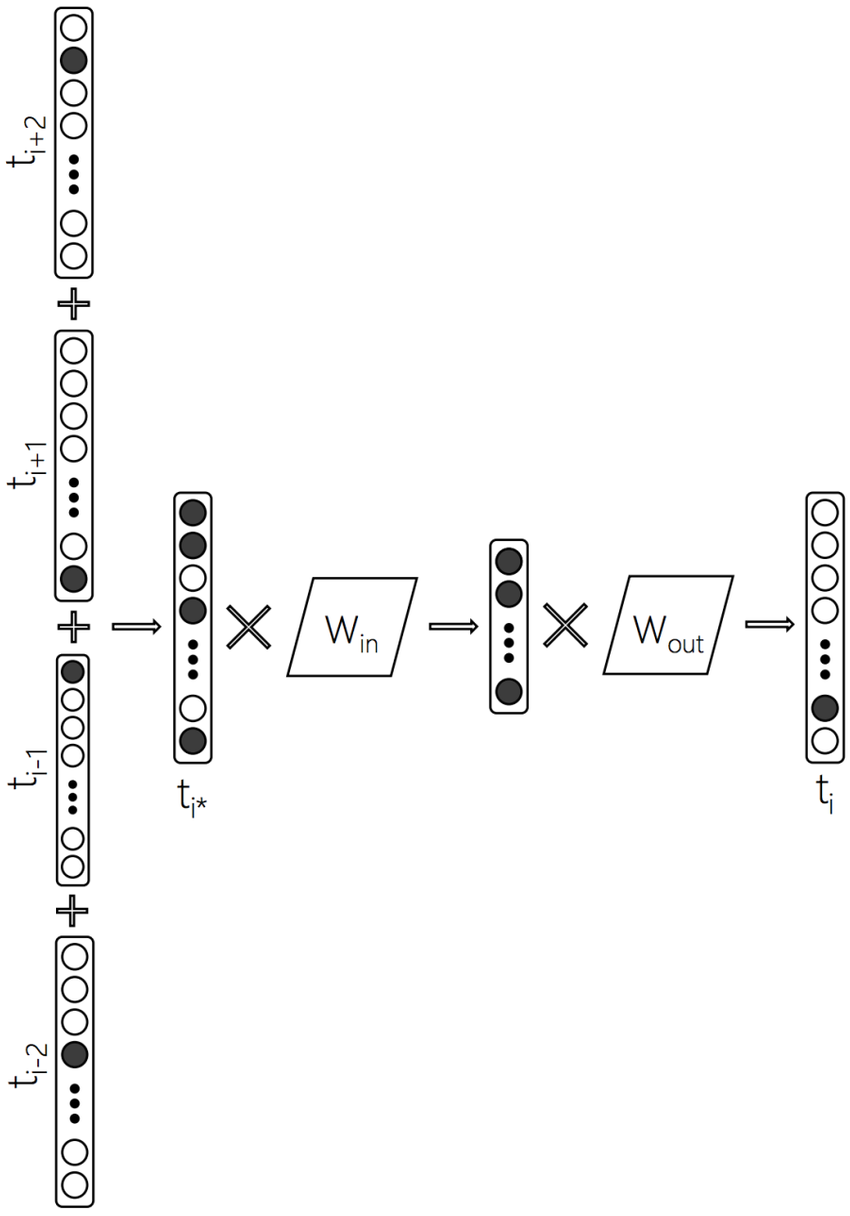
Skip-gram embeddings



- Skip-gram embeddings are learned in the task of **predicting the words in the surrounding context** of the target word

