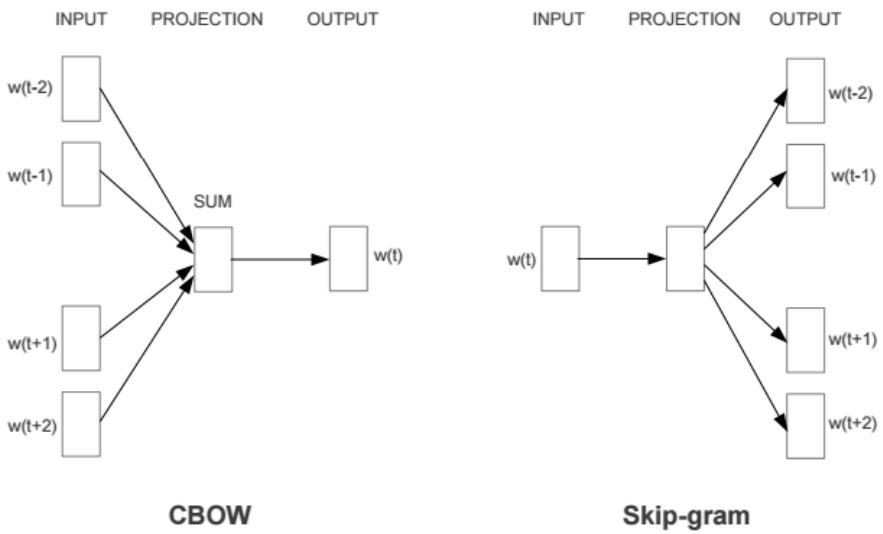


Word Representations

Jay Urbain, PhD
jay.urbain@gmail.com
https://www.linkedin.com/in/jayurbain/

EMBEDDINGS

WORKING WITH TEXT

- Neural networks don't take raw text as input. Only numeric tensors.
- *Vectorizing* text is the process of transforming text into numeric tensors. This can be done in multiple ways:
  - **Segment text into words, and transform each word into a vector.**
  - Segment text into characters, and transform each character into a vector.
  - Extract n-grams of words or characters, and transform each n-gram into a vector.
- Apply categorical transformation:
  - One-hot encoding
  - Word embedding

ONE-HOT ENCODING

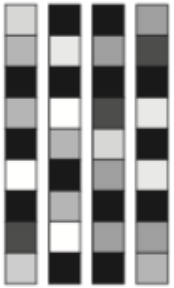
- Words are categorical, they have no *ordinal* relationship. So they can't just be numerically encoded.
- For each word:
  - Numerically encode each word: {red:0, green:1, blue:2}
  - Generate vector of zeros the length of all possible words/categories
  - Set the index value for the word in the vector

red,	green,	blue
1,	0,	0
0,	1,	0
0,	0,	1

WORD EMBEDDINGS

- One-hot results in *sparse vector representation* (mostly zeros, hi dimensions).
- Word embeddings provide *dense vector representations*.
- Idea is to “embed” words into a lower-dimensionality space.
- The dimensions of this space are typically defined by word context, i.e., semantically similar words are embedded near each other.
- Popular algorithms:
  - PMI,Word2Vec: Skip-gram or CBOW, GLoVE, fasttext, …

One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

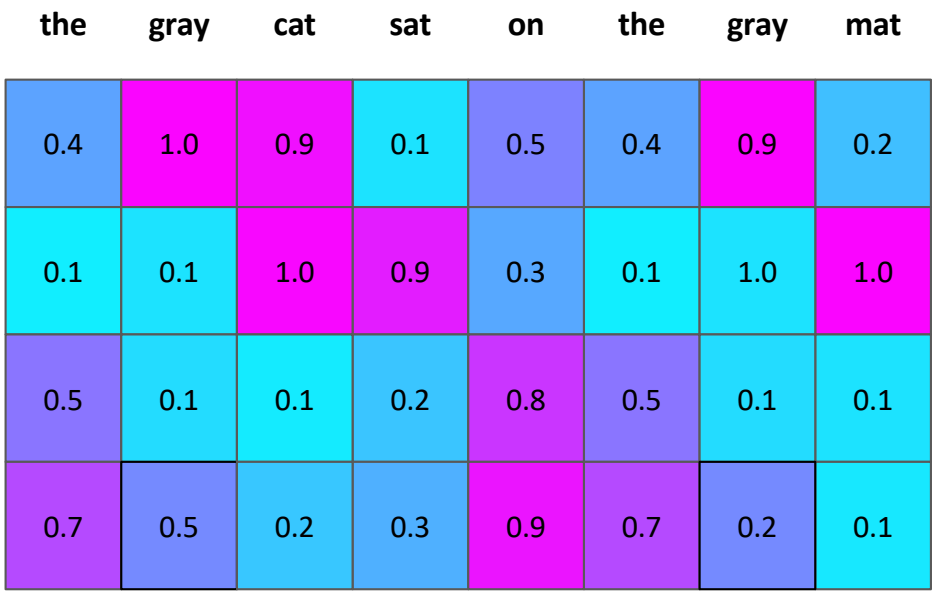
The Distributional Hypothesis (Firth, 1957)

