Distributional Semantics

Lecture 5. Lexical-level and Morphological-level Extensions of Word2Vec

Daniil Vodolazsky

March 23th, 2019

Lecture Plan

Lecture Plan

1. Lexical-level Extensions: GloVe, LexVec, Swivel
2. Morphology. Inflectional and Derivational Morphology
3. N-grams. FastText
4. Byte Pair Embeddings
5. Morphological Embeddings
6. Seminar

Introduction

Introduction: What do you think?

Why do we need word2vec extensions?
What drawbacks do word2vec models have?

Lexical-level Extensions

GloVe

The **Global Vector (GloVe)** model (2014) aims to combine the **count-based matrix factorization** and the **context-based skip-gram model** together.
We will define the co-occurrence probability as:

$$P(w_k \mid w_i) = \frac{\#(w_i, w_k)}{\#(w_i)}.$$

Let $w_i =$ "ice" and $w_j =$ "steam". The third word $w_k =$ "solid" is related to "ice" but not "steam", and thus $P(w_k \mid w_i)/P(w_k \mid w_j) \to \infty$.
If the third word $w_k =$ "water" is related to both or $w_k =$ "fashion" is unrelated to either of them, the equation above is expected to be close to one.

The word meanings are captured by the **ratios of co-occurrence probabilities** rather than the **probabilities themselves**. The global vector models the relationship between two words regarding to the third context word as:

$$F(w_i, w_j, w_k) = \frac{P(w_k \mid w_i)}{P(w_k \mid w_j)}.$$

GloVe

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe

Since the goal is to learn meaningful word vectors, F is designed to be a function of the linear difference between two words w_i and w_j:

$$F(w_i, w_j, w_k) = F((\boldsymbol{u}_{w_i} - \boldsymbol{u}_{w_j})^\top \boldsymbol{v}_{w_k}).$$

With the consideration of F being symmetric between target words and context words, the final solution is to model F as an exponential function. And we obtain the following equations:

$$F(\boldsymbol{u}_{w_i}^\top \boldsymbol{v}_{w_k}) = \exp(\boldsymbol{u}_{w_i}^\top \boldsymbol{v}_{w_k}) = P(w_k \mid w_i),$$

$$F((\boldsymbol{u}_{w_i} - \boldsymbol{u}_{w_j})^\top \boldsymbol{v}_{w_k}) = \exp((\boldsymbol{u}_{w_i} - \boldsymbol{u}_{w_j})^\top \boldsymbol{v}_{w_k}) = \frac{\exp(\boldsymbol{u}_{w_i}^\top \boldsymbol{v}_{w_k})}{\exp(\boldsymbol{u}_{w_j}^\top \boldsymbol{v}_{w_k})} = \frac{P(w_k \mid w_i)}{P(w_k \mid w_j)}.$$

Finally,

$$\boldsymbol{u}_{w_i}^\top \boldsymbol{v}_{w_k} = \log P(w_k \mid w_i) = \log \frac{\#(w_i, w_k)}{\#(w_i)} = \log \#(w_i, w_k) - \log \#(w_i).$$

GloVe

$$\boldsymbol{u}_{w_i}^\top \boldsymbol{v}_{w_k} = \log P(w_k \mid w_i) = \log \frac{\#(w_i, w_k)}{\#(w_i)} = \log \#(w_i, w_k) - \log \#(w_i).$$

After adding biases, we obtain

$$\log \#(w_i, w_k) = \boldsymbol{u}_{w_i}^\top \boldsymbol{v}_{w_k} + b_{w_i}^u + b_{w_k}^v.$$

The loss function for the GloVe model is designed to preserve the above formula by minimizing the sum of the squared errors

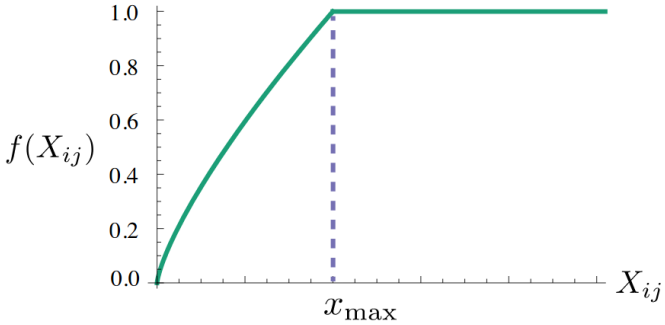
$$L = \sum_{w \in \mathcal{W}} \sum_{c \in \mathcal{W}} f\left(\#(w, c)\right) \left(\boldsymbol{u}_w^\top \boldsymbol{v}_c + b_w^u + b_c^v - \log \#(w, c)\right)^2.$$

GloVe

The paper proposed the following weighting function:

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha}, & \text{if } x < x_{\max}, \\ 1, & \text{otherwise} \end{cases}$$

with optimal values $\alpha = 0.75$ and $x_{\max} = 100$.