Mikolov et al. 2013

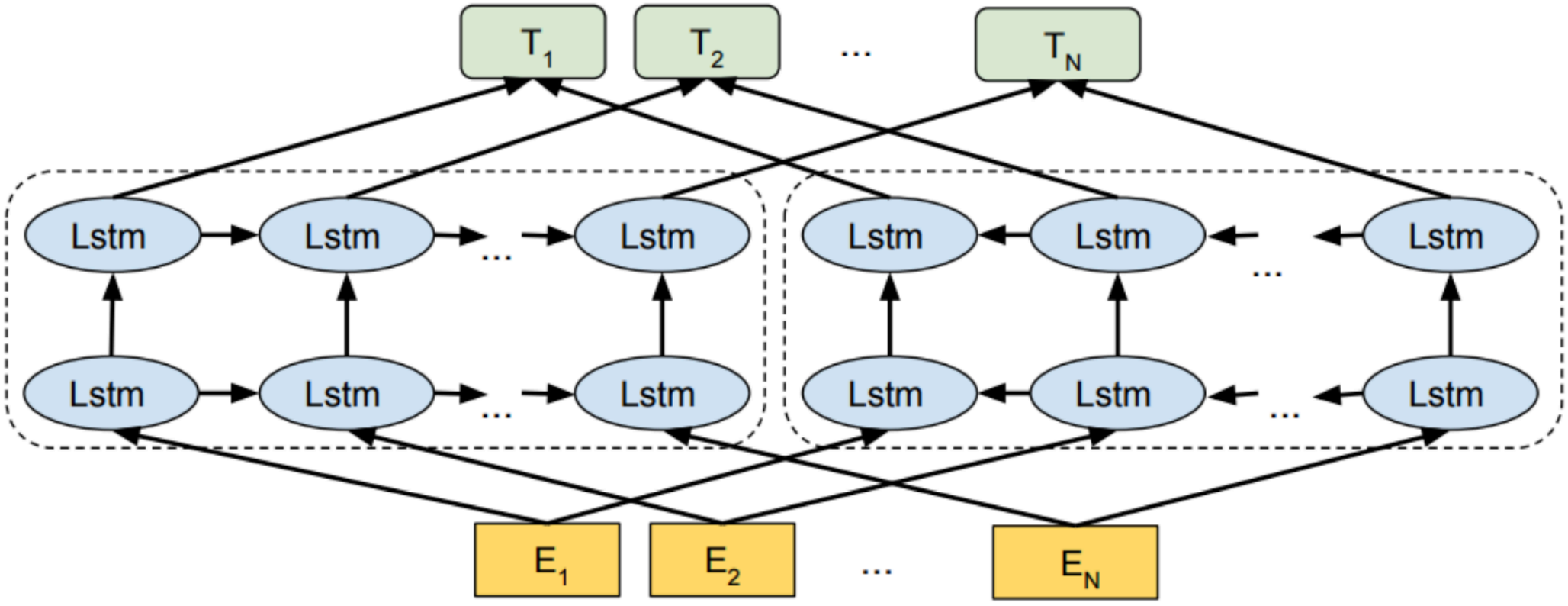
Continuous Bag-of-Words



- CBOW embeddings are learned in the context of **predicting the target word** in given the context of the surrounding words

Mikolov et al. 2013

Embeddings from Language Models 



- ELMo embeddings are learned using a **bidirectional LSTM** in a **language modelling task**

Peters et al. 2018

Bidirectional Encoder Representations from Transformers

- BERT embeddings are **contextual**, **deep**, and learned in a stacked **self-attention** language model

- BERT embeddings are learned through two tasks:

1. Language modelling task

2. Predicting whether a given sentence is the next sentence

Devlin et al. 2019

Relevance Estimation

- We are interested in estimating the relevance of a collection of documents $\mathbf{D_i}$ with document terms $\mathbf{t_d}$, given a query **Q** with query terms $\mathbf{t_q}$.

- In Lecture 1, you learned about how to estimate relevance using BM25 Ranking Model, Language Models, Translation Models, and Pseudo-Relevance Feedback.

- Translation Model: $$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d) \cdot p(t_d|d)$$

- Neural Translation Model: $$p(t_q|t_d) = \frac{cos(\vec{v}_{t_q}, \vec{v}_{t_d})}{\sum_{t \in T} cos(\vec{v}_t, \vec{v}_{t_d})}$$

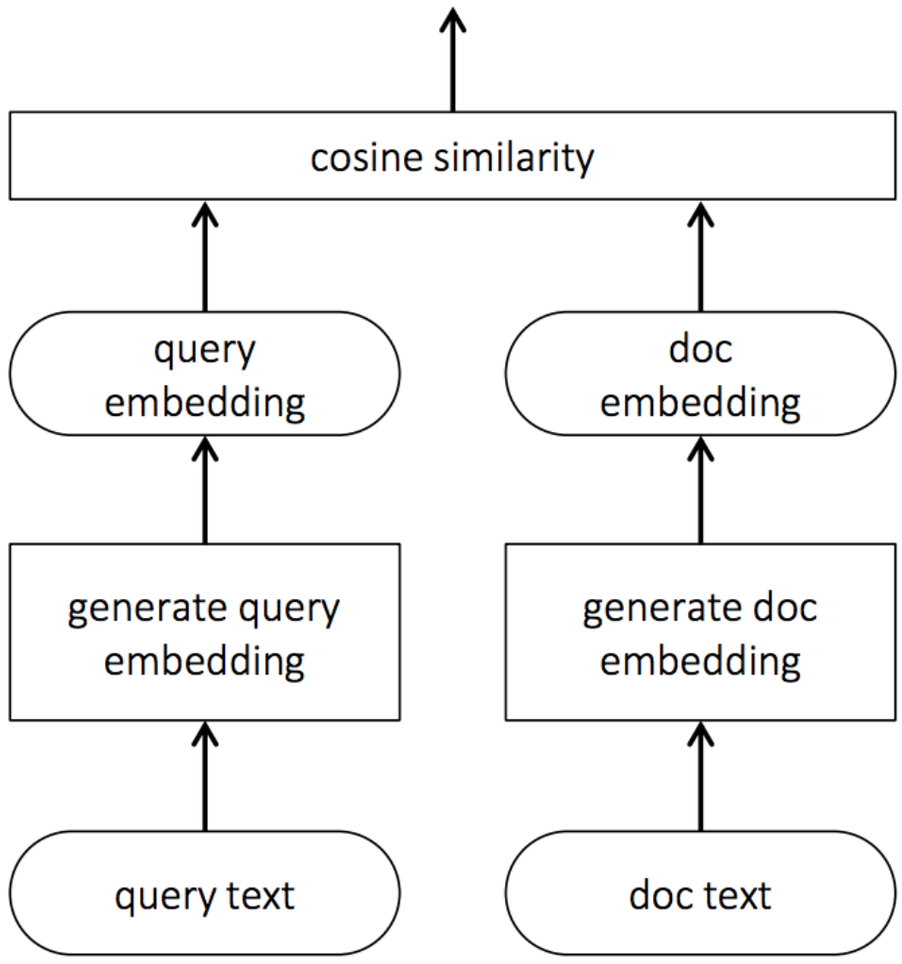
Relevance Estimation *with* *Neural Networks*

