L101: Machine Learning for Language Processing

Lecture 6

Guy Emerson

Today's Lecture

- Distributional semantics
  - Count vectors
  - Embedding vectors
- Challenges for distributional semantics

Distributional Semantics

... being hurt by another	horse	especially if some rider ...
... beaten by a better	horse	at the distance on ...
... these studies that	horses	reared with other ...
... reared with other	horses	in a free and ...
... 'Is that all your	horse	gets to eat?' in ...
... cache of cattle and	horse	bones, while from the ...
... was a sterling good	horse,	especially at Ascot ...
... way as a domestic	horse	that it is stabled ...
... 1790 – that is, one	horse	or two cows for ...
... as coarse as a	horse	's tail straying from ...

The aim of distributional semantics is to learn the meanings of linguistic expressions from a large corpus of text. The core idea, known as the *distributional hypothesis*, is that the contexts in which an expression appears give us information about its meaning. The hypothesis is often stated more narrowly, to say that similar words will appear in similar contexts.

Above are ten instances of *horse* in the British National Corpus. From these, we might learn that horses are animals used in racing and agriculture. Some relevant context words are highlighted.

Distributional semantics is an unsupervised task – we observe a corpus (without any annotated outputs), and we try to find semantic structure in the corpus.

Distributional Semantics

- Linguistic motivation: understand language
  - Harris (1954)
  - Firth (1951, 1957)
- Machine learning motivation: text is cheap

Harris and Firth and widely cited in the NLP literature, but I suspect most citing authors have not actually read these papers, and would be surprised to learn that Firth uses distributional ideas to analyse poetry. These papers are interesting for understanding the linguistic motivations, but not useful for understanding modern techniques. To read them, you will probably have to find the physical books (!)

Harris (1954) “Distributional Structure”, reprinted in: Harris (1970) “Papers in Structural and Transformational Linguistics”, chapter 36; and Harris (1981), “Papers on Syntax”, chapter 1.

Firth (1951) “Modes of meaning”, reprinted in: Firth (1957) “Papers in Linguistics”, chapter 15.

Firth (1957) “A synopsis of linguistic theory 1930–1955”, chapter 1 of “Studies in Linguistic Analysis”, special volume of the Philological Society.

Context

ASH-993: ... saying ‘Is that all your horse gets to eat?’ in amazement ...

- Word windows (saying, your, eat, ...)
- Dependencies (your-POSS, get-SUBJ)
- Documents (ASH-993)

To use distributional semantics, we need to formalise the notion of a context. Many choices are possible.

ASH-993 is the document identifier from the BNC. Document-level contexts are historically important, because early distributional techniques were developed for document retrieval. For example, Spärck-Jones (1964, reprinted 1986) “Synonymy and Semantic Classification”.

Word Window Hyperparameters

- Window size
- Lemmatisation?
- Stop list?
- Rare words?

Even for the simple approach of word windows, there are many pre-processing decisions, which can have an important impact on results. And remember, this is before we even get to the model itself!

Some techniques, like lemmatisation, lose information, but also reduce data sparsity.

