Word Sense Filtering Improves Embedding-Based Lexical Substitution

Anne Cocos[*], **Marianna Apidianaki**[*†] and **Chris Callison-Burch**[*]
[*] Computer and Information Science Department, University of Pennsylvania
† LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay
{acocos,marapi,ccb}@seas.upenn.edu

Abstract

The role of word sense disambiguation in lexical substitution has been questioned due to the high performance of vector space models which propose good substitutes without explicitly accounting for sense. We show that a filtering mechanism based on a sense inventory optimized for substitutability can improve the results of these models. Our sense inventory is constructed using a clustering method which generates paraphrase clusters that are congruent with lexical substitution annotations in a development set. The results show that lexical substitution can still benefit from senses which can improve the output of vector space paraphrase ranking models.

1 Introduction

Word sense has always been difficult to define and pin down (Kilgarriff, 1997; Erk et al., 2013). Recent successes of embedding-based, sense-agnostic models in various semantic tasks cast further doubt on the usefulness of word sense. Why bother to identify senses if even humans cannot agree upon their nature and number, and if simple word-embedding models yield good results without using any explicit sense representation?

Word-based models are successful in various semantic tasks even though they conflate multiple word meanings into a single representation. Based on the hypothesis that capturing polysemy could further improve their performance, several works have focused on creating sense-specific word embeddings. A common approach is to cluster the contexts in which the words appear in a corpus to induce senses, and relabel each word token with the clustered sense before learning embeddings (Reisinger and Mooney, 2010; Huang et al., 2012). Iacobacci et al. (2015) disambiguate the words in a corpus using a state-of-the-art WSD system and then produce continuous representations of word senses based on distributional information obtained from the annotated corpus. Moving from word to sense embeddings generally improves their performance in word and relational similarity tasks but is not beneficial in all settings. Li and Jurafsky (2015) show that although multi-sense embeddings give improved performance in tasks such as semantic similarity, semantic relation identification and part-of-speech tagging, they fail to help in others, like sentiment analysis and named entity extraction (Li and Jurafsky, 2015).

We show how a sense inventory optimized for *substitutability* can improve the rankings provided by two sense-agnostic, vector-based lexical substitution models. Lexical substitution requires systems to predict substitutes for target word instances that preserve their meaning in context (McCarthy and Navigli, 2007). We consider a sense inventory with high substitutability to be one which groups synonyms or paraphrases that are mutually-interchangeable in the same contexts. In contrast, sense inventories with low substitutability might group words linked by different types of relations. We carry out experiments with a syntactic vector-space model (Thater et al., 2011; Apidianaki, 2016) and a word-embedding model for lexical substitution (Melamud et al., 2015). Instead of using the senses to refine the vector representations as in (Faruqui et al., 2015), we use them to improve the lexical substitution rankings proposed by the models as a post-processing step. Our results show that senses can improve the performance of vector-space models in lexical substitution tasks.

Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, pages 110–119,
Valencia, Spain, April 4 2017.

2 A sense inventory for substitution

2.1 Paraphrase substitutability

The candidate substitutes used by our ranking models come from the Paraphrase Database (PPDB) XXL package (Ganitkevitch et al., 2013).[1] Paraphrase relations in the PPDB are defined between words and phrases which might carry different senses. Cocos and Callison-Burch (2016) used a spectral clustering algorithm to cluster PPDB XXL into senses, but the clusters contain noisy paraphrases and paraphrases linked by different types of relations (e.g. hypernyms, antonyms) which are not always substitutable. We use a slightly modified version of their method to cluster paraphrases where both the number of clusters (senses) and their contents are optimized for substitutability.

2.2 A measure of substitutability

We define a substitutability metric that quantifies the extent to which a sense inventory aligns with human-generated lexical substitution annotations. We then cluster PPDB paraphrases using the substitutability metric to optimize the sense clusters for substitutability.

Given a sense inventory C, we can define the senses of a target word t as a set of sense clusters, $C(t) = \{c_1, c_2, \ldots c_k\}$, where each cluster contains words corresponding to a single sense of t. Intuitively, if a sense inventory corresponds with substitutability, then each sense cluster c_i should have two qualities: first, words within c_i should be interchangeable with t in the same set of contexts; and second, c_i should not be missing any words that are interchangeable in those same contexts. We therefore operationalize the definition of substitutability as follows.