- ‘You can tell a word by the company it keeps’

  - *The cat sat on the mat*
  - *The dog sat on the mat*
  - *The elephant sat on the mat*
  - *The quickly sat on the mat*

Embeddings: distributed representation of text

Text is represented *categorically*, but we want similar words to have similar representations




```
embedding_layer = nn.Embedding(num_embeddings=vocab_size, embedding_dim=4)
embeddings = embedding_layer('the gray cat sat on the gray mat'.split())
```

Embeddings: Word2Vec

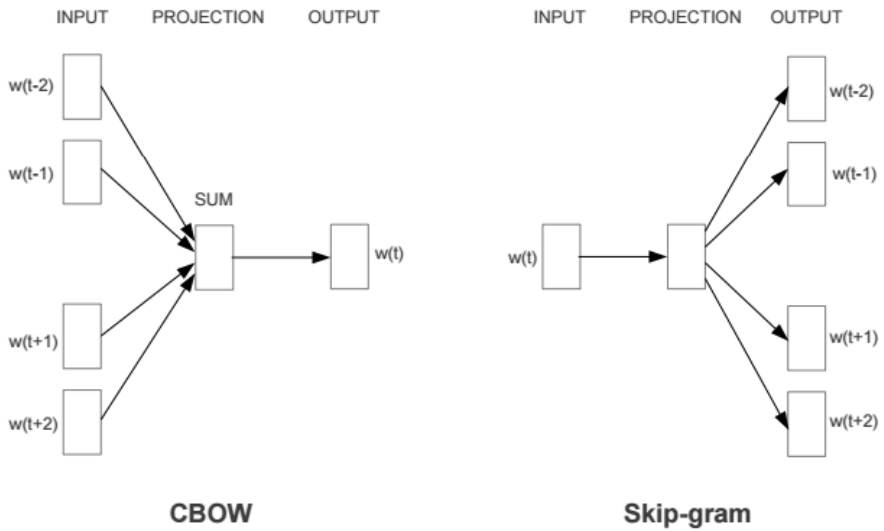
Idea: Maximize the probability of any context word given the current center word:

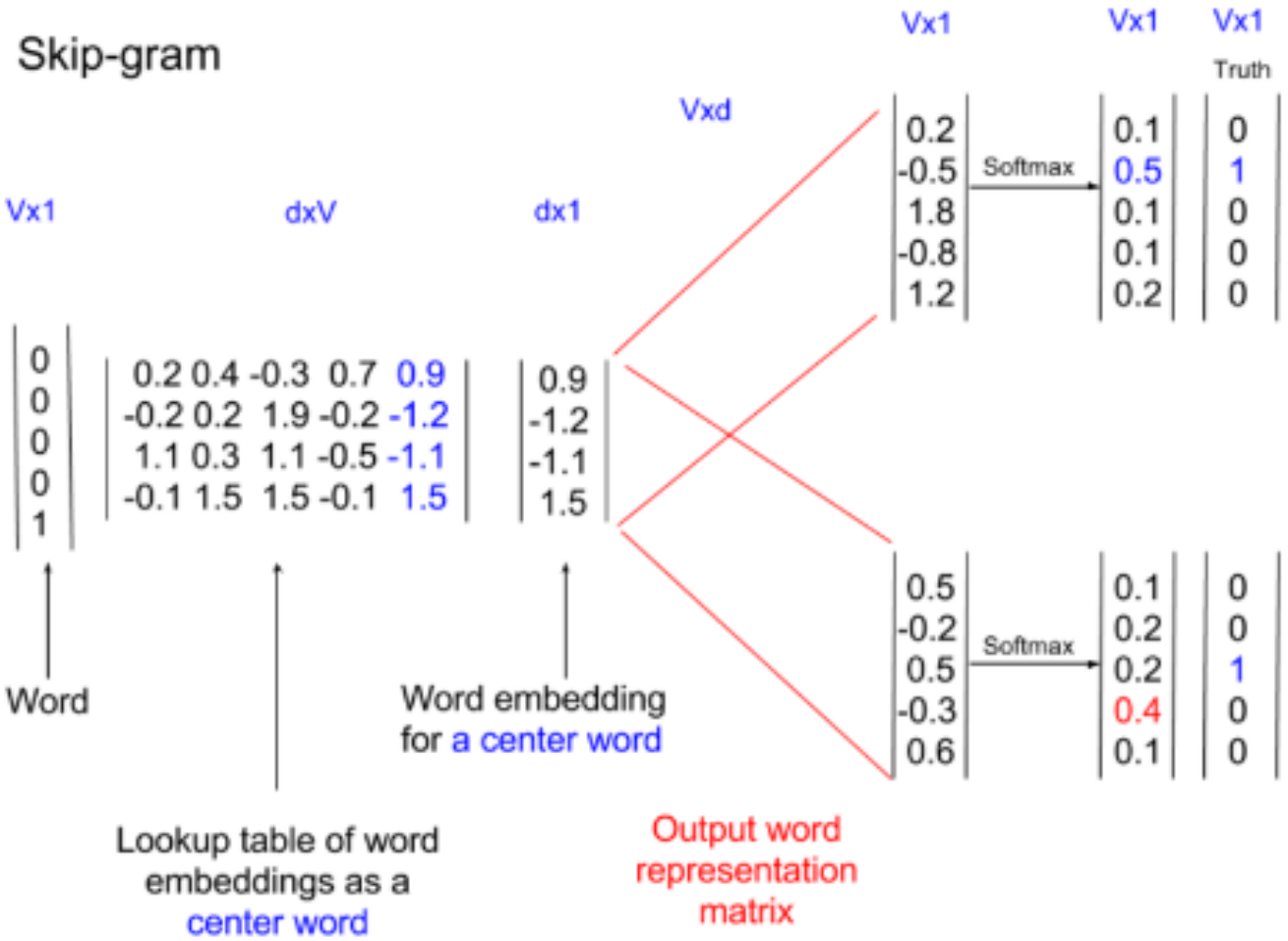
$$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m, \\ j \neq 0}} p(w_{t+j}|w_t;\theta)$$

- T - our text(corpus). For each word t=1...T, we try to predict surrounding words
- m - size of our window
- θ represents all variables we will optimize

Embeddings: Word2Vec

- **CBOW** model predict missing word (focus word) using context (surrounding words).
- **Skip gram** model predicts context based on the word in focus.
- Context is a fixed number of words to the left and right of the word in focus.
- Maximize average log probability of context words co-occurring with focus words.




Skip-gram
Vx1
dxV
dx1
Vxd
Vx1
Vx1
Vx1
Truth
0
0
0
0
1
0.2 0.4 -0.3 0.7 0.9
-0.2 0.2 1.9 -0.2 -1.2
1.1 0.3 1.1 -0.5 -1.1
-0.1 1.5 1.5 -0.1 1.5
0.9
-1.2
-1.1
1.5
0.2
-0.5
1.8
-0.8
1.2
Softmax
0.1
0.5
0.1
0.1
0.2
0
1
0
0
0
0.5
-0.2
0.5
-0.3
0.6
Softmax
0.1
0.2
0.2
0.4
0.1
0
0
1
0
0
Word
Word embedding
for a center word
Lookup table of word
embeddings as a
center word
Output word
representation
matrix

Pointwise Mutual Information

- Normalized skip gram probability
- Log probability of context word, x, co-occurring with target word y.