Count Matrix

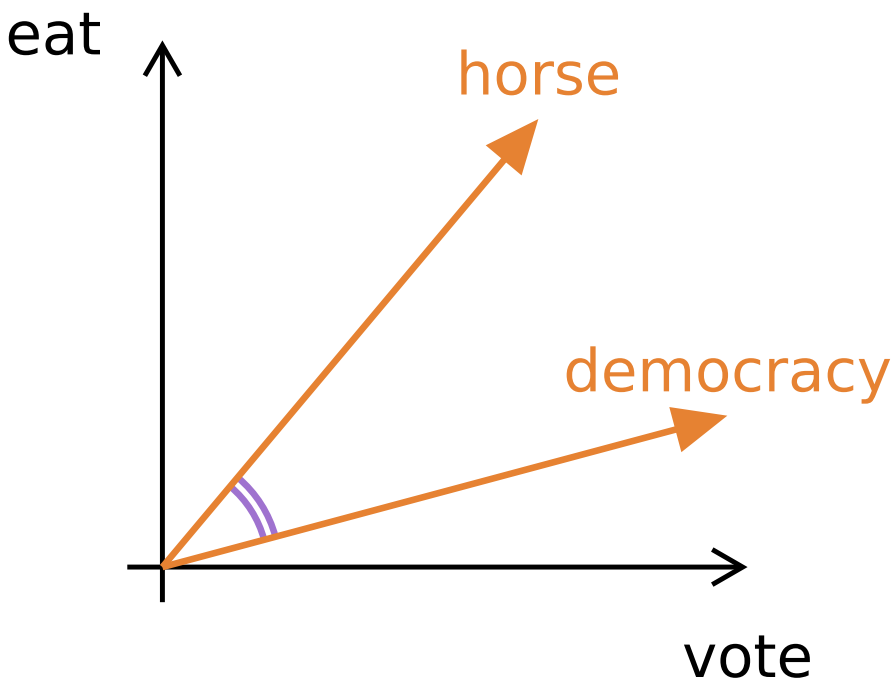
$$\text{target words} \begin{pmatrix} n_{11} & n_{12} & n_{13} & \dots & n_{1D} \\ n_{21} & n_{22} & n_{23} & \dots & n_{2D} \\ n_{31} & n_{32} & n_{33} & \dots & n_{3D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{V1} & n_{V2} & n_{V3} & \dots & n_{VD} \end{pmatrix} \text{contexts}$$

Entry n_{ij} is the number of times we observe the target word i with the context j.

If contexts are words, this will be a square matrix – but we may also use other kinds of context.

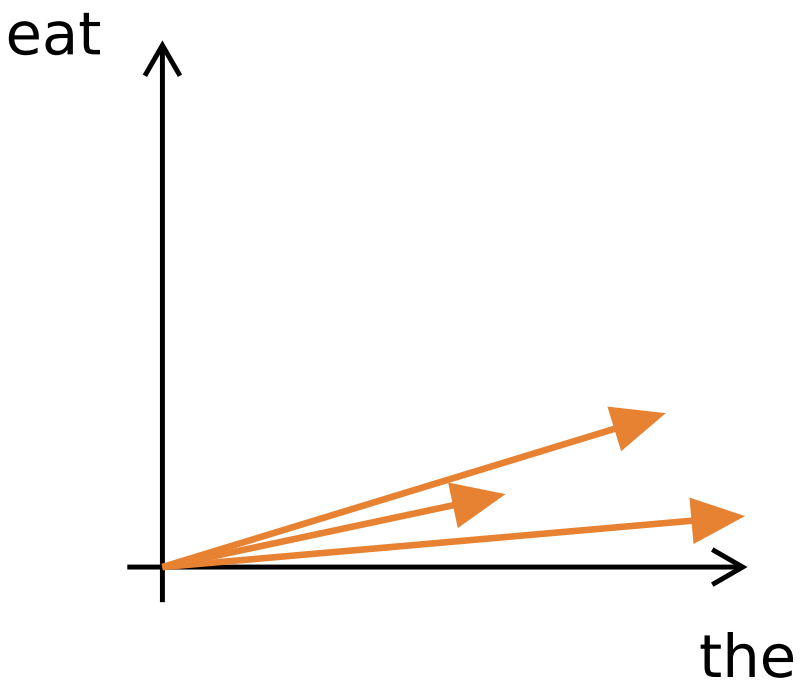
We can view each row as a vector, where each context defines a dimension.

Count Vectors



By representing the meanings of words as vectors, we get a notion of distance (or its inverse, similarity). For example, we could measure the angle between two vectors – the smaller the angle, the more similar the vectors. The cosine of the angle (cosine similarity) can be easily calculated using a dot product.

Count Vectors



If we directly use the counts, common contexts will dominate, even though they are not terribly informative – if we use word windows, almost all words will occur in the context of *the*.

Processing the Counts – TF-IDF

$$v_{ij} = \frac{n_{ij}}{\left|\left\{i' : n_{i'j} > 0\right\}\right|}$$

- Used in document retrieval
- TF: term frequency
- IDF: inverse document frequency

To avoid common contexts from dominating, we need to reweight the counts. Ideally, we want high values for informative contexts, rather than common contexts. A simple method is TF-IDF, which was developed for information retrieval.

Each count ("term frequency") is divided by ("inverse...") the number of target words the context appears with ("...document frequency").