We begin measuring substitutability with a lexical substitution dataset, consisting of sentences where content words have been manually annotated with substitutes (see example in Table 1). We then use normalized mutual information (NMI) (Strehl and Ghosh, 2002) to quantify the level of agreement between the automatically generated sense clusters and human-suggested substitutes. NMI is an information theoretic measure of cluster quality. Given two clusterings U and V over a set of items, it measures how much each clustering reduces uncertainty about the other (Vinh et al., 2009) in terms of their mutual information $I(U,V)$ and entropies $H(U), H(V)$:

$$NMI(U,V) = \frac{I(U,V)}{\sqrt{H(U)H(V)}}$$

To calculate the NMI between a sense inventory for target word t and its set of annotated substitutes, we first define the substitutes as a clustering, $B_t = \{b_1, b_2, \ldots b_n\}$, where b_i denotes the set of suggested substitutes for each of n sentences. Table 1, for example, gives the clustered substitutes for $n = 3$ sentences for target word $t =$ word.N, where $b_1 = \{$vow, utterance, tongue, speech$\}$. We then define the substitutability of the sense inventory, C_t, with respect to the annotated substitutes, B_t, as $NMI(C_t, B_t)$.[2] Given many target words, we can further aggregate the substitutability of sense inventory C over the set of targets T in B into a single substitutability score:

$$substitutability_B(C) = \sum_{t \in T} \frac{NMI(C_t, B_t)}{|T|}$$

Sentence	Annotated Substitutes (Count)
In this world, one's **word** is a promise.	vow (1), utterance (1), tongue (1), speech (1)
Silverplate: code **word** for the historic mission that would end World War II.	phrase (3), term (2), verbiage(1), utterance (1), signal (1), name (1), dictate (1), designation (1), decree (1)
I think she only heard the last **words** of my speech.	bit (3), verbiage (2), part (2), vocabulary (1), terminology (1), syllable (1), phrasing (1), phrase (1), patter (1), expression (1), babble (1), anecdote (1)

Table 1: Example annotated sentences for the target word *word.N* from the CoInCo (Kremer et al., 2014) lexical substitution dataset. Numbers after each word indicate the number of annotators who made that suggestion.

2.3 Optimizing for Substitutability

Having defined a substitutability score, we now automatically generate word sense clusters from the Paraphrase Database that maximize it. The idea is to use the substitutability score to choose the best number of senses for each target word which will be the number of output clusters (k) generated by our spectral clustering algorithm.

[1] PPDB paraphrases come into packages of different sizes (going from S to XXXL): small packages contain high-precision paraphrases while larger ones have high coverage. All are available from `paraphrase.org`

[2] In calculating NMI, we ignore words that do not appear in both C_t and B_t.

2.4 Spectral clustering method

2.4.1 Constructing the Affinity Matrix

The spectral clustering algorithm (Yu and Shi, 2003) takes as input an affinity matrix $A \in \mathbb{R}^{n \times n}$ encoding n items to be clustered, and an integer k. It generates k non-overlapping clusters of the n items. Each entry a_{ij} in the affinity matrix denotes a similarity measurement between items i and j. Entries in A must be nonnegative and symmetric. The affinity matrix can also be thought of as describing a graph, where the n rows and columns correspond to nodes, and each entry a_{ij} gives the weight of an edge between nodes i and j. Because the matrix must be symmetric, the graph is undirected.

Given a target word t, we call its set of PPDB paraphrases $PP(t)$. Note that $PP(t)$ excludes t itself. In our most basic clustering method, we cluster paraphrases for target t as follows. Given the length-n set of t's paraphrases, $PP(t)$, we construct the $n \times n$ affinity matrix A where each shared-index row and column corresponds to some word $p \in PP(t)$. We set entries equal to the cosine similarity between the applicable words' embeddings, plus one: $a_{ij} = cos(v_i, v_j) + 1$ (to enforce non-negative similarities). For our implementation we use 300-dimensional part-of-speech-specific word embeddings v_i generated using the gensim word2vec package (Mikolov et al., 2013b; Mikolov et al., 2013a; Řehůřek and Sojka, 2010).[3] In Figure 1a we show a set of paraphrases, linked by PPDB relations, and in Figure 1b we show the corresponding basic affinity matrix, encoding the paraphrases' distributional similarity.

In order to aid further discussion, we point out that the affinity matrix used for the basic clustering method encodes a fully-connected graph $G = \{PP(t), E_{PP}^{ALL}\}$ with paraphrases $PP(t)$ as nodes, and edges between every pair of words, $E_{PP}^{ALL} = PP(t) \times PP(t)$. As for all variations on the clustering method, the matrix entries correspond to distributional similarity.

2.4.2 Masking

The affinity matrix in Figure 1b ignores the graph structure inherent in PPDB, where edges connect only words that are paraphrases of one another. We experiment with enforcing the PPDB structure in the affinity matrix through a technique we call 'masking.' By masking, we mean allowing positive values in the affinity matrix only where the row and column correspond to paraphrases that appear as pairs in PPDB (Figure 1a). All entries corresponding to paraphrase pairs that are *not* connected in the PPDB graph (Figure 1a) are forced to 0.

