Formal Models of Language

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge

You shall know a word by the **company** it keeps—Firth

Consider the following sentences about the *rabbit* in *Alice in Wonderland*:

- *Suddenly a white rabbit with pink eyes ran close by her.*
- *She was walking by the white rabbit who was peeping anxiously into her face.*
- *The rabbit actually took a watch out of its waistcoat pocket and looked at it.*
- *'Oh hush', the rabbit whispered, in a frightened tone.*
- *The white rabbit read out at the top of his shrill little voice the name Alice.*

We learn a lot about the rabbit from the words in the local context.

You shall know a word by the **company** it keeps—Firth

- So far, we have been discussing grammars with discrete alphabets and algorithms that have discrete symbols as input.
- Many Natural Language Processing tasks require some notion of similarity between the symbols.

  e.g. *The queen looked angry. Her majesty enjoyed beheading.*

  To understand the implication of these sentences we need to know that *the queen* and *her majesty* are similar ways of expressing the same thing.
- Instead of symbols we can represent a word by a collection of key words from its context (as a proxy to its meaning)

  e.g instead of rabbit we could use

  rabbit = {white, pink, eyes, voice, read, watch, waistcoat, ...}

You shall know a word by the **company** it keeps—Firth

- But which key words do we include in the collection?
- We could look at a $\pm n$-word context **window** around the **target** word.
- We could select (and weight) keywords based on their frequency in the window:

  rabbit = {the 56, white 22, a 17, was 11, in 10, it 9, said 8, and 8, to 7...}

- This would become a little more informative if we removed the function words:

  rabbit ={white 22, said 8, alice 7, king 4, hole 4, hush 3, say 3, anxiously 2...}

  queen ={said 21, king 6, shouted 5, croquet 4, alice 4, play 4, hearts 4, head 3... }

  cat ={said 19, alice 5, cheshire 5, sitting 3, think 3, queen 2, vanished 2, grin 2...}

- This is all just illustrative, we can of course, do this for all words (not just the characters)— called distributional semantics.

We can replace symbols with **vector representations**

- Two words can be expected to be semantically similar if they have similar word co-occurrence behaviour in texts.

  e.g. in large amounts of general text we would expect *queen* and *monarch* to have similar word co-occurrences.
- Simple collections of context words don't help us easily calculate any notion of similarity.
- A trend in modern Natural Language Processing technology is to replace symbolic representation with a **vector representation**
- Every word is encoded into some vector that represents a point in a multi-dimensional word space.

	alice	croquet	grin	hurried	king	say	shouted	vanished
rabbit	7	0	0	2	4	3	0	1
queen	4	4	0	1	6	1	5	0
cat	5	1	2	0	0	0	0	2

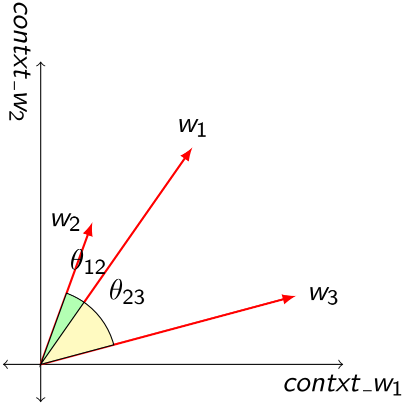
We can replace symbols with **vector representations**

- Note that there is an issue with polysemy (words that have more than one meaning):

- E.g. we have obtained the following vector for cat:
  cat = [5, 1, 2, 0, 0, 0, 0 2]

- But cat referred to two entities in our story:
  I wish I could show you our cat Dinah
  I didn't know that Cheshire cats always grinned in fact I didn't know that cats could grin

- The vector provides the coordinates of point/vector in the multi-dimensional word space.
- Assumption: **proximity in word space** correlates with **similarity** in meaning
- Similarity can now be measured using distance measures such as Jaccard, Cosine, Euclidean...



- e.g. cosine similarity
  $\text{cosine}(\mathbf{v_1}, \mathbf{v_2}) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\| \|\mathbf{v_2}\|}$
- Equivalent to dot product of normalised vectors (not affected by magnitude)
- cosine is 0 between orthogonal vectors
- cosine is 1 if $v_1 = \alpha v_2$, where $\alpha > 0$

Automatically derived vectors will be very **large** and **sparse**

- In certain circumstances we might select dimensions **expertly**
- For general purpose vectors we want to simply count in a large collection of texts, the number of times each word appears inside a window of a particular size around the target word.
- This leads to very **large sparse** vectors (remember Zipf's law)
- There are an estimated 13 *million* tokens for the English language—we can reduce this a bit by removing (or discounting) function words, grouping morphological variants (e.g, *grin*, *grins*, *grinning*)
- Is there some k-**dimensional space** (such that $k << 13 million$) that is sufficient to encode the word meanings of natural language?
- Dimensions might hypothetically encode tense (past vs. present vs. future), count (singular vs. plural), and gender (masculine vs. feminine)...