3. **Rank the documents** in the collection by their similarity

2. **Estimate the similarity** between the embeddings

1b. **Transform document** into embedding

1a. **Transform query** into embedding



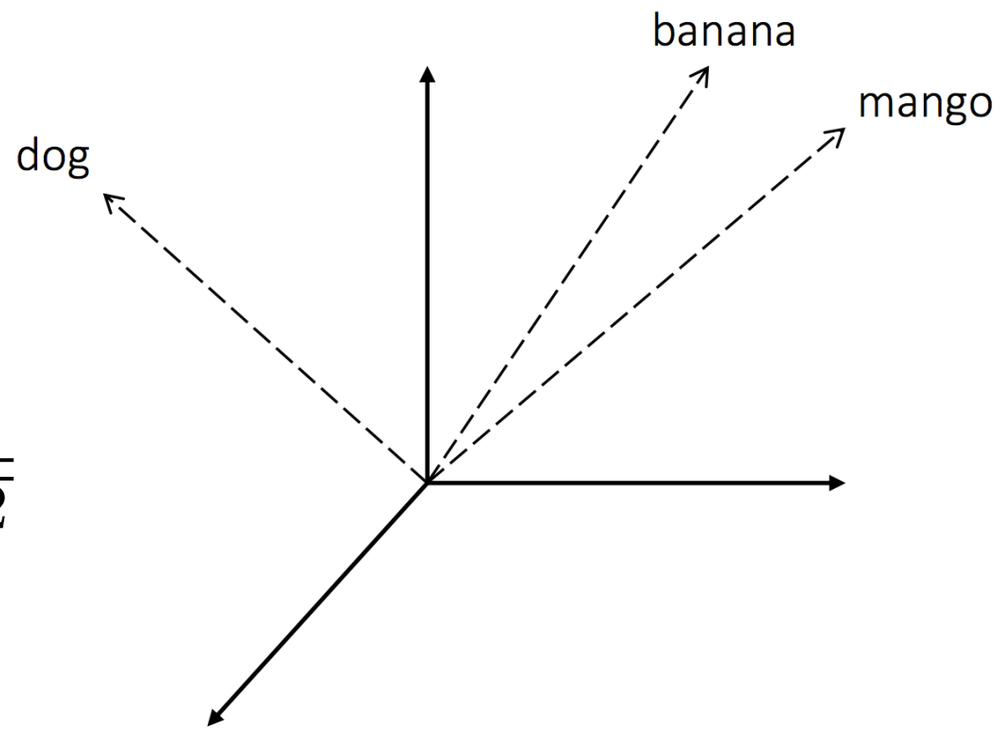
Cosine Similarity

(revisited)

- Cosine similarity will tell us the **angle formed between vectors A and B,** where A and B are representations of the different terms

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

$$= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



Averaged Word Embeddings

- Measure the cosine similarity between a vector representing the **average of the query term embeddings**, and a vector of the **average of the document embedding terms**

$$sim(q,d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^{\intercal}\vec{v}_d}{\|\vec{v}_q\|\|\vec{v}_d\|}$$

$$\text{where,} \quad \vec{v}_q = \frac{1}{|q|}\sum_{t_q \in q}\frac{\vec{v}_{t_q}}{\|\vec{v}_{t_q}\|}$$

$$\vec{v}_d = \frac{1}{|d|}\sum_{t_d \in d}\frac{\vec{v}_{t_d}}{\|\vec{v}_{t_d}\|}$$

What's wrong with AWE?

1. **Randomly shuffling a sentence** does not change its averaged word embedding representation

2. **Similarly-worded sentences with different meanings are equivalent** with an AWE representation

What's wrong with AWE?

1. **Randomly shuffling a sentence** does not change its averaged word embedding representation

2. **Similarly-worded sentences with different meanings are equivalent** with an AWE representation

Monty Python's Flying Circus, The Royal Philharmonic Orchestra Goes to the Bathroom, 1969

The Agatha Christie Sketch

Monty Python

This house is surrounded.

1. I’m afraid I must not ask anyone to leave the room.

No, I must ask nobody ... no, I must ask everybody to...

2. I must not ask anyone to leave the room.

3. No one must be asked by me to leave the room.

4. No, no one must ask the room to leave.

5. I ... I ... ask the room shall by someone be left. Not.

6. Ask nobody the room somebody leave shall I.

Shall I leave the room?

7. Everyone must leave the room… as it is... with them in it.

Phew. Understand?