More concretely, in the masked affinity matrix, we set each entry a_{ij} for which i and j are *not* paraphrases in PPDB to zero. The masked affinity matrix encodes the graph $G = \{PP(t), E_{PP}^{MASK}\}$ with edges connecting only pairs of words that are in PPDB, $E_{PP}^{MASK} = \{(p_i, p_j) \mid p_i \in P(p_j)\}$. Figure 1c shows the masked affinity matrix corresponding to the PPDB structure in Figure 1a.

2.4.3 Optimizing *k*

Because spectral clustering requires the number of output clusters, k, to be specified as input, for each target word we run the clustering algorithm for a range of k between 1 and the minimum of (n, 20). We then choose the k that maximizes the NMI of the resulting clusters with the human-annotated substitutes for that target in the development data.

2.5 Method variations

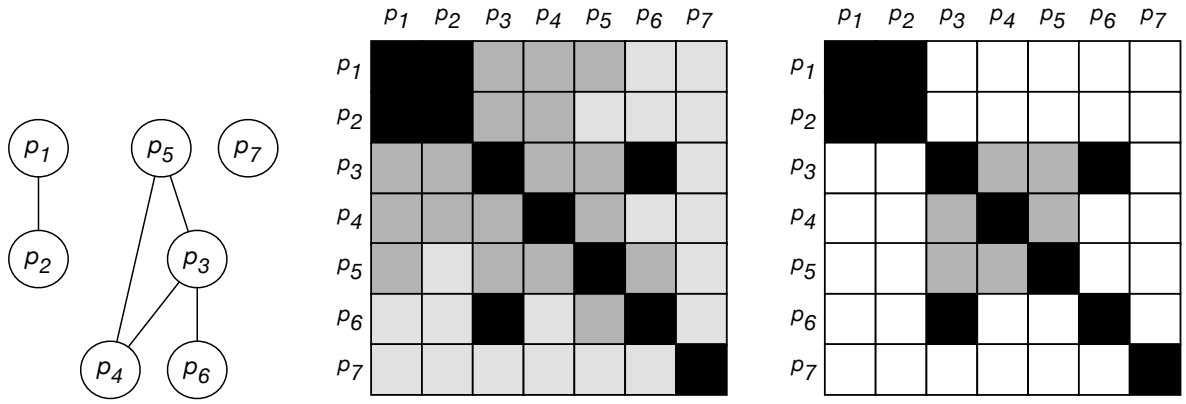
In addition to using the substitutability score to choose the best number of senses for each target word, we also experiment with two variations on the basic spectral clustering method to increase the score further: filtering by a paraphrase confidence score and co-clustering with WordNet (Fellbaum, 1998).

2.5.1 PPDB Score Thresholding

Each paraphrase pair in the PPDB is associated with a set of scores indicating the strength of the paraphrase relationship. The recently added PPDB2.0 Score (Pavlick et al., 2015) was calculated using a supervised scoring model trained on human judgments of paraphrase quality.[4] Apidianaki (2016) showed that the PPDB2.0 Score itself is a good metric for ranking substitution candidates in context, outperforming some vector space models when the number of candidates is high. With this in mind, we experimented with using a PPDB2.0 Score threshold to discard noisy PPDB

[3] The `word2vec` parameters we use are a context window of size 3, learning rate *alpha* from 0.025 to 0.0001, minimum word count 100, sampling parameter $1e^{-4}$, 10 negative samples per target word, and 5 training epochs.

[4] The human judgments were used to fit a regression to the features available in PPDB 1.0 plus numerous new features including cosine word embedding similarity, lexical overlap features, WordNet features and distributional similarity features.



(a) PPDB graph with $n = 7$ paraphrases $PP(t)$ to be clustered.

(b) *Unmasked* affinity matrix for input to basic clustering algorithm, for paraphrases in Fig 1a. This matrix encodes a fully-connected graph $G = \{PP(t), E_{PP}^{ALL}\}$.

(c) *Masked* affinity matrix for input to clustering algorithm, enforcing the paraphrase links from the graph in Fig 1a, $G = \{PP(t), E_{PP}^{MASK}\}$.

Figure 1: Unclustered PPDB graph and its corresponding affinity matrices, encoding distributional similarity, for input to the basic (1b) and masked (1c) clustering algorithms. Masking zeros-out the values of all cells corresponding to paraphrases not connected in the PPDB graph. Cell shading corresponds to the distributional similarity score between words, with darker colors representing higher measurements.

XXL paraphrases prior to sense clustering. Our objective was to begin the clustering process with a *clean* set of paraphrases for each target word, eliminating erroneous paraphrases that might pollute the substitutable sense clusters. We implemented PPDB score thresholds in a range from 0 to 2.5.

