CMPE 297, INSTRUCTOR: JORJETA JETCHEVA

WORD2VEC

DENSE EMBEDDINGS

- Size of dense embeddings 50-1000 dimensions
  - Rather than sparse embeddings that have a dimensionality which is equal to size of the vocabulary (10s of thousands), or the size of the document corpus (millions)
- Work better than sparse embeddings on every NLP task
- Some intuition behind why this may be the case:
  - Models require fewer weights (as there are fewer features) and thus are less likely to overfit
  - Distance metrics work better in lower dimensions - similar words are more clearly related in lower dimensions

WORD2VEC

- Learns static embeddings
  - Each word has the same vector embedding regardless of the context in which it appears
- Uses self-supervision (no supervised signal)
  - Uses the next word in running text as a supervisory signal
- Overall process
  1. Treat the target word and a neighboring context word as positive examples
  2. Randomly sample other words in the lexicon to get negative samples (target word, random other word)
  3. Use logistic regression to train a classifier to distinguish those two cases
  4. Use the learned weights as word embeddings in other NLP tasks

SKIP-GRAM WITH NEGATIVE SAMPLING (SGNS)

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

High-level process:

- Step 1: Assign a random embedding vector to each word in the vocabulary (e.g., vocabulary has size N)
- Step 2: Start shifting the embedding of each word to be more like the embeddings of co-occurring words (context words), and away from those of non-co-occurring words

Train a logistic classifier to differentiate between

- Positive examples: pairs of target word with 1 context word (found to be co-occurring with the word in question)
- Negative examples: pairs of target word with 1 noise word (randomly selected from the vocabulary)
  - Number of negative words is k × number of positive words (k is a hyper-parameter of the model)
  - Noise words are chosen based on their weighted unigram frequency:

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

α is typically set to 0.75 which gives rare noise words a slightly higher probability

Example: P(a) = 0.99, and P(b) = 0.01

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

LEARNING TASK FORMULATION

- Maximize the similarity of the target word w and context word c_{pos} pairs (w, c_{pos}) (positive examples)
  - $P(+|w, c_{pos})$ - probability that the pair (w, c_{pos}) belongs to the positive class
- Mimimize the similarity of the target word w and noise word c_{neg} pairs (w, c_{neg}) (negative examples)
  - $P(-|w, c_{neg})$ should be minimized - probability that the pair (w, c_{neg}) belongs to the negative class
  - Because we have **k** negative examples, we need to minimize: $\prod_{i=1}^{k} P(-|w, c_{neg_i})$
    - We assume independence due to randomly drawing these from the lexicon
- To achieve both of the above goals, we can combine them into a loss function **L** that we can minimize (minus sign is added to indicated that we want to minimize **L**), and we also take a log to simplify the computation:

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w, c_{pos}) \prod_{i=1}^{k} P(-|w, c_{neg_i})\right] \\
&= -\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log P(-|w, c_{neg_i})\right] \\
&= -\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log\left(1 - P(+|w, c_{neg_i})\right)\right]
\end{aligned}
$$

We multiply the two probabilities because we want them to be true at the same time, and we assume they are independent.

LEARNING TASK FORMULATION (CONT.)

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right]
\end{aligned}
$$

- The probability of a word and context word being "similar" is computed based on the dot product of their vectors (similarity is proportional to vector dot product - recall Cosine Similarity discussion)
- In order to get a probability-valued output, we use the sigmoid on top of the dot product and make sure the + and - probabilities add up to 1

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w})$$

$$P(-|w,c) = 1 - P(+|w,c) = \sigma(-\mathbf{c}\cdot\mathbf{w})$$

- So we can substitute those in the loss function formula as shown below:

$$L_{CE} = -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] = -\left[\log\sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]$$

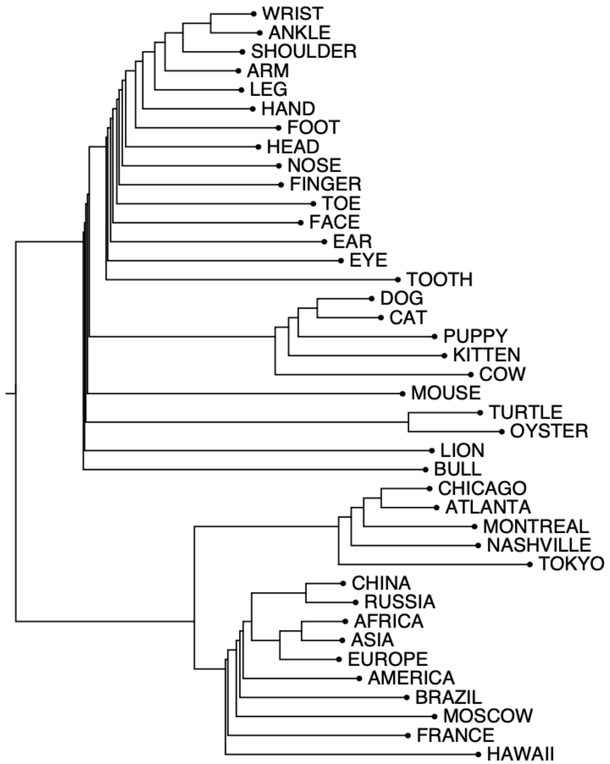
- Finally, we use gradient descent to find the w parameter vector of the logistic regression for which L_{CE} is minimized
- We can represent the word with w_i (target embedding)
- Context window size is varied to find a good value (increasing the range improves quality of the resulting word vectors, but it also increases the computational complexity)