$$\mathrm{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)}$$

- Urbain, J., Bushnee, G., Knudson, Kowalski, G., P., Taylor, B. “Distributional Semantic Concept Models for Entity Relation Discovery,” NAACL VSM-NLP (National Association of Computational Linguistics workshop on Vector Space Modeling in Natural Language Processing), June 2015.
- Levy, O., and Goldberg, Y., Neural Word Embedding as Implicit Matrix Factorization, NIPS, 2014.
- Urbain, J. A Distributed Dimensional Indexing Model for Information Retrieval and Extraction. NLM/NIBIB Workshop on Natural Language Processing: State of the Art, Future Directions and Applications for Enhancing Clinical Decision-Making, (April 23-24, 2012).

GloVe Word Embeddings

1. Collect word co-occurrence statistics in a word co-occurrence matrix X. Where each element Xij represents how often work I appears in context of word j.
2. Define soft constraints foreach word pair. Word vectors ~= log probability of co-occurrence.

$$w_i^T w_j + b_i + b_j = log(X_{ij})$$

3. Define a cost function to optimize.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

f(X) is a weighting function to prevent learning only extremely common word pairs.

https://nlp.stanford.edu/projects/glove/

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^{\alpha} & \text{if } X_{ij} < XMAX \\ 1 & \text{otherwise} \end{cases}$$

FastText

- Each word is represented as a bag of character n-grams.
- Words are represented as the sum of their character n-grams.
- Captures morphology of words.
- Provides a representation for rare and out-of-vocabulary words.
- Very fast.
- Outperforms Word2Vec and GloVe in most benchmarks.

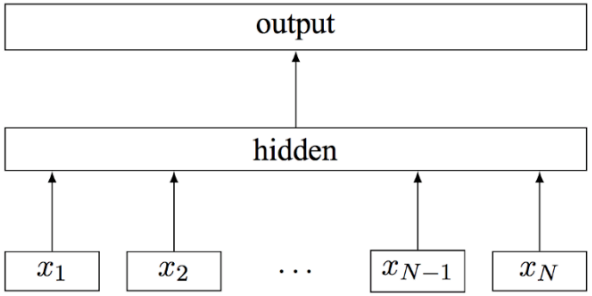


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

Image taken from the original paper

E.g., tri-grams for the word *apple* are *{_ap, app, ppl, ple, le_}*.

Bojanowski, P., Grave, E., Jouln, A, Mikolov, T. Enhancing Wore Vectors with Subword Information, 2017.

Selected word embedding: GloVe + Character Embedding

- Embedding for each word *w* is created by concatenating pretrained *p1=300* GloVe vector (Pennington et al., 2014) and the word's character embedding.
- GloVe word vectors are fixed during training.
- All the out-of-vocabulary words are mapped to a token, whose embedding is trainable with random initialization.
- Character embedding:
    - Each character is represented as a trainable vector of dimension *p2 = 200*, meaning each word can be viewed as the concatenation of the embedding vectors for each of its characters.
    - The length of each word is either truncated or padded to 16.
    - Take maximum value of each row of this matrix to get a fixed-size vector representation of each word.
    - Finally, the output of a given word *x* from this layer is the concatenation [xw; xc] ∈ Rp1+p2 , where xw and xc are the word embedding and the convolution output of character embedding of x respectively.
- Adapted from Seo et al., 2016, and Srivastava et al., 2015).

Problems with word embeddings?

Synonyms:

- River *bank*, *Bank* shot, *bank* deposit all have the same word representations.