LexVec

Just like GloVe, **LexVec** (2016) also tries to factorize PPMI matrices, emerging characteristics of both count-based and prediction-based models. Unlike GloVe, it penalizes errors of frequent co-occurrences more heavily, while still treating negative co-occurrences.
The LexVec loss function has two terms:

$$L^{WC}(w, c) = \frac{1}{2}\left(\boldsymbol{u}_w^\top \boldsymbol{v}_c - PPMI^*_{w,c}\right)^2,$$

$$L^{W}(w) = \frac{1}{2}\sum_{j=1}^{k} \mathbb{E}_{c_{n,j} \sim P_n}\left(\boldsymbol{u}_w^\top \boldsymbol{v}_{c_{n,j}} - PPMI^*_{w,\boldsymbol{c_{n,j}}}\right)^2$$

where $PPMI^*$ is an improved PPMI matrix.

Swivel

The central claim of the authors of **Submatrix-wise vector embedding learner (Swivel)** (2016) is that none of the mainstream word embeddings provide any special treatment to **unobserved** word-context co-occurrences, so the ability to capture them helped to outperform other embedding training algorithms in word similarity and word analogy tasks.
For co-occurrences that have been observed, Swivel computes the weighted squared error between the embedding dot product and the PMI of w and c:

$$L^+(w,c) = \frac{1}{2} f(\#(w,c)) \left(\boldsymbol{u}_w^\top \boldsymbol{v}_c - PMI_{w,c}\right)^2 = \\ = \frac{1}{2} f(\#(w,c)) \left(\boldsymbol{u}_w^\top \boldsymbol{v}_c - \log \#(w,c) - \log T + \log \#(w) + \log \#(c)\right)^2.$$

This encourages $\boldsymbol{u}_w^\top \boldsymbol{v}_c$ to correctly estimate the observed PMI. $f(x) = x^{\frac{1}{2}}$.

Swivel

If $\#(w, c) = 0$, then $PMI_{w,c} = -\infty$, and the squared error cannot be computed.
Treating $\#(w, c)$ as a sample, we can ask: how significant is it that its observed value is zero?
We address this by smoothing the PMI value as if a single co-occurrence had been observed (i.e., computing PMI as if $\#(w, c) = 1$), and using an asymmetric cost function that penalizes over-estimation of the smoothed PMI:

$$\begin{aligned} L^0(w, c) &= \log\left(1 + \exp\left(\boldsymbol{u}_w^\top \boldsymbol{v}_c - PMI^*_{w,c}\right)\right) = \\ &= \log\left(1 + \exp\left(\boldsymbol{u}_w^\top \boldsymbol{v}_c - \log T + \log \#(w) + \log \#(c)\right)\right). \end{aligned}$$

Here, PMI^* refers to the smoothed PMI computation where $\#(w, c)$'s actual count of 0 is replaced with 1. This loss penalizes the model for over-estimating the objective value; however, it applies negligible penalty—i.e., is noncommittal—if the model under-estimates it.

Morphology. Inflectional and Derivational Morphology

Morphology

Morphology is a study of internal structure of words.
Morpheme is the smallest linguistic unit which has a meaning or grammatical function. Words are composed of morphemes (one or more). There are some complications with this simple definition: *sing|er|s*, *moon|light*, *un|kind|ly*, *talk|s*, *ten|th*, *de|nation|al|iz|ation*. The order of morphemes matters: *talk|ed* $\neq$ **ed|talk*, *re|write* $\neq$ **write|re*.
Allomorphs are morphemes having the same function but different form. Unlike the synonyms they usually cannot be replaced one by the other. Here are

1. 1. indefinite article: **an** orange — **a** building
   2. plural morpheme: cat|**s** — dog|**s** — judg|**es**
2. 1. Czech: matk|**a** (mother$_{\text{nom}}$) — mat|ek| (mothers$_{\text{gen}}$) — matc|**e** (mother$_{\text{dat}}$) — matč|**in** (mother's)

Classification of Morphemes: Bound and Free

- **Bound** morpheme cannot appear as a word by itself: -s (dog|s), -ly (quick|ly), -ed (walk|ed).
- **Free** morpheme can appear as a word by itself; often can combine with other morphemes too: dog (dog|s), walk (walked), of, the, or.