Common contexts are observed with many target words (they have a high document frequency), and so they are heavily downweighted. Rare contexts are only mildly downweighted, and so they become more important.

TF-IDF is easy to understand, but there are many other ways that we could transform the counts. One transformation, which has a strong information-theoretic underpinning, is Pointwise Mutual Information – see next slide!

Processing the Counts – PMI

$$v_{ij} = \log \frac{n_{ij} n_{..}}{n_{i.} n_{.j}}$$

- Pointwise Mutual Information (from information theory)
- $\log \frac{P(x,y)}{P(x)P(y)}$ – more likely than expected?

$n_{i.}$ is the sum for row i
$n_{.j}$ is the sum for column j
$n_{..}$ is the sum for the whole matrix

This views the observations of target-context pairs as samples from a joint distribution over target words and contexts. Intuitively, a high probability of observing a particular combination (i.e. a large count for that combination) is not surprising if the target and context are both frequent. So, we can compare the probability of observing a combination with the marginal probability of the target and context separately. If the joint probability $P(x, y)$ is exactly the same as the product $P(x)P(y)$, this means the combination is just as likely as we would expect by chance, and we get a PMI of 0. If the combination is more likely than we would expect, we get a positive PMI, and if the combination is less likely than we would expect, we get a negative PMI.

After transforming the counts using PMI, we have high values for informative contexts, in an information-theoretic sense.

Processing the Counts – PPMI

$$v_{ij} = \max\left\{0, \log \frac{n_{ij} n_{..}}{n_{i.} n_{.j}}\right\}$$

- Pointwise Mutual Information (from information theory)
- $\log \frac{P(x,y)}{P(x)P(y)}$, positive only
- Avoids negative infinities

This formula uses the observed counts to estimate the probabilities. However, we've seen before that zero counts can be a problem – here, zero counts lead to negative infinities.

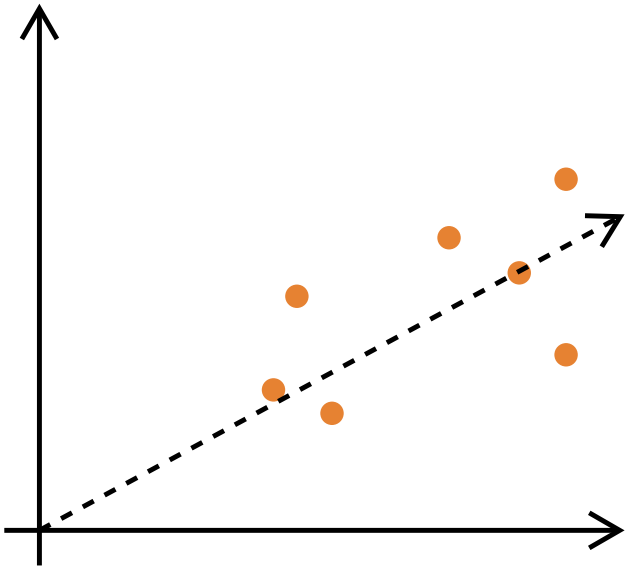
One solution would be to add smoothing. Another solution, is to use PPMI (positive PMI), where we set the value to zero, if the PMI is negative. For distributional semantics, this is much more commonly used than smoothing.

As well as removing the problem of negative infinities, this also means that we are only recording *positive associations* between targets and contexts. The PMI is positive if a target-context combination is observed more than expected, and we keep these values. The PMI is negative if it's observed less than expected, and all of these values are set to zero. In other words, we forget about all negative associations, and only remember the positive associations.

Singular Value Decomposition

- High dimensions difficult to work with
- Find directions of highest variance
- Only use these directions

Singular Value Decomposition



In this toy example, we can reduce this two-dimensional space to a one-dimensional space, by only looking at the direction along the dotted line. The plotted points vary a lot in this direction, and don't vary as much in the orthogonal direction.

Because we are working with a vector space, and looking for a linear directions, we can re-express the aim of finding high-variance directions in terms of a matrix factorisation. (If everything is linear, the maths will usually work out nicely.)

Embedding Vectors

- Directly learn lower-dimensional vectors
- Never construct count matrix

Using PPMI and SVD gives us useful low(ish)-dimensional vectors, but they are calculated via a high-dimensional count matrix. Can we learn low-dimensional vectors directly?