- *… more on learning general models of language later.*

Properties of embeddings

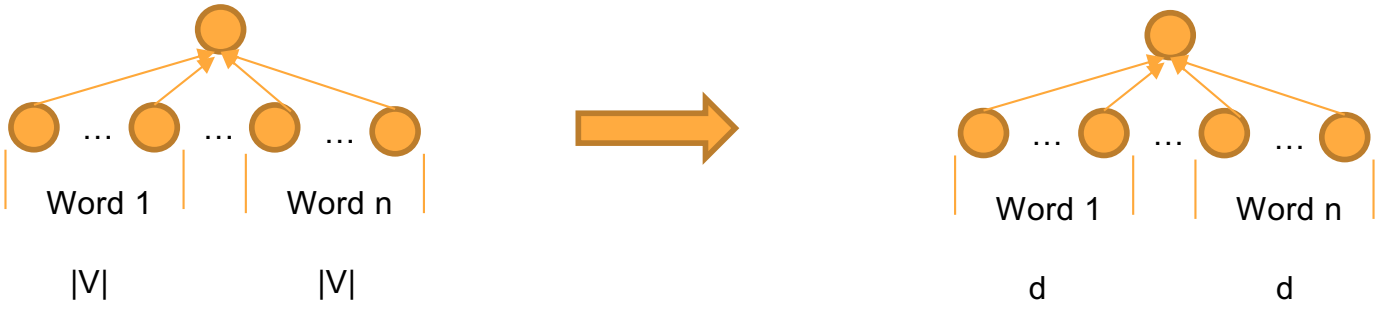
- They are dense: 0.53, 0.2, -1.2, ….
- They are low dimensional: (typically 50 <= d <= 300)
- Embed domain semantics:

  `king' - `man` + `woman` ≈ `queen`

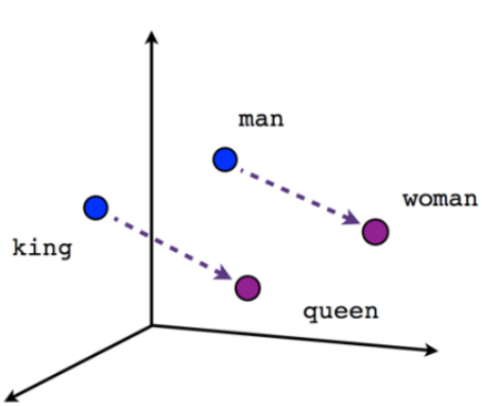
  `paris` - `france` + `spain` ≈ `madrid`
- Generalize easily!

Properties of embeddings (cont.)

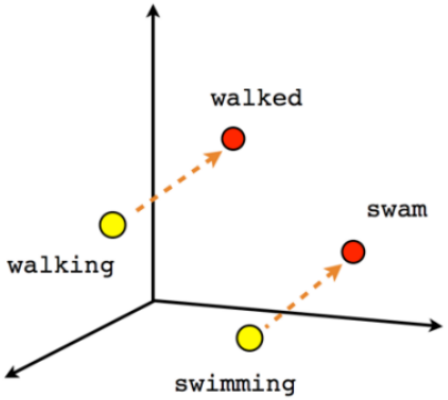
- They are used as input across a variety NLP tasks (transfer learning!):



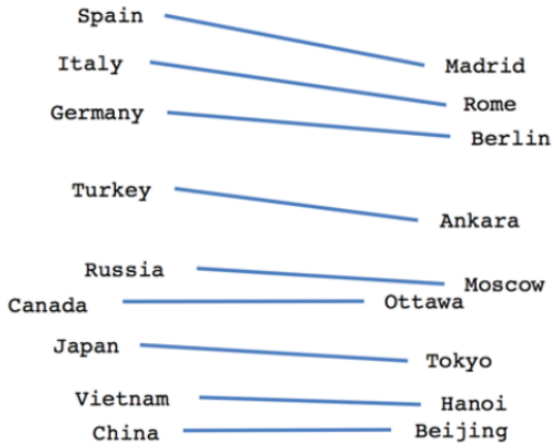
Total # of input parameters:	n * |V|	>>	n * d



