A Lists of electronic dictionaries

Electronic monolingual dictionaries are rather widespread compared to machine-readable lexical databases. Wiktionary[7] is a collaborative online dictionary, where 79 languages have more than 10,000 entries and 42 languages have more than 100,000 entries. BabelNet (Navigli and Ponzetto, 2012) aligns Wikipedia and WordNet to automatically obtain definitions ("glosses" in BabelNet terminology).

Alternatively, Wikipedia provides a list of monolingual dictionaries.[8] One can also translate "dictionary" into the desired language and look up this term. For example, there are dictionaries for Friulian and Frisian, which are minority languages spoken by less than a million speakers.[9]

B Data

The dump of Wikipedia that we have mentioned in Section 6 is the dump from the 14th June 2014. Although this dump is not available anymore online, results should be replicable with newer dumps.

B.1 Split dictionary setting

We briefly describe the algorithm used to split the dictionary. If we had used a regular dictionary instead of WordNet, we could have randomly distributed words into a train, validation and test split. However, by doing so, we would have put synonyms in two different splits, which could be a weak form of a test-set leak. On the other hand, WordNet has sets of synonyms, called synsets, where two words in a synset necessarily share at least one definition.

We create batches of words which definitions do not overlap across batches with the following algorithm: First, we create maps from words to their definitions and their converse. We will create batches indexed by i, and we now describe how to build the batch i. We create a set of definitions that is initialized as a singleton containing a random definition, $C_i = \{d\}$. We instantiate another set $S_i = \emptyset$ that will contain all the words which definitions overlap with at least one other word of the set. We iterate over the set C_i and pop a definition. Then, we go through all the words that possess that definition and add them to S_i, while we also add all the other definitions of these words to C_i. We keep iterating over C_i until it is empty. Then we start again the process on a new batch with a new $C_{i+1} = \{d'\}$ where d' has never been added to a C_i before.

Then, we order the batches S_i by the number of words they contain. We choose the largest batch to be the training set. It can be seen as a large subset of the vocabulary that is "central", in a way. By construction, it contains highly polysemous words and frequent words, as shown in Table 5.

Although our construction method is very different, it is slightly similar in spirit to the grounding kernel presented by (Picard et al., 2009). They report that words in the grounding kernel are more frequent, as we do for the training set. It is also related to the concept of semantic primes, a subset of the lexicon from which all other words can be defined (Wierzbicka, 1996).

C Hyperparameter search

All embeddings are of size 300 for the small Wikipedia dump experiments and 400 for the large Wikipedia dump experiments.

C.1 GloVe

GloVe is run only on the definition corpus because we found word2vec to be superior. We have fixed the number of epochs to be 50. The window size varies between $\{5, 7, 9, 11, 15, \mathbf{20}\}$ and x_{max} in $\{0, 5, \mathbf{20}, 100\}$, where the selected hyperparameters are bolded.

C.2 Word2vec

We have used gensim implementation.[10]

On definitions, we do a hyperparameter search on the window size in $\{5, \mathbf{15}\}$ and on the downsampling threshold in $\{0, \mathbf{0.1}, 0.01, 0.001, 0.0001\}$, and we also choose from the skip-gram or CBOW variant, where the skip-gram variant is selected.

On the small training corpus, we only use the skip-gram model and choose the number of iterations in $\{5, \mathbf{30}\}$, the window size in $\{3, \mathbf{5}, 7, 10\}$, and the downsampling threshold in $\{0, 0.1, 0.01, \mathbf{0.001}, 0.0001\}$, where the selected hyperparameters are boldened.