It is possible to **reduce** the **dimensions** of the vector

To find reduced dimensionality vectors (usually called **word embeddings**)

- Loop over a massive dataset and accumulate word co-occurrence counts in some form of a large sparse matrix X (dimensions n x n where n is vocabulary size)
- Perform **Singular Value Decomposition** on X to get a USV^T decomposition of X.

$$\begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & X & \vdots \\ x_{n1} & \dots & x_{nn} \end{bmatrix} = \begin{bmatrix} u_{11} & \vdots & \vdots & \vdots \\ \vdots & u_2 & \dots & u_n \\ u_{1n} & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 & \dots \\ 0 & s_2 & 0 & \dots \\ 0 & 0 & \ddots & \dots \\ \vdots & \vdots & \vdots & s_n \end{bmatrix} \begin{bmatrix} v_{1n} & \dots & v_{1n} \\ \dots & v_2 & \dots \\ \dots & \vdots & \dots \\ \dots & v_n & \dots \end{bmatrix}$$

It is possible to **reduce** the **dimensions** of the vector

- Note S matrix has diagonal entries only.
- Cut diagonal matrix at index k based on desired dimensionality (can be decided by desired percentage variance): $(\sum_{i=1}^{k} s_i)/(\sum_{i=1}^{n} s_i)$

$$\begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & X' & \vdots \\ x_{n1} & \dots & x_{nn} \end{bmatrix} = \begin{bmatrix} u_{11} & \vdots & \vdots \\ \vdots & \dots & u_k \\ u_{1n} & \vdots & \vdots \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & s_k \end{bmatrix} \begin{bmatrix} v_{1n} & \dots & v_{1n} \\ \dots & \vdots & \dots \\ \dots & v_k & \dots \end{bmatrix}$$

- Use rows of U for the word embeddings.
- This gives us a k-dimensional representation of every word in the vocabulary.

It is possible to **reduce** the **dimensions** of the vector

Things to note:

- Need all the counts before we do the SVD reduction.
- The matrix is extremely **sparse** (most words do not co-occur)
- The matrix is very **large** ($\approx 10^6 x 10^6$)
- SVD is **quadratic**

Points of methodological variation:

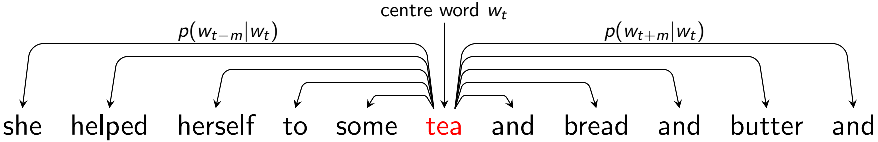
- Due to Zipf distribution of words there is **large variance** in co-occurrence frequencies (need to do something about this e.g. discount/remove stop words)
- Refined approaches might **weight** the co-occurrence counts based on distance between the words

Predict models can be more efficient than **count** models

- **word2vec** is a **predict model**, in contrast to the distributional models already mentioned which are **count models**.
- Instead of computing and storing a large matrix from a very large dataset, use a model that learns **iteratively**, eventually encoding the probability of a word given its context.
- The parameters of the model are the word embeddings.
- The model is trained on a certain objective.
- At every iteration we run our model, evaluate the errors, and then adjust the model parameters that caused the error.

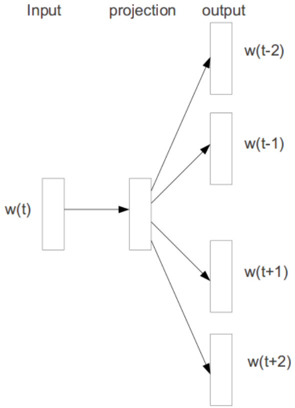
Predict models can be more efficient than **count** models

- There are two main word2vec architectures:
- **Continuous Bag of Words** CBOW: given some context word embeddings, predict the target word embedding.
- **Skip-gram**: given a target word embedding, predict the context word embeddings (below).



