Word embedding

training, evaluation, and known issues

Hyper-parameters

Training hyper-parameters

- **embedding dimension** (*d*): number of elements in each word's embedding vector
- **window size** (*m*): number of context words to the left, and the right (# context words = 2x window size)
- **number of epochs**: Number of times the model iterates over each position *t* in your corpus

for word2vec (skip-gram):

- number of negative examples
- some other hyperparameters (don't worry about them)

Training hyper-parameters – *how to choose*

- **embedding dimension** (*d*): number of elements in each word's embedding vector
- **window size** (*m*): number of context words to the left, and the right (# context words = 2x window size)
- **number of epochs**: Number of times the model iterates over each position *t* in your corpus

embedding dimension (*d*)

d = 300 is the standard value because performance gains from increasing *d* tend to flatten out at 300

But if you have a relatively small corpus (n ≤ 5m or so), decreasing d to 200 or 100 can improve embedding quality

Why? *d* determines the capacity of your model. Using less capacity to compress "knowledge" can be better with fewer data.

Training hyper-parameters – *how to choose*

- **embedding dimension** (*d*): number of elements in each word's embedding vector
- **window size** (*m*): number of context words to the left, and the right (# context words = 2x window size)
- **number of epochs**: Number of times the model iterates over each position *t* in your corpus

window size (*m*)

m should be at least 2

- the small, the more focus on words' functional similarities
- the bigger, the more focus on a documents' topics

Why? with larger *m, word* embedding models behave like LDA topic models (learn from words tend to co-occur in documents)

Training hyper-parameters – *how to choose*

- **embedding dimension** (*d*): number of elements in each word's embedding vector
- **window size** (*m*): number of context words to the left, and the right (# context words = 2x window size)
- **number of epochs**: Number of times the model iterates over each position *t* in your corpus

number of epochs

- choose a large value, e.g. 30 or 50
- if you train for "too many" epochs, the model will just converge on paramters
  - this *should* show in plateauing loss values
  - but gensim's loss reporting is off
- If you want to implement "early stopping" (stop iterating when the loss plateaus), you should read this blog post (not possible with gensim)

How much does size matter?

Corpus size

- all else equal, embedding quality tends to improve with corpus size
- a rule of thumb is that your corpus should have *at least* 5-7 million tokens

BUT interaction with domain

in many cases, there is a trade-off between training on a large corpus, and training on domain-specific data (here)

Rodriguez & Spirling (2021) say going for an off-domain, of-the-shelf model is good enough

But evidence here suggests that you should prioritize domain-specific data over size big corpus

Evaluating embedding models

Evaluation

- ***intrinsic***: compute performance on linguistic tasks (e.g., detecting synonyms, antonyms, etc.; solving analogy problems; etc.)
- ***extrinsic***: use embeddings as features for some downstream task (e.g., classification)

see here, here, and here

Benchmark

- *curated* datasets with data for intrinsic and/or extrinsic evaluation tasks
- used for performance reporting to gauge "progress" in NLP
- many established benchmarks
  - e.g., BATS
  - see here

Known methodological issues

Issues with word embeddings

Instability

randomness in initial values and in sampling negative examples (with skip-gram) means that final embeddings will vary (despite training on same data)

In such situations, you'd want to set the seed to control randomness. *But* in this case gensim can run only on one core (hence, will be super slow)

What to do about it (best practices)

train multiple models on the same data, and

- average their embeddings *or*
- compute your measures with each and average/summarize their scores (=> info about uncertainty)

Issues with word embeddings

Credibility

we have seen how many “researcher degrees of freedom” there are in constructing metrics/measures from word embeddings

You should read applied papers critically

What to do about it (best practices)

- evaluate the quality of your word embeddings (on benchmark task)
- validate your measures
  - against human judgments
  - external indicators
- try to get sense of measurement uncertainty
  - permutation test à la Caliskan
  - resample keywords, re-compute metrics, and check correlation

Issues with word embeddings

Out-of-domain application of benchmarks

- established benchmarks used in CS and NLP literature might not reflect word meaning and usage patterns in domains such as politics or law
- unclear how well they quantify "quality" in such applications

Issues with word embeddings

Multiple word senses

- each word has just one embedding
- embeddings of words with *multiple senses* will be a average of word senses' (latent) embeddings, weighted by their relative frequency in the training corpus
- so most prevalent senses' “meaning” will dominate the final word embedding

What to do about it (best practices)

- nothing to do about it unless you have a preprocessing technique to indicate words different senses (can't think of any good)
- just go for transformers