1 Paraphrase Lexicon

While PARANMT-50M consists of sentence pairs, we demonstrate how a paraphrase lexicon can be extracted from it. One simple approach is to extract and rank word pairs \( \langle u,v \rangle \) using the cross-sentence pointwise mutual information (PMI):

\[
\text{PMI}_{\text{cross}}(u,v) = \log \frac{\#(u,v)\#(\cdot,\cdot)}{\#(u)\#(v)}
\]

where joint counts \( \#(u,v) \) are incremented when \( u \) appears in a sentence and \( v \) appears in its paraphrase. The marginal counts (e.g., \( \#(u) \)) are computed based on single-sentence counts, as in ordinary PMI. This works reasonably well but is not able to differentiate words that frequently occur in paraphrase pairs from words that simply occur frequently together in the same sentence. For example, “Hong” and “Kong” have high cross-sentence PMI. We can improve the score by subtracting the ordinary PMI that computes joint counts based on single-sentence co-occurrences. We call the result the adjusted PMI:

\[
\text{PMI}_{\text{adj}}(u,v) = \text{PMI}_{\text{cross}}(u,v) - \text{PMI}(u,v)
\]

Before computing these PMIs from PARANMT-50M, we removed sentence pairs with a paraphrase score less than 0.35 and where either sentence is longer than 30 tokens. When computing the ordinary PMI with single-sentence context, we actually compute separate versions of this PMI score for translations and references in each PARANMT-50M pair, then we average them together. We did this because the two sentences in each pair have highly correlated information, so computing PMI on each half of the data would correspond to capturing natural corpus statistics in a standard application of PMI.

Table 2 shows an evaluation of the resulting score functions on the SimLex-999 word similarity dataset (Hill et al., 2015). As a baseline, we use the lexical portion of PPDB 2.0 (Pavlick et al., 2015), evaluating its ranking score as a similarity score and assigning a similarity of 0 to unseen word pairs. Our adjusted PMI computed from PARANMT-50M is on par with the best PPDB lexicon.

Table 1 shows examples from PPDB and our paraphrase lexicon computed from PARANMT-50M. Paraphrases from PPDB are ordered by the PPDB 2.0 scoring function. Paraphrases from our lexicon are ordered using our adjusted PMI scoring function; we only show paraphrases that appeared at least 10 times in PARANMT-50M.

2 General-Purpose Sentence Embedding Evaluations

We evaluate our sentence embeddings on a range of tasks that have previously been used for evaluating sentence representations (Kiros et al., 2015). These include sentiment analysis (MR, Pang and Lee, 2005; CR, Hu and Liu, 2004; SST, Socher et al., 2013), subjectivity classification (SUBJ; Pang and Lee, 2004), opinion polarity (MPQA; Wiebe et al., 2005), question classification (TREC; Li and Roth, 2002), paraphrase detection (MRPC; Dolan et al., 2004), semantic relatedness (SICK-R; Marelli et al., 2014), and textual entailment (SICK-E). We use the SentEval package from Conneau et al. (2017) to train models on our fixed sentence embeddings for each task.

Table 3 shows results on the general sentence embedding tasks. Each of our individual models produces 300-dimensional sentence embeddings, which is far fewer than the several thousands (often 2400-4800) of dimensions used in most prior

---

1 If both orderings for a SimLex word pair appear in PPDB, we average their PPDB 2.0 scores. If multiple lexical entries are found with different POS tags, we take the first instance.

2 github.com/facebookresearch/SentEval
Table 1: Example lexical paraphrases from PPDB ranked using the PPDB 2.0 scoring function and from the paraphrase lexicon we induced from PARA-NMT-50M ranked using adjusted PMI.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Score</th>
<th>( \rho \times 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPDB L</td>
<td>PPDB 2.0</td>
<td>37.97</td>
</tr>
<tr>
<td>PPDB XL</td>
<td>PPDB 2.0</td>
<td>52.32</td>
</tr>
<tr>
<td>PPDB XXL</td>
<td>PPDB 2.0</td>
<td>60.44</td>
</tr>
<tr>
<td>PPDB XXXL</td>
<td>PPDB 2.0</td>
<td>61.47</td>
</tr>
<tr>
<td>PARA-NMT-50M</td>
<td>cross-sentence PMI</td>
<td>52.12</td>
</tr>
<tr>
<td>PARA-NMT-50M</td>
<td>adjusted PMI</td>
<td><strong>61.59</strong></td>
</tr>
</tbody>
</table>

Table 2: Spearman’s \( \rho \times 100 \) on SimLex-999 for scored paraphrase lexicons.

work. While using higher dimensionality does not improve correlation on the STS tasks, it does help on the general sentence embedding tasks. Using higher dimensionality leads to more trainable parameters in the subsequent classifiers, increasing their ability to linearly separate the data.