Classification of Morphemes: Root and Affixes

- **Root** is a nucleus of the word that affixes attach too. In English, most of the roots are free. In some languages that is less common (Lithuanian: Bill|as Clinton|as). Compounds contain more than one root: home|work.
- **Affix** is a morpheme that is not a root; it is always bound. There are several kinds of affixes. The most common are the following:
  - **suffix**: *talk — talk|**ing***, *quick — quick|**ly***;
  - **prefix**: *happy — **un**|happy*, *existing — **pre**|existing*;
  - **circumfix**: Dutch: *berg (mountain) — **ge**|berg|**te** (mountains)*, ****ge**|berg*, **berg|**te***.
  - **infix**: Tagalog: *basa (read) — b|**um**|asa* ($read_{past}$);
  - **transfix**: Arabic: *k-t-b (write) — k|**a**|t|**a**|b|**a** (he wrote)*, *y**a**|kt|**u**|b|**u** (he is writing)*;
  - **interfix**[1]: *speed|**o**|meter*.

  Suffixes are more common than prefixes which are more common than infixes/circumfixes.

[1] Traditionally, interfix is not considered as morpheme because it has no individual value.

Classification of Morphemes: Content and Functional

- **Content** morphemes carry some semantic content car, -able-, un-, -ness.
- **Functional** morphemes provide grammatical information the, and, -s (plural), -s (3rd sg).

Classification of Morphemes: Inflectional and Derivational

- **Inflection** is a process of creating various forms of the same word: *big — bigger, biggest*. **Lexeme** is the set of all forms related by inflection. **Lemma** is a form from a lexeme chosen by convention to represent that set. E.g., *break, breaks, broke, broken, breaking* have the same lemma *break*. **Ending** is an inflectional suffix.
- **Derivation** is a process of creating new words: *slow — slowly, slowness*.
  Derivation tends to affects the meaning of the word, while inflection tends to affect only its syntactic function.

  However, the boundary between derivation and inflection is often fuzzy and unclear.

  For any word we can build a tree with a "history" of a word: *unbelievable = un + (believe + able)*, but *unbelievable* $\neq$ **(un + believe) + able*. At the same time, some words can be ambiguous: *unlockable = (un + lock) + able, unlockable = un + (lock + able)*.

Morphological Processes

- **Concatenation**. English: *happy — happi|ness*.
- **Reduplication**. Afrikaans: *amper (nearly) — amper|amper (very nearly)*.
- **Templates**. Hebrew: *lomed (learn$_{masc}$) — lomad (learnt$_{masc.sg.3rd}$) — limed (taught$_{masc.sg.3rd}$) — lumad (was taught$_{masc.sg.3rd}$)*.
- **Morpheme internal changes (apophony, ablaut)**. English: *sing — sang — sung, man — men, goose — geese*; German: *Hund (dog) — Hünd|chen (small dog)*.
- **Subtraction (Deletion)**. French: *grande (big$_f$) — grand (big$_m$), fausse (false$_f$) — faux (false$_m$)*.
- **Suppletion**: English: *be — am — is — was, go — went, good — better*.

Word Formation

- **Affixation**. Words are formed by adding affixes: *write — writer, productive — unproductive*.
- **Compounding**. Words are formed by combining two or more words: *rain + bow — rainbow, over + do — overdo*.
- **Acronyms** are like abbreviations, but act as a normal words: *light amplification by simulated emission of radiation — laser*.
- **Blending** parts of two different words are combined: *breakfast + lunch — brunch, motor + hotel — motel*.
- **Clipping** longer words are shortened: *doctor — doc, laboratory — lab, advertisement — ad, examination — exam*.

Morphological Types of Languages

- **Analytic** languages have only free morphemes, sentences are sequences of single morpheme words. A grammatical meaning is usually expressed with separate words.
- **Synthetic** languages have both free and bound morphemes. Unlike analytic languages, a grammatical meaning is usually expressed inside words.
  - **Agglutinating** — each morpheme has a single function, it is easy to separate them. E.g., Uralic lgs (Estonian, Finnish, Hungarian), Turkish, Basque, Dravidian languages (Tamil, Kannada, Telugu), Esperanto.
  - **Fusional** — like agglutinating, but affixes tend to "fuse together", one affix has more than one function. Common homonymy of inflectional affixes. E.g., Slavic, Romance languages, Greek.
  - **Polysynthetic** — extremely complex, many roots and affixes combine together, often one word corresponds to a whole sentence in other languages. E. g. Eskimo: *angyaghllangyugtuq (he wants to acquire a big boat)*.

Usually languages combine several types in different proportions.

N-grams. FastText

N-grams. FastText

FastText (2017) is a model to learn word representations while taking into account morphology. Each word w is represented as a bag of character n-grams. We also include the word w itself in the set of its n-grams, to learn a representation for each word. For a word `where` and $n = 3$, we obtain

`<wh, whe, her, ere, re>, <where>`.