2.5.2 Co-Clustering with WordNet

PPDB is large and inherently noisy. WordNet has smaller coverage but well-defined semantic structure in the form of synsets and relations. We sought a way to marry the high coverage of PPDB with the clean structure of WordNet by co-clustering the two resources, in hopes of creating a sense inventory that is both highly-substitutable and high-coverage.

The basic unit in WordNet is the *synset*, a set of lemmas sharing the same meaning. WordNet also connects synsets via relations, such as hypernymy, hyponymy, entailment, and 'similar-to'. We denote as $L(s)$ the set of lemmas associated with synset s. We denote as $R(s)$ the set of synsets that are related to synset s with a hypernym, hyponym, entailment, or similar-to relationship. Finally, we denote as $S(t)$ the set of synsets to which a word t belongs. We denote as $S^+(t)$ the set of t's synsets, plus all synsets to which they are related; $S^+(t) = S(t) \cup \bigcup_{s' \in S(t)} S(s')$. In other words, $S^+(t)$ includes all synsets to which t is connected by a path of length at most 2 via one of the relations encoded in $R(s)$.

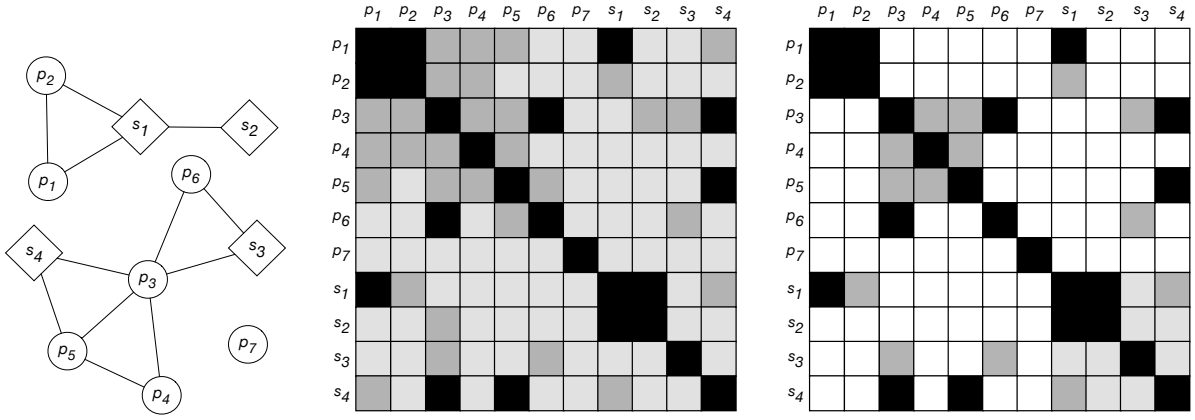
For co-clustering, we generate the affinity matrix for a graph with $m+n$ nodes corresponding to the n words in $PP(t)$ and the m synsets in $S^+(t)$, and edges between every pair of nodes. Because the edge weights are cosine similarity between vector embeddings, we need a way to construct an embedding for each synset in $S^+(t)$.[5] We therefore generate compositional embeddings for each synset s that are equal to the weighted average of the embeddings for the lemmas $l \in L(s)$, where the weights are the PPDB2.0 scores between t and l:

$$v_s = \frac{\sum_{l \in L(s)} PPDB2.0Score(t,l) \times v_l}{\sum_{l \in L(s)} PPDB2.0Score(t,l)}$$

The unmasked affinity matrix used for input to the co-clustering method, then, encodes the graph $G = \{PP(t) \cup S^+(t), E_{PP}^{ALL} \cup E_{PS}^{ALL} \cup E_{SS}^{ALL}\}$, where E_{PS}^{ALL} contains edges between every paraphrase and synset, and E_{SS}^{ALL} contains edges between every pair of synsets.

We also define masked versions of the co-clustering affinity matrix. In a masked affinity matrix, positive entries are only allowed for entries where the row and column correspond to enti-

[5] We don't use the NASARI embeddings (Camacho-Collados et al., 2015) because these are available only for nouns.



(a) Graph showing $n = 7$ PPDB paraphrases $PP(t)$ and $m = 4$ WordNet synsets $S^+(t)$ to be clustered.

(b) *Unmasked* affinity matrix for input to basic clustering algorithm, for paraphrases and synsets in Fig 2a. This matrix encodes a fully-connected graph $G = \{PP(t) \cup S^+(t), E_{PP}^{ALL} \cup E_{PS}^{ALL} \cup E_{SS}^{ALL}\}$.

(c) Affinity matrix for input to clustering algorithm, enforcing the paraphrase-paraphrase (E_{PP}^{MASK}) and paraphrase-synset (E_{PS}^{MASK}) links from the graph in 2a, but allowing all synset-synset links (E_{PP}^{ALL}).