OTHER COMMONLY USED STATIC EMBEDDINGS

- fasttext
  - Deals well with unknown words, and words that occur rarely
  - Stores both the word and some of its character **n-grams**, e.g., for the word "**where**", we could use: **wh**, **the**, **her**, **ere**, **re**
  - A separate embedding is learned for each **n-gram** and the word itself is represented as the sum of these n-gram embeddings
  - The character n-grams from the dataset can be used to represent unknown words
- GloVe
  - Incorporates some global corpus statistics from the word-word co-occurrence matrix

VISUALIZING EMBEDDINGS

- Show a sorted list of all words with most similar vectors (cosine similarity) to a given word w
- Or, use hierarchical clustering of the embeddings to illustrate word relationships:



VISUALIZING EMBEDDINGS (CONT.)

Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

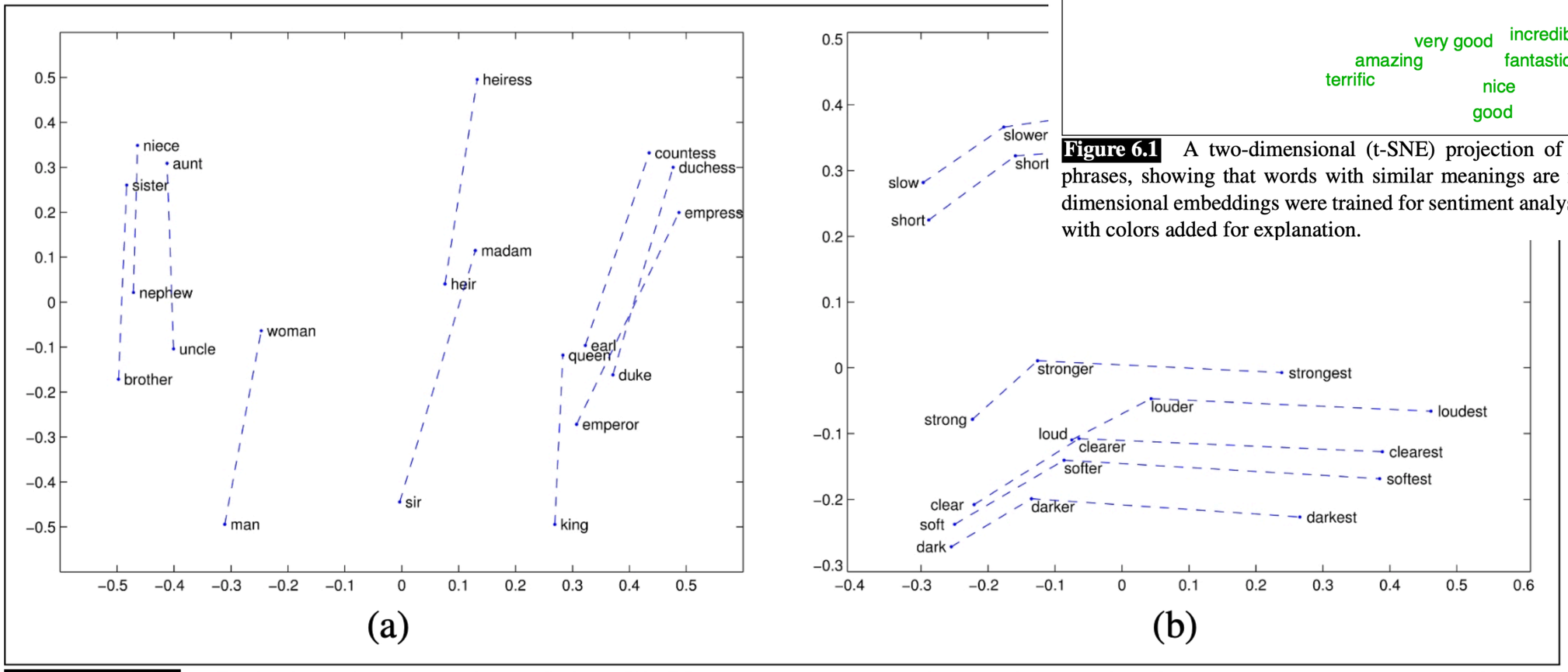


Figure 6.16 Relational properties of the GloVe vector space, shown by projecting vectors onto two dimensions. (a) $\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman}$ is close to $\overrightarrow{queen}$. (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

SEMANTICS AND ASSOCIATION PROPERTIES OF VECTOR EMBEDDINGS

Syntactic vs. Topical similarity

- Shorter context windows used by the embedding algorithms result in more similar vector embeddings in a syntactic sense where the words tend to be the same part of speech for example - specific
- Longer context windows used by the embedding algorithms result in similar embeddings for words that are topically related, but not necessarily similar – broad

First-order vs. Second-order association

- First-order co-occurrence (syntagmatic association) applies to words that are typically near each other, e.g., "book" and "poem" tend to be nearby "wrote"
- Second-order co-occurrence (paradigmatic association) is when words have similar neighbors, e.g., "wrote: is a second-order associate of "said" or "remarked"

ANALOGY-BASED/RELATIONAL SIMILARITY

- Vector embeddings (especially dense ones) are very good at analogy problems
- Analogy problems: a is to b as as $a*$ is to **?**
  - Example: Man is to Woman as King is to ?