Skip-Gram

- Observe target-context pairs (t, c)
- Treat as classification:
  predict context, given target
- Discriminative classifier:
  $P(c \mid t) \propto \exp(v_t \cdot u_c)$
- Like logistic regression, but:
  "input vectors" are learnt, not given

Skip-gram starts from the same place as (P)PMI count vectors – we treat the training data as target-context pairs. The crucial idea is to use tools from supervised learning, even though this is an unsupervised task. We will see this idea again next lecture.

Here, we use a log-linear model, like logistic regression. We learn a vector v_t for each target, and a vector u_c for each context.

(If contexts are words, note that we learn *two* vectors for each word, once as a target and once as a context.)

Skip-Gram & Negative Sampling

- $P(c \mid t) \propto \exp(v_t \cdot u_c)$
  requires *all* possible contexts
- Instead, sample a few other contexts c'
- Treat as binary classification:
  predict if context is real or sampled

The classifier on the previous slide is expensive to calculate:

$$P(c \mid t) \propto \exp(v_t \cdot u_c) = \frac{\exp(v_t \cdot u_c)}{\sum_{c'} \exp(v_t \cdot u_{c'})}$$

Instead, we can use a slightly different task. For each observed context, we sample a number of other contexts, called "negative" contexts. This means we effectively have two parts to our training data – the real observed data, and the negatively sampled data.

The task is now: given a target-context pair, predict whether this pair came from the real training data or the negatively sampled data.

There will be a small chance of sampling a real context, but most of the negative samples will not have been observed.

The number of samples we take for each real observation is a hyperparameter.

Skip-Gram & Negative Sampling

- $P(\text{real} \mid t, c) \propto \exp(v_t \cdot u_c)$
- $P(\text{sampled} \mid t, c) \propto 1$
- $P(\text{real} \mid t, c) = \sigma(v_t \cdot u_c) = \frac{1}{1+\exp(-v_t \cdot u_c)}$

Again, we use a log-linear model.

The reason that we set $P(\text{sampled} \mid t, c) \propto 1$ is because log-linear models can be rescaled, as explained in lecture 3.

σ is called the *sigmoid* function. The formula above follows immediately from the unnormalised probabilities in the first two lines. As the value of the dot product tends to infinity, the probability tends to 1, and as the dot product tends to negative infinity, the probability tends to 0. If the dot product is exactly 0, the probability is exactly ½.

(It's called a sigmoid because the shape of the graph is vaguely "s-shaped", particularly compared to historical versions of *s* that look more like $\int$. Some versions of the letter sigma (σ) look more like *s*, but the function is usually written with the symbol σ.)

Skip-Gram & Negative Sampling

- Want high: $v_t \cdot u_c$
- Want low: $v_t \cdot u_{c'}$

Skip-gram with negative sampling is trained to maximise the log-probability of the correct predictions (predicting real or sampled).

This means that we want a high dot product for observed target-context pairs, and a low dot product for unobserved target-context pairs.

Further reading:

The original paper (Mikolov et al., 2013) got scathing reviews `https://openreview.net/forum?id=idpCdOWtqXd60`

A more careful explanation of the model is given by Goldberg and Levy (2014) `https://arxiv.org/pdf/1402.3722.pdf` (careful except for D being used with two meanings).

Count vs. Embedding

- Skip-gram approximately factorises a PMI matrix!
- Hyperparameters important

At first sight, Skip-gram looks very different from count vectors – but because both Skip-gram and (P)PMI vectors start by looking at the data as a set of target-context pairs, there is a close connection between the two.

Before this connection was discovered, it was widely believed that Skip-gram was more effective than count vectors, but the difference in performance is down to a number of hyperparameters (as shown by Levy et al., to be discussed in the reading group).

Levy and Goldberg (2014) http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf

Li et al. (2015) http://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/download/10863/11249

Cotterell et al. (2017) http://aclweb.org/anthology/E17-2028

Evaluation

- Lexical semantics
- Compositional semantics
- Downstream tasks

Finding a good evaluation task is a real challenge for distributional semantics.

The simplest tasks look at lexical semantics, for example word similarity (see next slide).