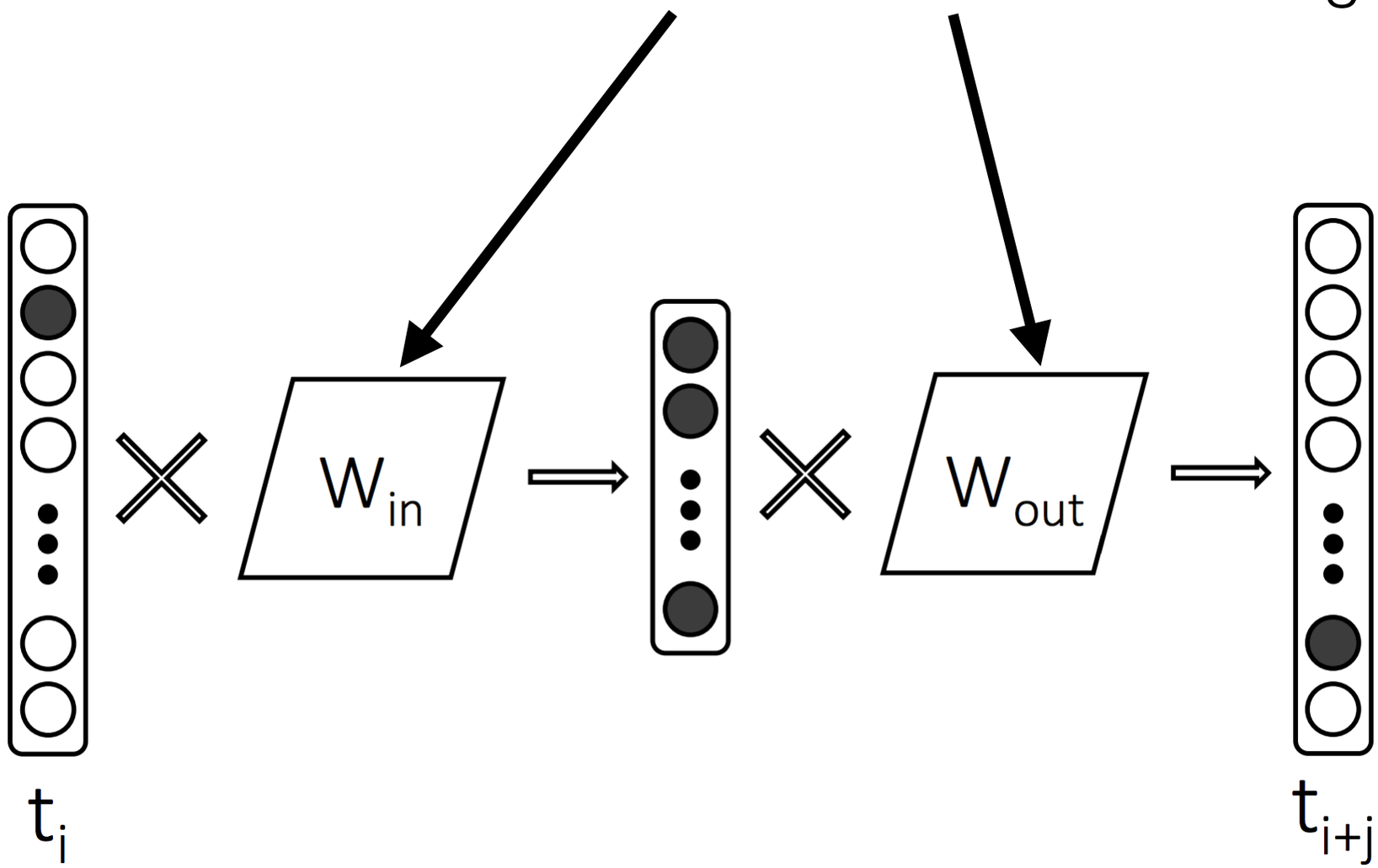
Cosine similarity of the incorrect sentences in the sketch

```
# wget https://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.en.vec
import sklearn.metrics
import numpy as np
import gensim.models.keyedvectors as kv
Import sklearn.metrics.pairwise as pairwise

model = kv.load_word2vec_format('wiki.en.vec',cbinary=False)

def average_word_embedding(model, words):
    # Handle OOV
    words = [word for word in words if word in model.vocab]
    if len(words) >= 1:
        return np.mean(model[words], axis=0)
    else:
        return []

s1 = "I'm afraid I must not ask anyone to leave the room."
s2 = "I must not ask anyone to leave the room."
v1 = average_word_embedding(model, s1).reshape(1, -1)
v2 = average_word_embedding(model, s2).reshape(1, -1)
sklearn.metrics.pairwise.cosine_similarity(v1, v2)
```

Average cosine similarity is 0.982

Input / Output Embeddings

- Some methods for learning embeddings actually learn two projections. M&C call these **IN** and **OUT** embeddings.