Figure 2: Paraphrase/synset graph for input to the co-clustering model, and its corresponding affinity matrices for the basic (2b) and masked (2c) clustering algorithms. Cell shading corresponds to the distributional similarity score between words/synsets.

ties (paraphrases or synsets) that are connected by the underlying knowledge base (PPDB or WordNet). Just as we defined masking for paraphrase-paraphrase links (E_{PP}) to allow only positive values corresponding to paraphrase pairs found in PPDB, here we separately define masking for paraphrase-synset (E_{PS}) and synset-synset (E_{SS}) based on WordNet synsets and relations. When applying the clustering algorithm, it is possible to elect to use the masked version for any or all of E_{PP}, E_{PS}, and E_{SS}. In our experiments we try all combinations.

For the synset-synset links, we define the masked version E_{SS}^{MASK} as including only nonzero edge weights where a hypernym, hyponym, entailment or similar-to relationship connects two synsets: $E_{SS}^{MASK} = \{(s_u, s_v) \mid s_u \in R(s_v) \text{ or } s_v \in R(s_u)\}$. For the paraphrase-synset links, we define the masked version E_{PS}^{MASK} to include only nonzero edge weights where the paraphrase *is* a lemma in the synset, or *is a paraphrase of* a lemma in the synset (excluding the target word): $E_{PS}^{MASK} = \{(p_i, s_u) \mid p_i \in L(s_u) \text{ or } |(P(p_i) - t) \cap L(s_u)| > 0\}$. We need to exclude the target word when calculating the overlap because otherwise all words in $PP(t)$ would connect to all synsets in $S(t)$. Figure 2 depicts the graph, unnmasked and masked affinity matrices for the co-clustering method.

2.6 Clustering Experiments

2.6.1 Datasets

We run clustering experiments using targets and human-generated substitution data drawn from two lexical substitution datasets. The first is the "Concepts in Context" (CoInCo) corpus (Kremer et al., 2014), containing over 15K sentences corresponding to nearly 4K unique target words. We divide the CoInCo dataset into development and test sets by first finding all target words that have at least 10 sentences. For each of the 327 resulting targets, we randomly divide the corresponding sentences into 60% development instances and 40% test instances. The resulting development and test sets have 4061 and 2091 sentences respectively. We cluster the 327 target words in the resulting subset of CoInCo, performing all optimization using the development portion.

In order to evaluate how well our method generalizes to other data, we also create clusters for target words drawn from the SemEval 2007 English Lexical Substitution shared task dataset (McCarthy and Navigli, 2007). The entire test portion of the SemEval dataset contains 1700 annotated sentences for 170 target words. We filter this data to keep only sentences with one or more human-annotated substitutes that overlap our PPDB XXL paraphrase vocabulary. The resulting test set, which we use for evaluating SemEval

targets, has 1178 sentences and 157 target words. We cluster each of the 157 targets, using the CoInCo development data to optimize substitutability for the 32 SemEval targets that also appear in CoInCo. For the rest of the SemEval targets we choose a number of senses equal to its WordNet synset count.

2.6.2 Clustering Method Variations

We try all combinations of the following parameters in our clustering model:

- Clustering method: We try the basic clustering – clustering Paraphrases Only – and the WordNet Co-Clustering method.
- PPDB2.0 Score Threshold: We cluster paraphrases of each target having a PPDB2.0 Score above a threshold, ranging from 0-3.0.
- Masking: When clustering paraphrases only, we can either use the PP-PP mask or allow positive similarities between all words. When co-clustering, we try all combinations of the PP-PP, PP-SYN, and SYN-SYN masks.

For each combination, we evaluate the NMI substitutability of the resulting sense inventory over our CoInCo and SemEval test instances. The substitutability results are given in Tables 2 and 3.

3 Filtering Substitutes

3.1 A WSD oracle

We now question whether it is possible to improve the rankings of current state-of-the-art lexical substitution systems by using the optimized sense inventory as a filter. Our general approach is to take a set of ranked substitutes generated by a vector-based model. Then, we see whether filtering the ranked substitutes to bring words belonging to the correct sense of the target to the top of the rankings would improve the overall ranking results.

Assuming that we have a WSD oracle that is able to choose the most appropriate sense for a target word in context, this corresponds to nominating substitutes from the applicable sense cluster and elevating them in the list of ranked substitutes output by the state-of-the-art lexical substitution system. If sense filtering successfully improves the quality of ranked substitutes, we can say that the sense inventory captures substitutability well.

3.2 Ranking Models

Our approach requires a set of rankings produced by a high-quality lexical substitution model to start. We generate substitution rankings for each target/sentence pair in the test sets using a syntactic vector-space model (Thater et al., 2011; Apidianaki, 2016) and a state-of-the-art model based on word embeddings (Melamud et al., 2015).

