1 NLU: Introduction

Synonymy: equivalence of near-equivalence in meaning, **polysemy**: multitude of meanings

Regarding polysemy, disambiguating the whole corpus would be cumbersome. Even more problematically, it is unclear what the meanings are anyway.

Raw dot-product, has a problem as a similarity metric: it favors long vectors. The dot product is higher if a vector is longer, with higher values in each dimension. The simplest way to modify the dot product to normalize for the vector length is to divide the dot product by the lengths of each of the two vectors. This normalized dot product turns out to be the same as the **cosine** of the angle between the two vectors.

Pointwise mutual information (is one of the most important concepts in NLP. It is a measure of how often two events x and y occur, compared with what we would expect if they were independent: $I(x, y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$. It is more common to use **Positive PMI** (called PPMI) which replaces all negative PMI values with zero.

Distribution similarity: Compare words by comparing their distributions over context words, e.g. KL divergence, Jensen-Shannon divergence

Two words have **first-order cooccurrence** (sometimes called **syntagmatic association**) if they are typically nearby each other. Thus wrote is a first-order associate of book or poem. Two words have **second-order cooccurrence** (sometimes called **paradigmatic association**) if they have similar neighbors. Thus wrote is a second-order associate of words like said or remarked.

Word embeddings are based on the distributional assumption that words appearing within similar context possess similar meaning.

Word2Vec tends to embed both syntactical and semantic information and it is very effective for the compositionality.

2 ML4H: Motivation

Discrete representations: use a taxonomy that has hypernyms (is-a) relationships and synonym sets (e.g. WordNet).
Problems with discrete representations:

- Missing new words - hard to keep up with the evaluation of the language.
- Requires human labor to create and adapt.
- Subjective.
- Hard to compute accurate word similarity.
- Dimensionality problems (one-hot encoding).

Problems with simple co-occurrence vectors:

- Increase in size with vocabulary. Very high dimensional: require a lot of storage.
- Subsequent classification models have sparsity issues. Less robust models as a result.
- Dimensionality reduction with SVD: Computational cost scales quadratically for $n * m$ matrix: $O(mn2)$, hard to incorporate new words or documents

Why use embeddings?

- Reduce dimensionality of representation.
- Encodes similarity information, useful for other tasks.
- Learn representations of entities (words) as well as relationships between them.

3 NLU: skip-gram architecture

The word2vec family of models, including skip-gram and CBOW, is a popular efficient way to compute dense embeddings.

The **intuition of word2vec** is that instead of counting how often each word w occurs near, say, apricot, we'll instead train a classifier on a binary prediction task: "Is word w likely to show up near apricot?" We don't actually care about this prediction task; instead we'll take the learned classifier weights as the word embeddings. The revolutionary intuition here is that we can just use running text as implicitly supervised training data for such a classifier. Skip-gram intuition:

- Treat the target word and a neighboring context word as positive examples.
- Randomly sample other words in the lexicon to get negative samples
- Use logistic regression to train a classifier to distinguish those two cases
- Use the regression weights as the embeddings

Softmax in log-bilinear model needs the computation of partition function. Noise contrastive estimation or negative sampling allows an alternative.

The noise words are chosen according to their weighted unigram frequency $p_\alpha w = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$, where α is a weight.

For $k = 1$ (no oversampling) and $p_n(w_i, w_j) = p(w_i)p(w_j)$ negative sampling yields pointwise mutual information.

word2vec vs count-based methods:

+ scale with corpus size; capture complex patterns beyond word similarity
− slower than occurrence count based model; efficient usage of statistics

4 etc: CBOW vs. Skip-Gram

Reference website: In the "skip-gram" mode alternative to "CBOW", rather than averaging the context words, each is used as a pairwise training example. That is, in place of one CBOW example such as [predict 'ate' from average('The', 'cat', 'the', 'mouse')], the network is presented with four skip-gram examples [predict 'ate' from 'The'], [predict 'ate' from 'cat'], [predict 'ate' from 'the'], [predict 'ate' from 'mouse']. (The same random window-reduction occurs, so half the time that would just be two examples, of the nearest words.)

- Skip-gram: works well with small amount of the training data, represents well even rare words or phrases.

- CBOW: several times faster to train than the skip-gram, slightly better accuracy for the frequent words

5 ML4H: Word2Vec

Word2Vec

- Train a classifier on a binary prediction task of words occurring in the neighbourhoods of other words, take the learned classifier weights as the word embeddings.
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary.
- **Continuous Bag-of-Words (CBoW) model**: predict center word from sum of surrounding word vectors (see NLU notes)
- **Skip-gram model**: predicting surrounding single words from center word (see NLU notes). Take the target word and a neighboring context word as positive examples, randomly sample other words in the lexicon to get negative samples.
- Normalized dot-product gives **cosine similarity**. Normalize similarities to get probabilities: $P(+ \mid t, c) = 1/(1 + e^{-sim(t,c)})$
- This means we maximise the overlap (via dot product) between a word and the context it appeared in. By transitivity, any other word with a similar context will have a large overlap with the original word. For example, jumps $\sim$ leaps because their context vectors are similar.

FastText

- Extension of word2vec model, treats each word as composed of character n-grams.
- Can generate embeddings for OOV words, generate better embeddings than word2vec for rare words.
- Requires bigger memory and longer training times.