The Effect of using Input /Output Embeddings

- Different nearest neighbours depending on how you choose the vectors used in `cosine()`

$$sim(q,d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^{\intercal}\vec{v}_d}{\|\vec{v}_q\|\|\vec{v}_d\|}$$

$\vec{v}_q$ $\vec{v}_d$	**yale**
IN-IN	
yale	
harvard	
nyu	
cornell	
tulane	
tufts	

The Effect of using Input /Output Embeddings

- Different nearest neighbours depending on how you choose the vectors used in `cosine()`

$$sim(q,d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^{\intercal}\vec{v}_d}{\|\vec{v}_q\|\|\vec{v}_d\|}$$

$\vec{v}_q$ $\vec{v}_d$	**yale**
IN-IN	OUT-OUT
yale	yale
harvard	uconn
nyu	harvard
cornell	tulane
tulane	nyu
tufts	tufts

The Effect of using Input /Output Embeddings

- Different nearest neighbours depending on how you choose the vectors used in `cosine()`

$$sim(q,d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^{\intercal}\vec{v}_d}{\|\vec{v}_q\|\|\vec{v}_d\|}$$

$\vec{v}_q$ $\vec{v}_d$	**yale**	
IN-IN	OUT-OUT	IN-OUT
yale	yale	yale
harvard	uconn	faculty
nyu	harvard	alumni
cornell	tulane	orientation
tulane	nyu	haven
tufts	tufts	graduate

DESM: Dual Embedding Space Model

- Question: How can we benefit from the difference between representing words using the INPUT or OUTUT embeddings?

Nalisnick et al. 2016

DESM: Dual Embedding Space Model

- Question: How can we benefit from the difference between representing words using the INPUT or OUTUT embeddings?

- Key idea of DESM:

- **Query embeddings** are from the INPUT

- **Document embeddings** are from the OUTPUT

Nalisnick et al. 2016

DESM (continued)

- **Precompute document representations** as they are independent of a query

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$$

- Two flavours
  - Query: **IN** / Doc **OUT**

$$DESM_{IN-OUT}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{q_{IN,i}^T \overline{D_{OUT}}}{\|q_{IN,i}\| \|D_{OUT}\|}$$

  - Query: **IN** / Doc **IN**

$$DESM_{IN-IN}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{q_{IN,i}^T \overline{D_{IN}}}{\|q_{IN,i}\| \|D_{IN}\|}$$

DESM (continued)

- **Precompute document representations** as they are independent of a query

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}$$

- Two flavours
  - Query: **IN** / Doc **OUT**

$$DESM_{IN-OUT}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{q_{IN,i}^T \overline{D_{OUT}}}{\|q_{IN,i}\| \|D_{OUT}\|}$$

  - Query: **IN** / Doc **IN**

$$DESM_{IN-IN}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{q_{IN,i}^T \overline{D_{IN}}}{\|q_{IN,i}\| \|D_{IN}\|}$$

Take-away: **IN-OUT performs better than IN-IN**

Quick Quiz

http://kahoot.it/

Language Models for IR

- Estimate the probability of translating terms in the document into the query:

$$p_t(w|d) = \sum_{u \in d} p_t(w|u) p(u|d)$$

All words in the document

Translating document term into a query term

Prior probability that the word belongs in this document

Zhai and Lafferty 2004

Neural Translation Language Model

- Estimate $p_t(w|u)$ using word embedding similarity

$$p_t(w|d) = \sum_{u \in d} p_t(w|u) p(u|d)$$

Zuccon et al. 2015

Neural Translation Language Model

- Estimate $p_t(w|u)$ using word embedding similarity

$$p_t(w|d) = \sum_{u \in d} p_t(w|u) p(u|d)$$

$$p\left(t_q|t_d\right) = \frac{\cos\left(\vec{v}_{t_q}, \vec{v}_{t_d}\right)}{\sum_{t \in T} \cos\left(\vec{v}_t, \vec{v}_{t_d}\right)}$$

Zuccon et al. 2015

Neural Translation Language Model

- Estimate $p_t(w|u)$ using word embedding similarity

$$p_t(w|d) = \sum_{u \in d} p_t(w|u) p(u|d)$$

$$p\left(t_q|t_d\right) = \frac{\cos\left(\vec{v}_{t_q}, \vec{v}_{t_d}\right)}{\sum_{t \in T} \cos\left(\vec{v}_t, \vec{v}_{t_d}\right)}$$

- Averaged word embeddings $\vec{v}_q = \frac{1}{|q|} \sum_{t_q \in q} \frac{\vec{v}_{t_q}}{\|\vec{v}_{t_q}\|}$

$$\vec{v}_d = \frac{1}{|d|} \sum_{t_d \in d} \frac{\vec{v}_{t_d}}{\|\vec{v}_{t_d}\|}$$

Zuccon et al. 2015

Word Mover's Distance

- Word Mover's Distance is inspired by **Earth Mover's Distance**

    - **EMD** measures the `min` cost of transforming one distribution X into another distribution Y