Male-Female



Verb tense



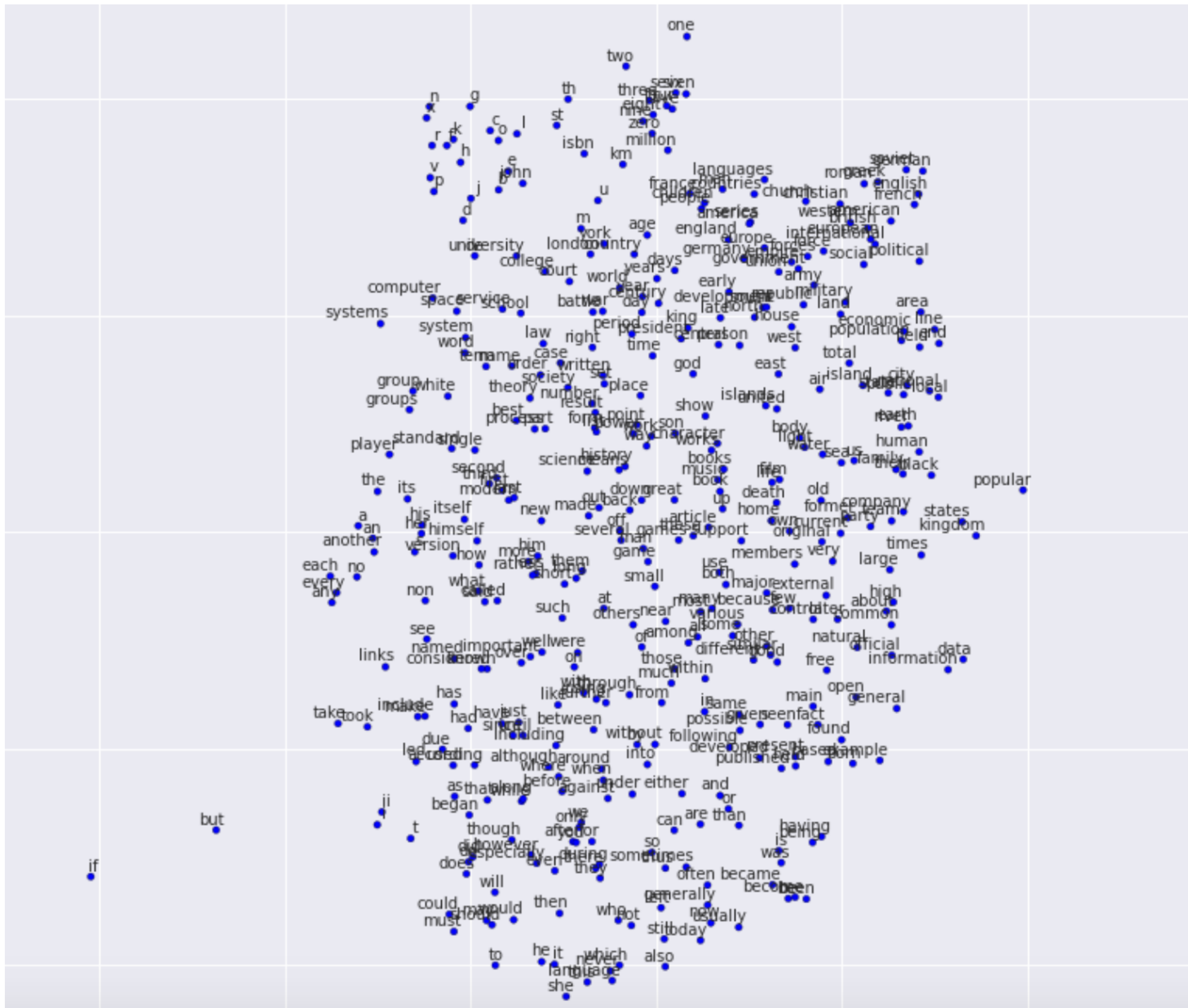
Country-Capital

https://www.tensorflow.org/tutorials/representation/word2vec

Word2Vec Galaxy

- https://www.youtube.com/watch?v=SnmQdKq_2hk

https://www.youtube.com/watch?v=SnmQdKq_2hk



one
two
th
n
g
isbn
million
zero
km
languages
christian
english
french
american
political
age
england
europe
social
computer
systems
service
college
court
world
army
military
area
battle
king
house
law
right
period
time
population
system
word
west
written
god
east
total
island
society
place
theory
groups
show
islands
player
process
body
human
science
history
books
music
works
black
popular
second
modern
death
old
company
states
kingdom
itself
new
made
home
himself
another
version
original
times
each
every
no
rather
short
small
members
large
non
such
at
others
near
various
major
external
high
see
common
links
considered
natural
official
data
information
free
has
main
open
general
take
took
had
between
possible
fact
found
due
following
example
although
around
either
and
as
began
but
can
than
having
if
though
often
became
generally
usually
then
could
must
will
today
still
also
to
he
she

USING WORD EMBEDDINGS

1. Learn an Embedding
   - Use embedding layer in your deep learning model

2. Reuse an Embedding
   - Download pre-trained word vectors

Lab 2 – Word Representations