Stanford CS224N NLP with Deep Learning

Winter 2021

Lecture 1 – Introduction and Word Vectors

How Do We Represent the Meaning of a Word?

The commonest linguistic way of thinking of meaning is the link between the signifier (the linguistic symbol) and the signified (the idea or thing). This is a *denotational semantics* approach.

To make meaning usable in a computer, we use a thesaurus-like organisation of words into a hierarchy of hypernyms and hyponyms, like Wordnet.
However, this has problems – it misses nuance, cannot keep up with new meanings, requires a lot of human labour and can't accurately compute word similarity.

Representing Words as Discrete Symbols

This approach is called a *localist* representation. Mathematically, this could correspond to representing each word as a distinct one-hot vector (a vector with 1 in one position and 0 in all others). This would make the number of dimensions equal to the vocabulary size.

This method, moreover, gives no clue as to the similarities among words – all the vectors are unit-sized and orthogonal.

Representing Words by Their Context

Under this approach, a word's meaning is taken to be given by those that frequently appear close by. This is a very successful idea in NLP.

The context of a word w occuring in a text is the set of words occurring nearby, in a fixed window. The many contexts of w are used to build its representation. Thus we build a dense vector for each word, chosen to be similar to the vectors of words appearing in similar contexts.

Word vectors are also called word embeddings or neural word representations. This is a *distributional* approach.

Word2Vec

Word2Vec is a framework for learning word vectors. It uses a large corpus, and goes through it, calculating the probabilities of centre words given context words. We adjust the word vectors to maximise this probability.

Objective Function

For each position $t \in \{1, \ldots, T\}$, we have to predict context words within a window of size m, given the centre word w_t. The *likelihood* is defined as

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

where θ represents all the parameters over which we are optimising.

The objective function J is the average negative log likelihood:

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

Minimising the objective function maximises predictive accuracy.

Calculation of Probability

In order to calculate $P(w_{t+j} \mid w_t; \theta)$, we use two vectors per word w: v_w when w is a centre word and u_w when it is a context word. This is done to simplify the optimisation process.

Then, for a centre word c and a context word o,

$$P(o \mid c) = \frac{\exp(u_o \cdot v_c)}{\sum_{w \in V} \exp(u_w \cdot v_c)}.$$

- Exponentiation makes anything positive.
- The dot product compares the similarity of o and c.
- We normalise over the entire vocabulary to get a probability distribution.

In fact, this is an example of the softmax function $\mathbb{R}^n \to (0, 1)^n$:

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

which maps arbitrary values x_i to a probability distribution p_i. It is frequently used in deep learning.

The Goal

To train the model, we gradually adjust the parameters (in this case, the word vectors themselves) to minimise the loss given by the objective function.

We carry this out by walking down the gradient w.r.t each vector component. That is, if the gradient of the object function w.r.t the i^{th} component of the vector is negative, we increase that component to decrease the objective function.

$$
\begin{aligned}
\frac{\partial}{\partial v_c} \log p(o \mid c) &= \frac{\partial}{\partial v_c} \log \frac{\exp(u_o \cdot v_c)}{\sum_{w \in V} \exp(u_w \cdot v_c)} \\
&= \frac{\partial}{\partial v_c} \log \exp(u_o \cdot v_c) - \frac{\partial}{\partial v_c} \log \sum_{w \in V} \exp(u_w \cdot v_c) \\
&= u_o - \frac{\partial}{\partial v_c} \log \sum_{w \in V} \exp(u_w \cdot v_c) \\
&= u_o - \frac{1}{\sum_{w \in V} \exp(u_w \cdot v_c)} \cdot \sum_{x \in V} \exp(u_x \cdot v_c) \cdot u_x \\
&= u_o - \sum_{x \in V} \frac{\exp(u_x \cdot v_c)}{\sum_{w \in V} \exp(u_w \cdot v_c)} u_x \\
&= u_o - \sum_{x \in V} p(x \mid c) u_x \\
&= \text{observed} - \text{expected},
\end{aligned}
$$

since the second term is the average over the vectors of all words weighted by their probabilities.