The syntactic vector space model of Apidianaki (2016) (Syn.VSM) demonstrated an ability to correctly choose appropriate PPDB paraphrases for a target word in context. The vector features correspond to syntactic dependency triples extracted from the English Gigaword corpus [6] analyzed with Stanford dependencies (Marneffe et al., 2006). Syn.VSM produces a score for each (target, sentence, substitute) tuple based on the cosine similarity of the substitute's basic vector representation with the target's contextualized vector (Thater et al., 2011). The contextualized vector is derived from the basic meaning vector of the target word by reinforcing its dimensions that are licensed by the context of the specific instance under consideration. More specifically, the contextualized vector of a target is obtained through vector addition and contains information about the target's direct syntactic dependents.

The second set of rankings comes from the AddCos model of Melamud et al. (2015). AddCos quantifies the fit of substitute word s for target word t in context C by measuring the semantic similarity of the substitute to the target, and the similarity of the substitute to the context:

$$AddCos(s,t,W) = \frac{|W| \cdot cos(s,t) + \sum_{w \in W} cos(s,w)}{2 \cdot |W|} \quad (1)$$

The vectors s and t are word embeddings of the substitute and target generated by the *skip-gram with negative sampling* model (Mikolov et al., 2013b; Mikolov et al., 2013a). The context W is the set of words appearing within a fixed-width window of the target t in a sentence (we use a window (cwin) of 1), and the embeddings c are context embeddings generated by *skip-gram*. In our implementation, we train 300-dimensional word and context embeddings over the 4B words in the Annotated Gigaword (AGiga) corpus (Napoles et al., 2012) using the gensim word2vec package

[6] http://catalog.ldc.upenn.edu/LDC2003T05

(Mikolov et al., 2013b; Mikolov et al., 2013a; Řehůřek and Sojka, 2010).[7]

3.3 Substitution metrics

Lexical substitution experiments are usually evaluated using generalized average precision (GAP) (Kishida, 2005). GAP compares a set of predicted rankings to a set of gold standard rankings. Scores range from 0 to 1; a perfect ranking, in which all high-scoring substitutes outrank low-scoring substitutes, has a score of 1. For each sentence in the CoInCo and SemEval test sets, we consider the PPDB paraphrases for the target word to be the candidates, and we set the test set annotator frequency to be the gold score. Words in PPDB that were not suggested by annotators receive a gold score of 0.001. Predicted scores are given by the two ranking models, Syn.VSM and AddCos.

3.4 Filtering Method

Sense filtering is intended to boost the rank of substitutes that belong to the most appropriate sense of the target given the context. We run this experiment as a two-step process.

First, given a target and sentence, we obtain the PPDB XXL paraphrases for the target word and rank them using the Syn.VSM and the AddCos models.[8] We calculate the overall *unfiltered* GAP score on the test set for each ranking model as the average GAP over sentences.

Next, we evaluate the ability of a sense inventory to improve the GAP score through filtering. We implement sense filtering by adding a large number (10000) to the ranking model's score for words *belonging to a single sense*. We assume an oracle that finds the cluster which maximally improves the GAP score using this sense filtering method. If the sense inventory corresponds well to substitutability, we should expect this filtering to improve the ranking by downgrading proposed substitutes that do not fall within the correct sense cluster.

We calculate the maximum sense-restricted GAP score for the inventories produced by each variation on our clustering model, and compare this to the unfiltered GAP score for each ranking model.

3.5 Baselines

We compare the extent to which our optimized sense inventories improve lexical substitution rankings to the results of two baseline sense inventories.

- **WordNet+**: a sense inventory formed from WordNet 3.0. For each CoInCo target word that appears in WordNet, we take its sense clusters to be its synsets, plus lemmas belonging to hypernyms and hyponyms of each synset.
- **PPDBClus**: a much larger, abeit noisier, sense inventory obtained by automatically clustering words in the PPDB XXL package. To obtain this sense inventory we clustered paraphrases for all targets in the CoInCo dataset using the method outlined in Cocos and Callison-Burch (2016), with PPDB2.0 Score serving as the similarity metric.

We assess the substitutability of these sense baseline inventories with respect to the human-annotated substitutes in the CoInCo and SemEval datasets, and also use them for sense filtering.

Finally, we wish to estimate the impact of the NMI-based optimization procedure (Section 2.4.3) on the quality of the senses used for filtering. We compare the performance of the optimized CoInCo sense inventory, where the number of clusters, k, for a target word is defined through NMI optimization (called 'Choose-K: Optimize NMI'), to an inventory induced from CoInCo where k equals the number of synsets available for the target word in WordNet (called 'Choose-K: #WN Synsets).

4 Results

We report substitutability, and the unfiltered and best sense-filtered GAP scores achieved using the paraphrase-only clustering method and the co-clustering method in Tables 2 and 3.

