guiding principle behind these approaches is that sense-level representations can be imputed (or improved) from other representations that are known to correspond to generalizations due to the network's taxonomical structure. Vial et al. (2018) leverages relations in WordNet to reduce the sense inventory to a minimal set of entries, making the task easier to model while maintaining the ability to distinguish senses. We take the inverse path of leveraging relations to produce representations for additional senses.

On §3.1 we covered synsets, hypernyms and lexnames, which correspond to increasingly abstract generalizations. Missing sense embeddings are imputed from the aggregation of sense embeddings at each of these abstraction levels. In order to get embeddings that are representative of higher-level abstractions, we simply average the embeddings of all lower-level constituents. Thus, a synset embedding corresponds to the average of all of its sense embeddings, a hypernym embedding corresponds to the average of all of its synset embeddings, and a lexname embedding corresponds to the average of a larger set of synset embeddings. All lower abstraction representations are created before next-level abstractions to ensure that higher abstractions make use of lower generalizations. More formally, given all missing senses in WordNet $\hat{s} \in W$, their synset-specific sense embeddings $S_{\hat{s}}$, hypernym-specific synset embeddings $H_{\hat{s}}$, and lexname-specific synset embeddings $L_{\hat{s}}$, the procedure has the following stages:

(1) $if|S_{\hat{s}}| > 0, \quad \vec{v}_{\hat{s}} = \frac{1}{|S_{\hat{s}}|} \sum \vec{v}_s, \forall \vec{v}_s \in S_{\hat{s}}$

(2) $if|H_{\hat{s}}| > 0, \quad \vec{v}_{\hat{s}} = \frac{1}{|H_{\hat{s}}|} \sum \vec{v}_{syn}, \forall \vec{v}_{syn} \in H_{\hat{s}}$

(3) $if|L_{\hat{s}}| > 0, \quad \vec{v}_{\hat{s}} = \frac{1}{|L_{\hat{s}}|} \sum \vec{v}_{syn}, \forall \vec{v}_{syn} \in L_{\hat{s}}$

In Table 1 we show how much coverage extends while improving both recall and precision.

		F1 / P / R (without MFS)					
Source	Coverage	BERT			ELMo		
SemCor	16.11%	68.9	72.4	65.7	63.0	66.2	60.1
+ synset	26.97%	70.0	72.6	70.0	63.9	66.3	61.7
+ hypernym	74.70%	73.0	73.6	72.4	67.2	67.7	66.6
+ lexname	100%	73.8	73.8	73.8	68.1	68.1	68.1

Table 1: Coverage of WordNet when extending to increasingly abstract representations along with performance on the ALL test set of Raganato et al. (2017a).

4.3 Improving Senses using the Dictionary

There's a long tradition of using glosses for WSD, perhaps starting with the popular work of Lesk (1986), which has since been adapted to use distributional representations (Basile et al., 2014). As a sequence of words, the information contained in glosses can be easily represented in semantic spaces through approaches used for generating sentence embeddings. There are many methods for generating sentence embeddings, but it's been shown that a simple weighted average of word embeddings performs well (Arora et al., 2017).

Our contextual embeddings are produced from NLMs using attention mechanisms, assigning more importance to some tokens over others, so they already come 'pre-weighted' and we embed glosses simply as the average of all of their contextual embeddings (without preprocessing). We've also found that introducing synset lemmas alongside the words in the gloss helps induce better contextualized embeddings (specially when glosses are short). Finally, we make our dictionary embeddings ($\vec{v}_d$) sense-specific, rather than synset-specific, by repeating the lemma that's specific to the sense, alongside the synset's lemmas and gloss words. The result is a sense-level embedding, determined without annotations, that is represented in the same space as the sense embeddings we described in the previous section, and can be trivially combined through concatenation or average for improved performance (see Table 2).

Our empirical results show improved performance by concatenation, which we attribute to preserving complementary information from glosses. Both averaging and concatenating representations (previously L_2 normalized) also serves to smooth possible biases that may have been learned from the SemCor annotations. Note that while concatenation effectively doubles the size of our embeddings, this doesn't equal doubling the expressiveness of the distributional space, since they're two representations from the same NLM. This property also allows us to make predictions for contextual embeddings (from the same NLM) by simply repeating those embeddings twice, aligning contextual features against sense and dictionary features when computing cosine similarity. Thus, our sense embeddings become:

$$\vec{v}_s = \begin{bmatrix} ||\vec{v}_s||_2 \\ ||\vec{v}_d||_2 \end{bmatrix}, dim(\vec{v}_s) = 2048$$