A	B	C	D = C + (B − A)	Relationship
Athens	Greece	Oslo	Norway	*Capital*
Astana	Kazakhstan	Harare	Zimbabwe	*Capital*
Angola	kwanza	Iran	rial	*Currency*
copper	Cu	gold	Au	*Atomic Symbol*
Microsoft	Windows	Google	Android	*Operating System*
New York	New York Times	Baltimore	Baltimore Sun	*Newspaper*
Berlusconi	Silvio	Obama	Barack	*First name*
Switzerland	Swiss	Cambodia	Cambodian	*Nationality*
Einstein	scientist	Picasso	painter	*Occupation*
brother	sister	grandson	granddaughter	*Family Relation*
Chicago	Illinois	Stockton	California	*State*
possibly	impossibly	ethical	unethical	*Negative*
mouse	mice	dollar	dollars	*Plural*
easy	easiest	lucky	luckiest	*Superlative*
walking	walked	swimming	swam	*Past tense*

Figure 24.2 A word embedding model can sometimes answer the question "**A** is to **B** as **C** is to [what]?" with vector arithmetic: given the word embedding vectors for the words **A**, **B**, and **C**, compute the vector **D** = **C** + (**B** − **A**) and look up the word that is closest to **D**. (The answers in column **D** were computed automatically by the model. The descriptions in the "Relationship" column were added by hand.) Adapted from ? (?, ?).

Russel & Norvig: AI, A Modern Approach

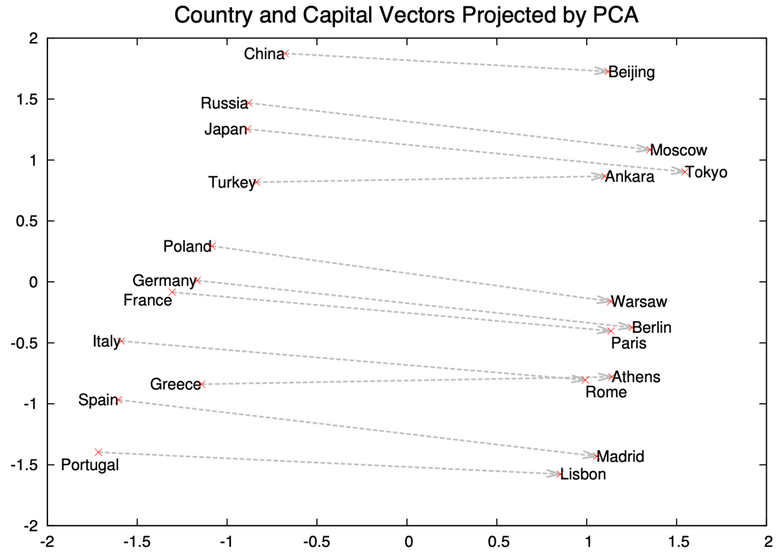


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.

EVALUATING VECTOR MODELS

- Most important evaluation is end-to-end/extrinsic (how well does the NLP task perform, e.g., sentiment classification when using a particular embedding representation)

Intrinsic evaluation methods

- Using human-labeled word similarity data for comparison
  - Basic similarity: similarity of words without taking context into account
    - Labeled word pairs, multiple choice to identify synonyms
  - Semantic similarity of sentences
  - Analogy tasks

WORD2VEC PITFALLS

OVERVIEW

- We don’t fully understand why algorithms like word2vec work
- There is no guarantee that a particular algorithm on a particular corpus will capture a particular semantic relationship
- Algorithms even trained on the same dataset may produce different embeddings (e.g. due to randomness during initialization, differences in the random choices during negative sampling)
- Word embeddings can be biased (capturing gender and racial biases from the data on which they are trained)

BIAS IN EMBEDDINGS

- A number of scientists have shown that word2vec and other embeddings display gender bias
  - This can lead to allocation harm, e.g., recruiting applications can rate male applicants higher and result in them getting allocated more jobs
- When trained on news, word2vec embeddings used in analogies produce the following results:
  - In the analogy ‘man’-‘computer programmer’ + ‘women’, the algorithm outputs ‘home maker’.
  - In the analogy ‘father’-doctor’+’mother’, the algorithm outputs ‘nurse’
- Embeddings have been found to amplify bias beyond the statistical differences in actual labor employment statistics
- Representational harm — demeaning or causing groups of people to be ignored
  - African American names have been found to have a higher GloVe coding similarity with unpleasant words, while European-American names with pleasant words - impacting applications that use sentiment analysis
- Debiasing that tries to eliminate bias by transforming the embedding space is an active area of research