The average unfiltered GAP scores for the Syn.VSM rankings over the CoInCo and SemEval test sets are 0.528 and 0.673 respectively.[9] All

[7] The `word2vec` training parameters we use are a context window of size 3, learning rate *alpha* from 0.025 to 0.0001, minimum word count 100, sampling parameter $1e^{-4}$, 10 negative samples per target word, and 5 training epochs.

[8] For the SemEval test set, we rank PPDB XXL paraphrases having a PPDB2.0 Score with the target of at least 2.54.

[9] The score reported by Apidianaki (2016) for the Syn.VSM model with XXL PPDB paraphrases on CoInCo was 0.56. The difference in scores is due to excluding from our clustering experiments target words that did not have at least 10 sentences in CoInCo.

		Syn.VSM		AddCos (cwin=1)	
	$subst_{CoInCo}$	Unfiltered GAP	Oracle GAP	Unfiltered GAP	Oracle GAP
PPDBClus	0.254	0.528	0.661	0.533	0.656
WordNet	0.252		0.655		0.651
Choose-K: # WN Synsets (avg)	0.205		0.639		0.636
Choose-K: # WN Synsets (max, no co-clustering)	0.250*		0.695*		0.690*
Choose-K: # WN Synsets (max, co-clustering)	0.241**		0.690**		0.683**
Choose-K: Optimize NMI (avg)	0.282		0.668		0.662
Choose-K: Optimize NMI (max, no co-clustering)	0.331*		0.719 *		0.714 ***
Choose K: Optimize NMI (max, co-clustering)	0.314**		0.718 ****		0.710 **

Table 2: Substitutablity (NMI) of resulting sense inventories, and GAP scores of the unfiltered and best sense-filtered rankings produced by the Syn.VSM and AddCos models, for the CoInCo annotated dataset. Configurations for the best-performing sense inventories were: * Min PPDB Score 2.0, cluster PP's only, use PP-PP mask; ** Min PPDB Score 2.0, co-clustering, use PP-PP mask only; *** Min PPDB Score 1.5, cluster PP's only, use PP-PP mask; **** Min PPDB Score 2.0, co-clustering, use PP-PP, Syn-Syn masks only

		Syn.VSM		AddCos (cwin=1)	
	$subst_{SemEval}$	Unfiltered GAP	Oracle GAP	Unfiltered GAP	Oracle GAP
PPDBClus	0.357	0.673	0.855	0.410	0.634
WordNet	0.291		0.774		0.595
Average of all Clustered Sense Inventories	0.367		0.841		0.569
Max basic (no co-clustering) sense inventory	0.448*		0.917*		0.626*
Max co-clustered sense inventory	0.449**		0.906***		0.612****

Table 3: Substitutablity (NMI) of resulting sense inventories, and GAP scores of the unfiltered and best sense-filtered rankings produced by the Syn.VSM and AddCos models, for the SemEval07 annotated dataset. Configurations for the best-performing sense inventories were: * Min PPDB Score 2.31, cluster PP's only, use PP-PP mask; ** Min PPDB Score 2.54, co-clustering, use PP-SYN mask only; *** Min PPDB Score 2.54, co-clustering, use PP-SYN mask only; **** Min PPDB Score 2.31, co-clustering, use PP-SYN mask only.

baseline and cluster sense inventories are capable of improving these GAP scores when we use the best sense as a filter. Syntactic models generally give very good results with small paraphrase sets (Kremer et al., 2014) but their performance seems to degrade when they need to deal with larger and noisier substitute sets (Apidianaki, 2016). Our results suggest that finding the most appropriate sense of a target word in context *can* improve their lexical substitution results.

The trend in results is similar for the AddCos rankings. The average unfiltered GAP scores for the AddCos rankings over the CoInCo and SemEval test sets are 0.533 and 0.410 respectively. The GAP scores of the unfiltered AddCos rankings are much lower than after filtering with any baseline or cluster sense inventory, showing that lexical subsitutition rankings based on word-embeddings can also be improved using senses.

To assess the impact of the NMI-based optimization procedure on the results, we compare the performance of two sense inventories on the CoInCo rankings: one where the number of clusters (k) for a target word is defined through NMI optimization and another one, where k is equal to the number of synsets available for the target word in WordNet. We find that for both the Syn.VSM and AddCos ranking models, filtering using the sense inventory with the NMI-optimized k outperforms the results obtained when the inventory with k equal to the number of synsets is used.

Furthermore, we find that the NMI substitutability score is a generally good indicator of how much improvement we see in GAP score due to oracle sense filtering. We calculated the Pearson correlation of a sense inventory's NMI with its oracle GAP score to be 0.644 (calculated over all target words in the CoInCo test set, including GAP results for both the Syn.VSM and AddCos ranking models). This suggests that NMI is a reasonable measure of substitutability.