To enlarge the dimensionality, we concatenate the forward and backward states prior to averaging. This is similar to Conneau et al. (2017), though they used max pooling. We experimented with both averaging (“BLSTM (Avg., concatenation)”) and max pooling (“BLSTM (Max, concatenation)”) using recurrent networks with 2048-dimensional hidden states, so concatenating them yields a 4096-dimension embedding. These high-dimensional models outperform SkipThought (Kiros et al., 2015) on all tasks except SUBJ and TREC. Nonetheless, the InferSent (Conneau et al., 2017) embeddings trained on AllNLI still outperform our embeddings on nearly all of these general-purpose tasks.

We also note that on five tasks (SUBJ, MPQA, SST, TREC, and MRPC), all sentence embedding methods are outperformed by supervised baselines. These baselines use the same amount of supervision as the general sentence embedding methods; the latter actually use far more data overall than the supervised baselines. This suggests that the pretrained sentence representations are not capturing the features learned by the models engineered for those tasks.

We take a closer look of how our embeddings compare to InferSent (Conneau et al., 2017). InferSent is a supervised model trained on a large textual entailment dataset (the SNLI and MultiNLI corpora (Bowman et al., 2015; Williams et al., 2017), which consist of nearly 1 million human-labeled examples).

While InferSent has strong performance across all downstream tasks, our model obtains better results on semantic similarity tasks. It consistently reach correlations approximately 10 points higher than those of InferSent.

Regarding the general-purpose tasks, we note that some result trends appear to be influenced by the domain of the data. InferSent is trained on a dataset of mostly captions, especially the model trained on just SNLI. Therefore, the datasets for the SICK relatedness and entailment evaluations are similar in domain to the training data of InferSent. Further, the training task of natural language inference is aligned to the SICK entailment task. Our results on MRPC and entailment are significantly better than SkipThought, and on a paraphrase task that does not consist of caption data (MRPC), our embeddings are competitive with InferSent. To quantify these domain effects, we performed additional experiments that are described in Section 2.1.

There are many ways to train sentence embeddings, each with its own strengths. InferSent, our models, and the BYTE mLSTM of Radford et al. (2017) each excel in particular classes of downstream tasks. Ours are specialized for semantic similarity. BYTE mLSTM is trained on review data and therefore is best at the MR and CR tasks. Since the InferSent models are trained using entailment supervision and on caption data, they excel on the SICK tasks. Future work will be needed to combine multiple supervision signals to generate embeddings that perform well across all tasks.

2.1 Effect of Training Domain on InferSent

We performed additional experiments to investigate the impact of training domain on down-
stream tasks. We first compare the performance of our “WORD, TRIGRAM (concatenation)” model to the InferSent SNLI and AllNLI models on all STS tasks from 2012-2016. We then compare the overall mean with that of the three caption STS datasets within the collection. The results are shown in Table 4. The InferSent models are much closer to our model on the caption datasets than overall, and InferSent trained on SNLI shows the largest difference between its overall performance and its performance on caption data.

We also compare the performance of these models on the STS Benchmark under several conditions (Table 5). Unsupervised results were obtained by simply using cosine similarity of the pre-trained embeddings on the test set with no training or tuning. Supervised results were obtained by training and tuning using the training and development data of the STS Benchmark.

We first compare unsupervised results on the entire test set, the subset consisting of captions (3,250 of the 8,628 examples in the test set), and...
Table 5: STS Benchmark results (Pearson’s $r \times 100$) comparing our WORD, TRIGRAM model to InferSent trained on AllNLI and SNLI. We report results using all of the data (All), only the caption portion of the data (Cap.), and all of the data except for the captions (No Cap.).

<table>
<thead>
<tr>
<th>Model</th>
<th>All</th>
<th>Cap.</th>
<th>No Cap.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsupervised</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InferSent (AllNLI)</td>
<td>70.6</td>
<td>83.0</td>
<td>56.6</td>
</tr>
<tr>
<td>InferSent (SNLI)</td>
<td>67.3</td>
<td>83.4</td>
<td>51.7</td>
</tr>
<tr>
<td>WORD, TRIGRAM</td>
<td>79.9</td>
<td>87.1</td>
<td>71.7</td>
</tr>
<tr>
<td><strong>Supervised</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InferSent (AllNLI)</td>
<td>75.9</td>
<td>85.4</td>
<td>64.8</td>
</tr>
<tr>
<td>InferSent (SNLI)</td>
<td>75.9</td>
<td>86.4</td>
<td>63.1</td>
</tr>
</tbody>
</table>

The remainder. We include analogous results in the supervised setting, where we filter the respective training and development sets in addition to the test sets. Compared to our model, InferSent shows a much larger gap between captions and non-captions, providing evidence of a bias. Note that this bias is smaller for the model trained on AllNLI, as its training data includes other domains.

References


Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models