- **skip-gram** model predicts relationship between a centre word w_t and its context words: $p(context|w_t) = \ldots$
- Predict context word embeddings based on the target word embedding.
- A loss function is used to score the prediction (usually **cross-entropy** loss function).

  (Cross-entropy measures the information difference between the expected word embeddings and the predicted ones.)
- Adjust the word embeddings to minimise the loss function.
- Repeat over many positions of t in a very big language corpus.



Distributional models have improved NLP applications

In general, distributional models have had a positive impact on NLP and provided improvement over symbolic systems:

- There has been a change in state-of-the-art for some applications: (e.g. Google Translate)
- Multi-modal experiments have become more straightforward (by combining vector representations)
- But these models are statistical (need very large amounts of data and have to find a way to handle unseen words)
- There has been a lot of hype and not much work on the problems the distributional models can't solve.

Predict models versus **count** models

- $+$ Predict models can be more efficient than count models because we can learn **iteratively** and don't have to hold statistics on the whole dataset.
- $-$ Need to **initialise** the word embeddings (several possible methods).
- $\pm$ The size of the embeddings is a chosen parameter of the system (usually a few hundred).
- $+$ Predict models are learning structure **without hand-crafting** of features.
- $-$ Dimensionality of the embeddings are assumed to capture meaningful generalisations, but the dimensions are **not directly interpretable**.

Predict models versus **count** models

- After training, predict models are found to be equivalent to a count model with dimensionality reduction.
- Tuning hyper-parameters is a matter of much (often brute-force) experimentation.
- Predict models perform better than count models with dimensionality on some tasks (but perhaps due to tuning hyper-parameters).
- For some tasks count vectors without dimensionality reduction are the most effective.

Word embeddings can **correlate** with human **intuitions**

Researchers test their word embeddings against datasets of **human similarity judgements**:

- For a test set of words, participants rate word pairs for **relatedness** (e.g. Miller & Charles, Rubenstein & Goodenough)
- A rank of relatedness can be drawn up between items in the test set.
- A **rank correlation** between embeddings and human judgements can be calculated.
- Good embeddings have a correlation of 0.8 or better with the human judgements.

Reasoning may be possible based on word embeddings

- *Mikolov et al.* analogy puzzles:

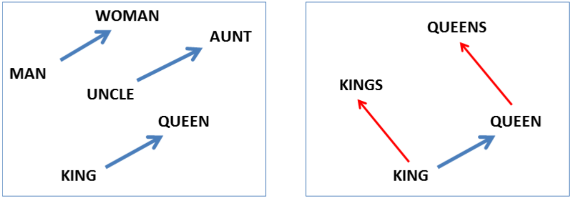
  Can we use word embeddings to solve puzzles such as:

  man is to woman as king is to .... queen

- Can we do vector-oriented reasoning based on the offsets between words?

Reasoning may be possible based on word embeddings

- Derive the vector between the pair of words *man* and *woman* and then add it to *king*.
- The nearest word to the region of vector space that results will be the answer to the analogy.



- *Mikolov* found that word2vec embeddings are good at capturing syntactic and semantic regularities in language, and that each relationship is characterised by a relation-specific vector offset.
- Note that the space is very sparse and that there are word pairs for which this does not work...

Relationship between embeddings and **brain activity**?

- Humans have the capacity to translate thoughts into words, and to infer others' thoughts from their words.
- There must be some mental representations of meaning that are mapped to language, but we have no direct access to these representations.

$$\text{—}\ W \rightarrow \boxed{\textit{encoder}}\text{—}\ X \rightarrow \boxed{\begin{array}{c}\textit{channel}\\ p(y|x)\end{array}}\text{—}\ Y \rightarrow \boxed{\textit{decoder}}\text{—}\ W' \rightarrow$$

mental representation | *words* | *words′* | *mental representation′*

- Do word embeddings provide a model that successfully captures some aspects of our mental representation of meaning?

Relationship between embeddings and **brain activity**?

- Natural language appears to be a discrete symbolic system.
- The brain encodes information through continuous signals of activation.
- Language symbols are transmitted via continuous signals of sound/vision.

- *Pereira et al.* trained a system using brain imaging data and word embeddings.
- Demonstrated the ability to generalise to new meanings from limited imaging data.

`https://www.nature.com/articles/s41467-018-03068-4`

Relationship between embeddings and **brain activity**?