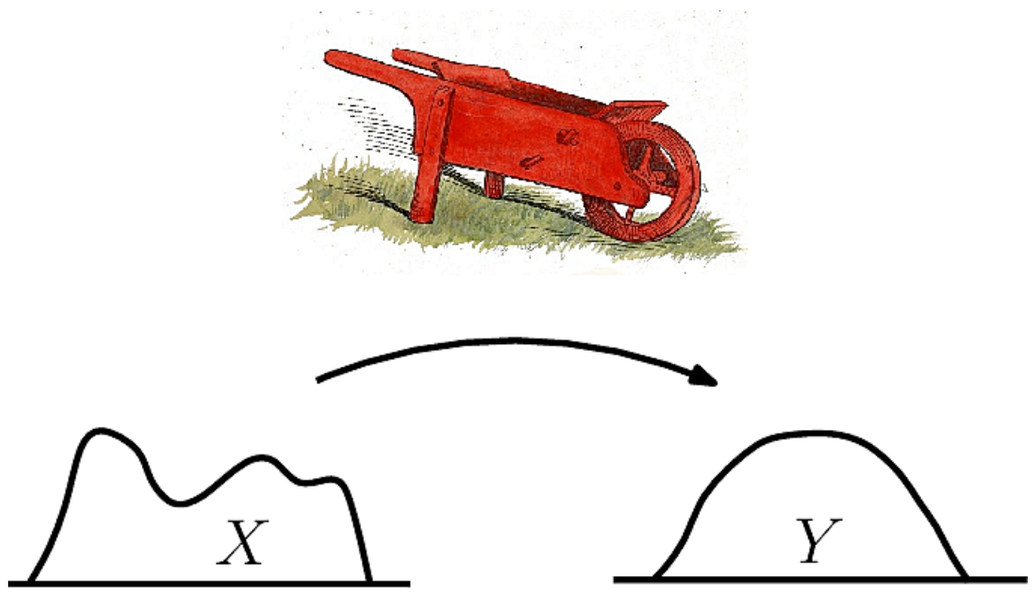


Figure Credit: Structural Bioinformatics Library

WMD Explained

1. Convert the document to normalised Bag-of-Words vector

   a. Count the repeated words in the document

The woman changed the wheel of the car.

↓

The woman changed wheel of car .

d

the	…	changed	…	of	car	.	woman	…	…	wheel
3	…	1	…	1	1	1	1	…	…	1

$$d_i = \frac{c_i}{\sum_{j=1}^{n} c_j}$$

WMD Explained

- Need a cost to estimate the distance between words i and j
  - Euclidean $c(i,j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$
- Estimate the document cost as a transportation problem

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j)$$

$$\text{subject to: } \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \forall i \in \{1,\ldots,n\}$$

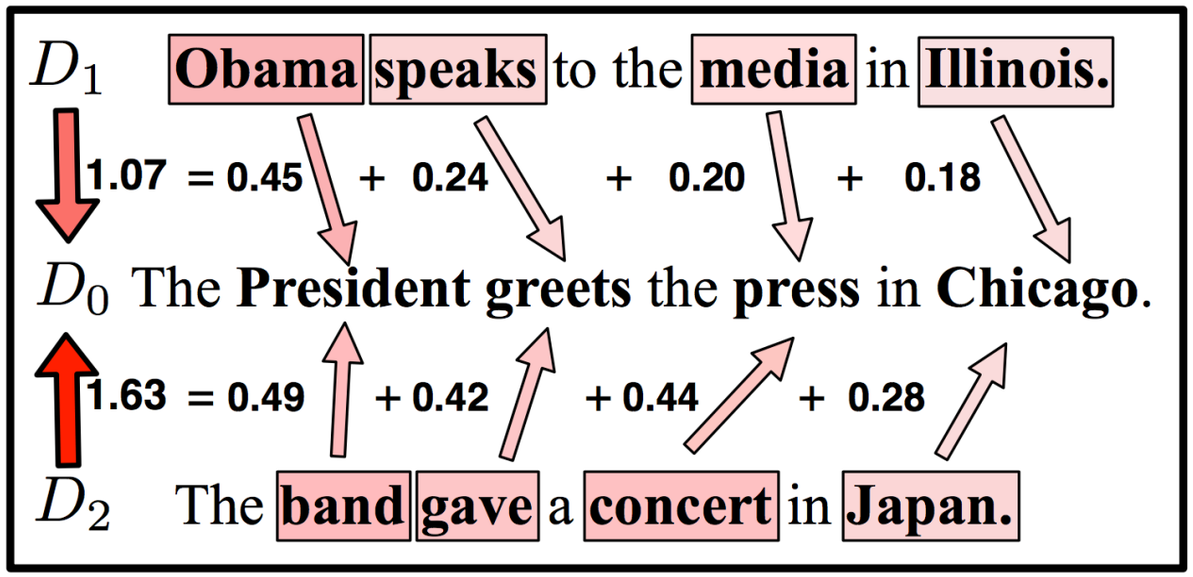
The outgoing "flow" from $\mathbf{d_i}$ must be entirely used

$$\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1,\ldots,n\}$$

The incoming flow to $\mathbf{d_j}$ in the target document must not exceed $\mathbf{d_j}$

WMD Example

- In WMD, it costs less to transform $D_1 \rightarrow D_0$ than $D_2 \rightarrow D_0$.



- WMD can estimate the cost when there are **no overlapping words**.
  - Question: What is the cost if we replace **Obama** with **Trump** or **Lars**?