The other extreme is to use distributional vectors in a downstream task. However, it can be difficult to isolate the effect of distributional vectors, separate from the rest of the system. It can also be difficult to find tasks where detailed semantic representations are necessary.

In the middle, there are a nubmer of datasets which try to measure various aspects of meaning at a phrase level. Designing such datasets is challenging, and this is an ongoing area of research.

Lexical Semantics

democracy	*water*	*happiness*
aubergine	*flood*	*joy*
computer	*law*	*cat*
earthquake	*lawyer*	*dog*

The pairs on the left are unrelated.

The pairs on the right are *similar* – they are near synonyms (*happiness* and *joy* are often inter-changeable), or co-hyponyms (dogs and cats are both animals and both pets).

The pairs in the middle are *related* but not similar – there is some connection between them (a flood is an overflow of water, and a lawyer practises law), but the words are not interchangeable, and not co-hyponyms.

Different similarity datasets target relatedness, similarity, or both.

Lexical Semantics

- Give annotators pairs of words
- Ask to score (e.g. from 1 to 7)
- Get system's similarity scores
- Measure Spearman rank correlation

Depending on the instructions given to annotators, the scores may reflect similarity or relatedness.

Spearman rank correlation measures whether the scores are in the same relative order – e.g. if annotators judge the pair (a,b) to be more similar than the pair (c,d), we want the system to do the same, but the exact numerical values don't matter.

Challenges for Dist. Sem.

- Grounding
- Lexical Semantics
  - Word senses
  - Hyponymy
- Sentence Semantics
  - Composition
  - Logic

Everything presented so far in this lecture is relatively well-established. However, there are still many open research questions. For further discussion, see chapter 2 of my PhD thesis:
`https://www.cl.cam.ac.uk/~gete2/thesis.pdf`

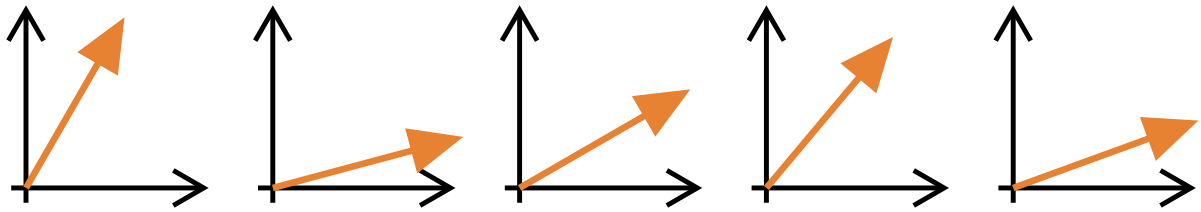
Word Senses

... the last kick of the	match.	It was entertaining ...
... the Duddon are no	match,	after all, for a route ...
... first or second round	matches	of any consequence ...
... Tried soaking the	matches	in paint, he wrote, ...
... is very much a	match	for Berowne; this is ...
... to win and the	match	is therefore ...
... to lose you the	match	even though no ...
... of an elimination	match	is fought. If this ...
... needed to watch the	match,	needed a diversion ...
... drop in a burning	match.	The plastic of the ...

In the example we saw last lecture, there are different senses (clusters of usages) – these will all be combined when constructing a distributional vector.

How should a distributional model deal with this?

Semantic Composition



- Addition?
- Componentwise multiplication?
- Linguistically-motivated approach?

If we have a vector for each word, how do we combine these to get a representation for the whole phrase?

Vector addition and componentwise multiplication are competitive methods for many tasks, but we know that they are not enough, because they are insensitive to word order – we would get the same representation for “every picture tells a story”, “every story tells a picture”, and “story picture a every tells”.

Are vectors even the right kind of representation?

Summary

- Distributional semantics – context
- Count models
  - PPMI, SVD
- Embedding models
  - Skip-gram with negative sampling
- Similarity and relatedness
- Challenges – word senses, composition

Further reading:

Turney and Pantel (2010) https://www.jair.org/index.php/jair/article/view/10640/25440

Erk (2012) https://onlinelibrary.wiley.com/doi/epdf/10.1002/lnco.362

Clark (2015) https://www.cl.cam.ac.uk/~sc609/pubs/sem_handbook.pdf