Let $\mathcal{G}_w^n$ be the set of n-grams appearing in w. We associate a vector representation $\boldsymbol{z}_{g^n}$ to each n-gram g^n. We represent a word by the mean of the vector representations of its n-grams.
The embedding of a word w, then, can be expressed as

$$\boldsymbol{u}_w = \frac{1}{|\mathcal{G}_w^n|} \sum_{g^n \in \mathcal{G}_w^n} \boldsymbol{z}_{g^n}.$$

This simple model allows sharing the representations across words, thus allowing to learn reliable representation for rare words.

Byte Pair Embeddings

Byte Pair Embeddings

Byte pair encoding (BPE) (1994) is a variable-length encoding that views text as a sequence of symbols and iteratively merges the most frequent symbol pair into a new symbol. Since the BPE algorithm works with any sequence of symbols, it requires no preprocessing and can be applied to untokenized text.

1. aaabdaaabac
2. ZabdZabac, Z=aa
3. ZYdZYac, Y=ab, Z=aa
4. XdXac, X=ZY, Y=ab, Z=aa

In NLP BPE was first applied in machine translation (2016).

BPEmb (2018) is a collection of pre-trained subword unit embeddings in 275 languages, based on BPE.

Morphological Embeddings

Subword-level Embeddings for Korean (2018)

Every Korean word can be decomposed into a sequence of characters, which in turn can be decomposed into *jamo*s, the smallest lexicographic units representing the consonants and vowels of the language. This combination of *jamo*s completes a character as a syllable.

chosung

joongsung

jongsung

Subword-level Embeddings for Korean (2018)

Similarly to FastText, we construct a vector representation of a Korean word by using the extracted two types of *n*-grams. We compute the sum of *jamo*-level *n*-grams, sum of character-level *n*-grams, and compute mean of the vectors. Let us denote character-level *n*-grams of w to $\mathcal{G}_w^c$, and inter-character *jamo*-level *n*-grams $\mathcal{G}_w^j$, then we obtain the equation for embedding of word w as follows:

$$\boldsymbol{u}_w = \frac{1}{|\mathcal{G}_w^c| + |\mathcal{G}_w^j|} \left(\sum_{g^c \in \mathcal{G}_w^c} \boldsymbol{z}_{g^c} + \sum_{g^j \in \mathcal{G}_w^j} \boldsymbol{z}_{g^j} \right)$$

where $\boldsymbol{z}_{g^j}$ is the vector representation of the *jamo*-level *n*-gram g^j, and $\boldsymbol{z}_{g^c}$ is that of the character-level *n*-gram g^c.

Embeddings Based on Morpheme Segmentation

Morpheme segmentation is a process of deriving words into morphemes.
An embedding for a word w can be computed as

$$\boldsymbol{u}_w = \frac{1}{|\mathcal{G}_w^m|} \sum_{g^m \in \mathcal{G}_w^m} \boldsymbol{z}_{g^m},$$

where $\mathcal{G}_w^m$ represents the set of morphemes of word w.
But the segmentation process itself can be a significantly difficult task, especially for fusional and polysynthetic languages.

Compositionality on a Morphological Level

All models described assume that the base word (input) b and derived word (output) d are represented as vectors in some underlying distributional space. A pattern (semantic shift) p is represented as vector or matrix.

Simple Additive Model: $\boldsymbol{v}_d = \boldsymbol{v}_b + \boldsymbol{v}_p$.

Simple Multiplicative Model (2010): $\boldsymbol{v}_d = \boldsymbol{v}_b \odot \boldsymbol{v}_p$.

Weighted Additive Model: $\boldsymbol{v}_d = \alpha \boldsymbol{v}_b + \beta \boldsymbol{v}_p$.

Dilation Model: $\boldsymbol{v}_d = (\alpha - 1)(\boldsymbol{v}_p^\top \boldsymbol{v}_b)\boldsymbol{v}_p + (\boldsymbol{v}_p^\top \boldsymbol{v}_p)\boldsymbol{v}_b$.

Lexical Function Model (2010): $\boldsymbol{v}_d = \boldsymbol{P}_p \boldsymbol{v}_b$.
Non-cummutativity: $\boldsymbol{P}_{able}(\boldsymbol{P}_{un}(\boldsymbol{v}_{lock})) \neq \boldsymbol{P}_{un}(\boldsymbol{P}_{able}(\boldsymbol{v}_{lock}))$.

Full Additive Model: $\boldsymbol{v}_d = \boldsymbol{A}\boldsymbol{v}_b + \boldsymbol{B}\boldsymbol{v}_p$.

Questions?

Seminar

FastText and GloVe

1. Downloading pre-trained embeddings via gensim
2. Using FastText and GloVe embeddings
3. Homework: FastText for Spelling Correction

TODO

Link