6 CIL: Skip-gram and negative sampling

- **Skip-gram model:** predict context word, given active word
- Distributional semantics model = distribution of co-occurring words determines lexical semantics.
- An alternative to using windows would be to use words within same sentence.
- log-bilinear model where $w \mapsto (\mathbf{x}_w, b_w) \in \mathbb{R}^{d+1}$:

$$\log p_\theta(w \mid w') = \langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + b_w + const$$

- log-likelihood of the basic model becomes

$$\mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\Delta \in \mathcal{I}} \left[\langle \mathbf{x}_{w^{(t+\Delta)}}, \mathbf{x}_{w^{(t)}} \rangle + b_{w^{(t+\Delta)}} - \log \sum_{v \in \mathcal{V}} \exp \left[\langle \mathbf{x}_v, \mathbf{x}_{w^{(t)}} \rangle + b_v \right] \right]$$

- **Context Vectors:** using the same vectors for context and active words would be imposing and implicit constraint (bi-linearity), so we can use different vectors for inputs and output embeddings. Model dimensionality grows larger but this adds more flexibility to the model.
- **Negative sampling:** simplified version of noise contrastive estimation where the aim is to reduce estimation to binary classification. Negative sampling is done through sampling "random" ($P(w_j)^\alpha$, e.g. $\alpha = 3/4$) context words. Exponent (smaller than one) of the unigram probability dampens frequent words. We move to logistic function from softmax, that needs the computation of partition function with large cardinality.
- log-likelihood of the model with context vectors and negative sampling:

$$\mathcal{L}(\theta) = \sum_{(i,j)\in\Delta^+} \log\sigma\left(\langle \mathbf{x}_i, \mathbf{y}_j\rangle\right) + \sum_{(i,j)\in\Delta^-} \log\sigma\left(-\langle \mathbf{x}_i, \mathbf{y}_j\rangle\right)$$

- In a way, what skip-gram model with negative sampling does is essentially performing a low-rank decomposition of the pointwise mutual information matrix. Negative sampling idea which was a bit heuristic actually seems to be doing something quite reasonable.

7 CIL: GloVe

Using the same notation for output vocabulary $\mathcal{V}$ and input vocabulary $\mathcal{C}$, co-occurence matrix $\mathcal{N} = (n_{ij}) \in \mathbb{N}^{|\mathcal{V}|\cdot|\mathcal{C}|}$ where $n_{ij} = \#$ occurences of $w_i \in \mathcal{V}$ in context of $w_j \in \mathcal{C}$. It is a sparse matrix which can be computed in one pass over the corpus.

GloVe objective is the weighted least squares fit of log-counts:

$$\mathcal{H}(\theta; \mathbf{N}) = \sum_{i,j} f(n_{ij}) \left(\log n_{ij} - \log \widetilde{p}_\theta(w_i \mid w_j)\right)^2$$

with unnormalized distribution

$$\widetilde{p}_\theta(w_i \mid w_j) = \exp\left[\langle \mathbf{x}_i, \mathbf{y}_j\rangle + b_i + c_j\right]$$

and weighting function $f(n) = \min\left\{1, \left(\frac{n}{n_{\max}}\right)^\alpha\right\}$ e.g. $\alpha = \frac{3}{4}$

GloVe with $f \coloneqq 1$ solves a matrix factorization problem of the log-count matrix.

Advantage of the GloVe is that we model unnormalized probabilities which does not need the computation of the partition function.

Optimization through SGD: full gradient is often too expensive to compute for $\mathcal{H}(\theta; \mathbf{N})$, instead use stochastic optimization. We sample (i, j) such that $n_{ij} > 0$ uniformly at random. While the inner product is not big enough (i.e. w.r.t. log-counts) we move the two vectors closer to each other.

$$\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i + 2\eta f(n_{ij})(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y_j}\rangle)\mathbf{y}_j$$
$$\mathbf{y}_j^{\text{new}} \leftarrow \mathbf{y}_j + 2\eta f(n_{ij})(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y_j}\rangle)\mathbf{x}_i$$

8 ML4H: Semantic Relationships

Combining embeddings with prior knowledge: from analogical reasoning, abstract relationships were translations in the embedded space. Take this idea and extend the concept of context to include "appears in a relationship with" alongside "appears in a sentence with" and represent these new context-relationships as arbitrary affine transformations (basically, matrices).

Enforcing similarity: define an energy function $\mathcal{E}(S, R, T)$, energy is low if S is related to T through R is true (R is often non-symmetric). An example energy function is

$$\mathcal{E}(S, R, T \mid \theta) = -\frac{\mathbf{v}_T \cdot G_R \mathbf{c}_S}{||\mathbf{v}_T|| ||G_R \mathbf{c}_S||}$$

9 etc: Practicalities

"Off-task" data helps due to shared semantic information.

Although pre-trained word vectors work very well in many practical downstream tasks, in some settings it's best to continue to learn (i.e. 'retrain') the word vectors as parameters of our neural network. Explain why retraining the word vectors may hurt our model if our dataset for the specific task is too small: See TV/television/telly example. Word vectors in training data move around; word vectors not in training data don't move around. Destroys structure of word vector space. Could also phrase as an overfitting or generalization problem.

Evaluating word vectors:

- Intrinsic: Word Vector Analogies ; Word vector distances and their correlation with human judgments.
- Extrinsic: Name entity recognitions: finding a person, location or organization and so on.