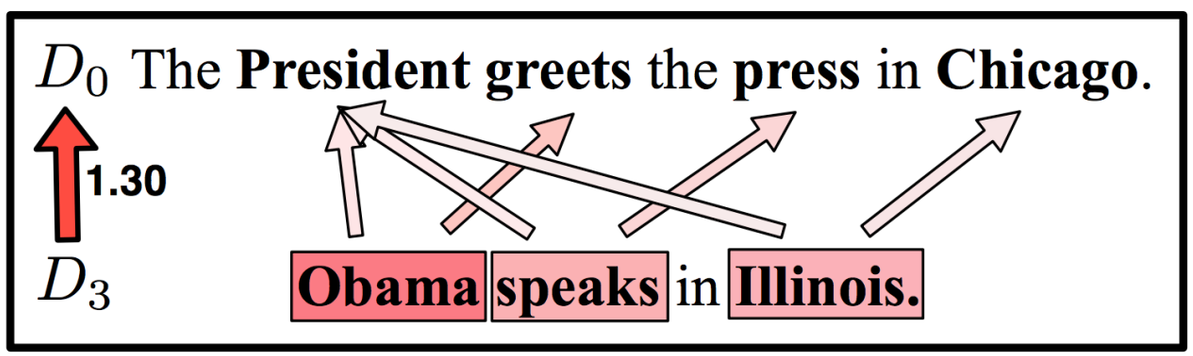


Figure Credit: Kusner et al. 2015

Query Expansion

- Query expansion is **the task of reformulating the original query** to **find a new query** that will **retrieve a better list of candidate documents**.

- Given query $\mathbf{q}$, document collection $\mathbf{D}$, and performance measure $\mathbf{perf}$, find $\mathbf{q'}$ such that $\mathbf{perf(q'|D) > perf(q|D)}$

- Example  $\mathbf{q}$: party April Copenhagen University

Figure Credits: Video Blocks, Docubyte

Query Expansion

- Query expansion is **the task of reformulating the original query** to **find a new query** that will **retrieve a better list of candidate documents**.

- Given query $\mathbf{q}$, document collection $\mathbf{D}$, and performance measure $\mathbf{perf}$, find $\mathbf{q}$' such that $\mathbf{perf(q'|D) > perf(q|D)}$

- Example  $\mathbf{q}$: party April Copenhagen University

Figure Credits: Video Blocks, Docubyte

Query Expansion

- Query expansion is **the task of reformulating the original query** to **find a new query** that will **retrieve a better list of candidate documents**.

- Given query $\mathbf{q}$, document collection $\mathbf{D}$, and performance measure $\mathbf{perf}$, find $\mathbf{q'}$ such that $\mathbf{perf(q'|D) > perf(q|D)}$

- Example  $\mathbf{q}$: party April Copenhagen University

$\mathbf{q'}$: party April Copenhagen University ***spring festival***

Figure Credits: Video Blocks, Docubyte

Term Embeddings for Query Expansion

- Term Embeddings can also be used for query expansion
- Approaches are based on estimating a `score` between a candidate term $\mathrm{t_c}$ and the query terms q.

$$\mathrm{score}\left(t_c, q\right) = \frac{1}{|q|} \sum_{t_q \in q} \cos\left(\vec{v}_{t_c}, \vec{v}_{t_q}\right)$$

- Performs worse than pseudo-relevance feedback on its own; **performs better than PRF when used as a combination of approaches.**

Practical Matters: Telescoping

- Term Embeddings for IR **rarely work well on their own**. They are often **used in conjunction** with more traditional ranking models (BM25, etc.)

- The conjunction typically consists of **a pipeline**, in which the term embeddings are used to estimate the relevance of a **smaller set of candidate documents.**