[7] https://www.wiktionary.org/

[8] https://en.wikipedia.org/wiki/List_of_dictionaries_by_number_of_words

[9] https://taalweb.frl/wurdboekportaal http://www.arlef.it/grant-dizionari-talian-furlan/htdocs/gdbtf.pl

[10] https://radimrehurek.com/gensim/

	Development		Similarity			Relatedness				
	SV-d	MENd	SL999	SL333	SV-t	RG	SCWS	MENt	MT	353
word2vec	28.2	73.5	36.5	14.2	23.0	76.4	64.2	74.6	64.0	**68.5**
retrofitting	32.8	**74.4**	40.2	20.7	28.2	**84.0**	**65.3**	**77.7**	**65.6**	62.9
dict2vec	36.5	67.9	41.0	24.2	32.0	74.1	61.7	68.1	57.9	64.8
Hill	29.4	62.4	31.9	17.2	20.0	67.1	53.6	62.9	52.6	50.7
AE	33.0	45.6	34.7	24.1	30.3	71.5	49.3	45.5	38.4	42.5
CPAE-P ($\lambda = 16$)	**39.3**	63.1	**45.2**	**31.5**	**37.4**	69.6	59.3	60.7	50.2	58.5

Table 4: **Improving pretrained embeddings computed on a large corpus**. Spearman's correlation coefficient $\rho \times 100$ on benchmarks. Same as Table 2 but word2vec is trained on the entire Wikipedia dump. Dict2vec fails to improve. Retrofitting especially improves relatedness, while CPAE improves similarity.

	# defs	# defs / # words	Avg. counts
Train	36,722	2.87	11.40
Valid	2,109	1.81	4.07
Test	128,223	1.13	1.92

Table 5: **Statistics of the split dictionary**. By construction, the train set contains much more frequent and polysemous words than the train set. The average counts are geometric averages of the smoothed counts of words, computed on the first 50M tokens of the Wikipedia dump.

On the full training corpus, we have used the same hyperparameters except that the number of iteration is reduced to 5, and we have filtered out words that appear less than 50 times.

C.3 AE, CPAE, Hill's model

We train the AE and CPAE models with Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of $3 \cdot 10^{-4}$, a batch size of 32. In the settings where we use part of the dictionary, we use early stopping: every $2,000$ batches, we compute the mean cost on the validation set and stop after 20000 batches without improvement. On the full dictionary, where we do not have a validation set, we train for 50 epochs.

CPAE and AE models always use a reconstruction cost so $\alpha = 1$. The λ parameter that weights the cost of the consistency penalty varies in $\{1, 2, 4, 8, 16, 32, 64\}$, and the value chosen by the model selection is indicated in the tables.

C.4 Retrofitting

We have used the original implementation.[11] We have tuned the hyperparameter α that controls the proximity of the retrofitted vector to the original vector by grid-search: $\alpha \in \{0.25, 0.5, 1, 2, 4\}$. The selected value is $\alpha = 0.5$ for both the small and the large corpora.

C.5 Dict2vec

We have used the original implementation.[12] Dict2vec has many hyperparameters. We fixed $K = 5$, where the K closest neighbours to each word in the original embeddings are promoted to form a strong pair, as well as $n_s = 4$ and $n_w = 5$, the number of pairs to sample. We run a grid-search over the hyperparameters, the coefficients that weight the strong and weak pairs importance in the auxiliary cost: $\beta_s \in \{0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$ and $\beta_w \in \{0.0, 0.2, 0.4, 0.6\}$. In the smaller data regime (small dump), $\beta_s = 1.2$ and $\beta_w = 0$, and in the larger data regime (the entire Wikipedia dump), the model selection procedure picks $\beta_s = 0.8$ and $\beta_w = 0.2$. When focusing on the similarity relation, it seems better not to use weak pairs, or at least to weight them very low.

D Improving pretrained embeddings on the full Wikipedia dump

The scores of word2vec are not really improved by using the much larger full Wikipedia dump, so we increase the size of the embeddings from 300 to 400. The results are given in Table 4.

The trends are similar to what we have observed on the first 50 million tokens of Wikipedia. However, dict2vec seems a bit better and improves over retrofitting in similarity.

[11] https://github.com/mfaruqui/retrofitting

[12] https://github.com/tca19/dict2vec