We find that for all methods, applying a PPDB-Score Threshold prior to clustering is an effective way of removing noisy, non-substitutable paraphrases from the sense inventory. When we use the resulting sense inventory for filtering, this effectively elevates only high-quality paraphrases in the lexical substitution rankings. This supports the finding of Apidianaki (2016), who showed that the PPDB2.0 Score itself is an effective lexical substitution ranking metric when large and noisy paraphrase substitute sets are involved.

Finally, we discover that co-clustering with WordNet did not produce any significant improvement in NMI or GAP score over clustering paraphrases alone. This could suggest that the added structure from WordNet did not improve overall substitutability of the resulting sense inventories, or that our co-clustering method did not effectively incorporate useful structural information from WordNet.

5 Conclusion

We have shown that despite the high performance of word-based vector-space models, lexical substitution can still benefit from word senses. We have defined a substitutability metric and proposed a method for automatically creating sense inventories optimized for substitutability. The number of sense clusters in an optimized inventory and their contents are aligned with lexical substitution annotations in a development set. Using the best fitting cluster in each context as a filter over the rankings produced by vector-space models boosts good substitutes and improves the models' scores in a lexical substitution task.

For choosing the cluster that best fits a context, we used an oracle experiment which finds the maximum GAP score achievable by a sense by boosting the ranking model's score for words *belonging to a single sense*. The cluster that achieved the highest GAP score in each case was selected. The task of finding the most appropriate sense in context still remains. But the improvement in lexical substitution results shows that word sense induction and disambiguation can still benefit state-of-the-art word-based models for lexical substitution.

Our sense filtering mechanism can be applied to the output of any vector-space substitution model at a post-processing step. In future work, we intend to experiment with models that account for senses during embedding learning. The models of Huang et al. (2012) and Li and Jurafsky (2015) learn multi-prototype, or sense-specific, embedding representations and are able to choose the best-fitted ones for words in context. These models have up to now been tested in several NLP tasks but have not yet been applied to lexical substitution. We will experiment with using the embeddings chosen by these models for specific word instances for ranking candidate substitutes in context. The comparison with the results presented in this paper will show whether it is preferable to account for senses before or after actual lexical substitution.

Acknowledgments

This material is based in part on research sponsored by DARPA under grant number FA8750-13-2-0017 (the DEFT program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

This work has also been supported by the French National Research Agency under project ANR-16-CE33-0013.

We would like to thank our anonymous reviewers for their thoughtful and helpful comments.

References

Marianna Apidianaki. 2016. Vector-space models for PPDB paraphrase ranking in context. In *Proceedings of EMNLP*, pages 2028–2034, Austin, Texas.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. NASARI: a Novel Approach to a Semantically-Aware Representation of Items. In *Proceedings of NAACL/HLT 2015*, pages 567–577, Denver, Colorado.

Anne Cocos and Chris Callison-Burch. 2016. Clustering paraphrases by word sense. In *Proceedings of NAACL/HLT 2016*, pages 1463–1472, San Diego, California.

Katrin Erk, Diana McCarthy, and Nicholas Gaylord. 2013. Measuring word meaning in context. *Computational Linguistics*, 39(3):511–554, 9.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of NAACL/HLT*, Denver, Colorado.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Juri Ganitkevitch, Benjamin VanDurme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL*, Atlanta, Georgia.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the ACL (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of the ACL/IJCNLP*, pages 95–105, Beijing, China.

Adam Kilgarriff. 1997. I don't believe in word senses. *Computers and the Humanities*, 31(2):91–113.

Kazuaki Kishida. 2005. *Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments*. National Institute of Informatics Tokyo, Japan.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What Substitutes Tell Us - Analysis of an "All-Words" Lexical Substitution Corpus. In *Proceedings of EACL*, pages 540–549, Gothenburg, Sweden.

Jiwei Li and Dan Jurafsky. 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding? In *Proceedings of the EMNLP*, pages 1722–1732, Lisbon, Portugal.

M. Marneffe, B. Maccartney, and C. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC-2006*, Genoa, Italy.

Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 Task 10: English Lexical Substitution Task. In *Proceedings of SemEval*, pages 48–53, Prague, Czech Republic.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A Simple Word Embedding Model for Lexical Substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montreal, Canada.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL/IJCNLP*, pages 425–430, Beijing, China.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-Prototype Vector-Space Models of Word Meaning. In *Proceedings of HLT/NAACL*, pages 109–117, Los Angeles, California.

Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word Meaning in Context: A Simple and Effective Vector Model. In *Proceedings of IJCNLP*, pages 1134–1143, Chiang Mai, Thailand.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2009. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080. ACM.

Stella X. Yu and Jianbo Shi. 2003. Multiclass spectral clustering. In *Proceedings of International Conference on Computer Vision (ICCV 03)*, pages 313–319. IEEE.