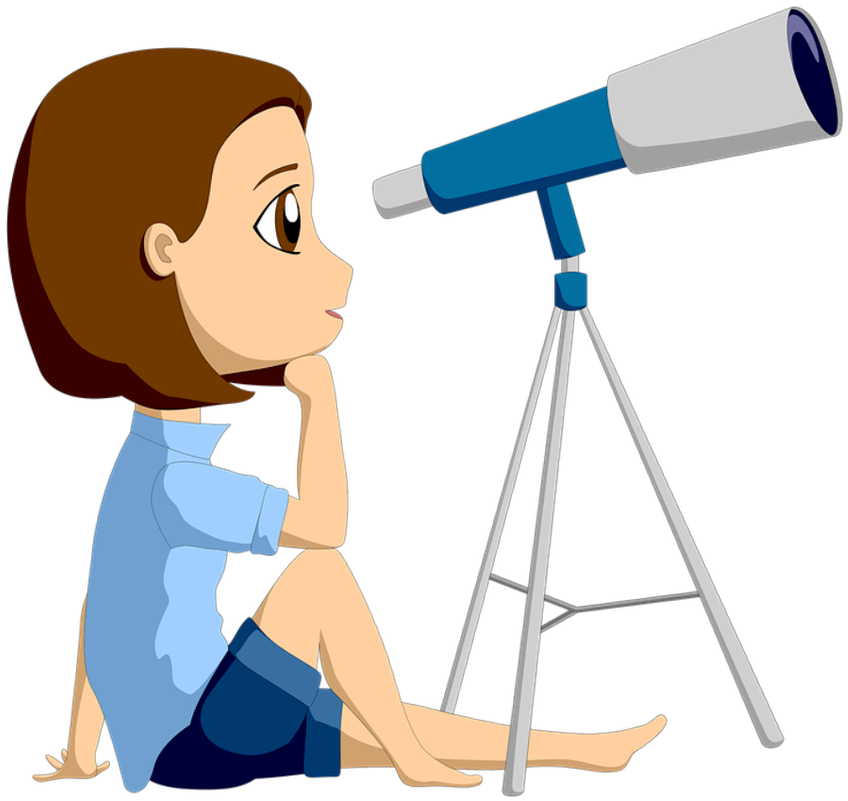
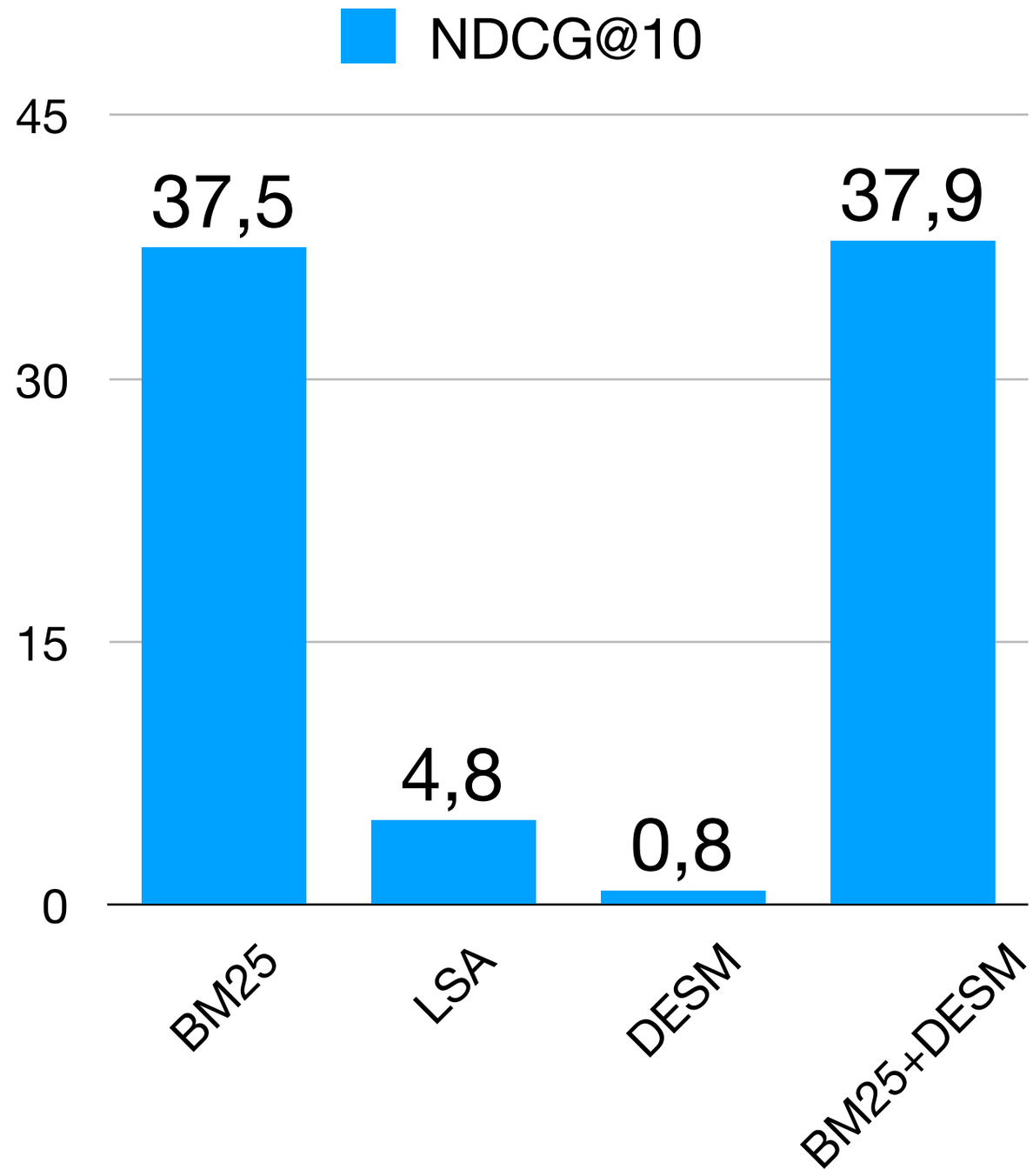


Figure Credit: Pixabay

Telescoping in DESM

- Rank the initial set of documents using BM25
- Re-rank the documents using $DESM_{IN\text{-}OUT}$

➡ Improvement in retrieval performance



Lecture Summary

- Query-Document Relevance Estimation with Neural Networks
    - Averaged Word Embeddings
    - Dual Embedding Space Model
    - Word Mover's Distance
- Query Expansion
- Telescoping for Improved Model Performance