Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors

Marco Baroni and **Georgiana Dinu** and **Germán Kruszewski**
Center for Mind/Brain Sciences (University of Trento, Italy)
(marco.baroni|georgiana.dinu|german.kruszewski)@unitn.it

Abstract

Context-predicting models (more commonly known as embeddings or neural language models) are the new kids on the distributional semantics block. Despite the buzz surrounding these models, the literature is still lacking a systematic comparison of the predictive models with classic, count-vector-based distributional semantic approaches. In this paper, we perform such an extensive evaluation, on a wide range of lexical semantics tasks and across many parameter settings. The results, to our own surprise, show that the buzz is fully justified, as the context-predicting models obtain a thorough and resounding victory against their count-based counterparts.

1 Introduction

A long tradition in computational linguistics has shown that contextual information provides a good approximation to word meaning, since semantically similar words tend to have similar contextual distributions (Miller and Charles, 1991). In concrete, *distributional semantic models* (DSMs) use vectors that keep track of the contexts (e.g., co-occurring words) in which target terms appear in a large corpus as proxies for meaning representations, and apply geometric techniques to these vectors to measure the similarity in meaning of the corresponding words (Clark, 2013; Erk, 2012; Turney and Pantel, 2010).

It has been clear for decades now that raw co-occurrence counts don't work that well, and DSMs achieve much higher performance when various transformations are applied to the raw vectors, for example by reweighting the counts for context informativeness and smoothing them with dimensionality reduction techniques. This vector optimization process is generally unsupervised, and based on independent considerations (for example, context reweighting is often justified by information-theoretic considerations, dimensionality reduction optimizes the amount of preserved variance, etc.). Occasionally, some kind of indirect supervision is used: Several parameter settings are tried, and the best setting is chosen based on performance on a semantic task that has been selected for tuning.

The last few years have seen the development of a new generation of DSMs that frame the vector estimation problem directly as a supervised task, where the weights in a word vector are set to maximize the probability of the contexts in which the word is observed in the corpus (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2013a; Turian et al., 2010). The traditional construction of context vectors is turned on its head: Instead of first collecting context vectors and then reweighting these vectors based on various criteria, the vector weights are directly set to optimally predict the contexts in which the corresponding words tend to appear. Since similar words occur in similar contexts, the system naturally learns to assign similar vectors to similar words.

This new way to train DSMs is attractive because it replaces the essentially heuristic stacking of vector transforms in earlier models with a single, well-defined supervised learning step. At the same time, supervision comes at no manual annotation cost, given that the context windows used for training can be automatically extracted from an unannotated corpus (indeed, they are the very same data used to build traditional DSMs). Moreover, at least some of the relevant methods can efficiently scale up to process very large amounts of input data.[1]

[1]The idea to directly learn a parameter vector based on an objective optimum function is shared by Latent Dirichlet

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 238–247,
Baltimore, Maryland, USA, June 23-25 2014.

We will refer to DSMs built in the traditional way as *count* models (since they initialize vectors with co-occurrence counts), and to their training-based alternative as *predict(ive)* models.[2] Now, the most natural question to ask, of course, is which of the two approaches is best in empirical terms. Surprisingly, despite the long tradition of extensive evaluations of alternative count DSMs on standard benchmarks (Agirre et al., 2009; Baroni and Lenci, 2010; Bullinaria and Levy, 2007; Bullinaria and Levy, 2012; Sahlgren, 2006; Padó and Lapata, 2007), the existing literature contains very little in terms of direct comparison of count vs. predictive DSMs. This is in part due to the fact that context-predicting vectors were first developed as an approach to language modeling and/or as a way to initialize feature vectors in neural-network-based "deep learning" NLP architectures, so their effectiveness as semantic representations was initially seen as little more than an interesting side effect. Sociological reasons might also be partly responsible for the lack of systematic comparisons: Context-predictive models were developed within the neural-network community, with little or no awareness of recent DSM work in computational linguistics.

Whatever the reasons, we know of just three works reporting direct comparisons, all limited in their scope. Huang et al. (2012) compare, in passing, one count model and several predict DSMs on the standard WordSim353 benchmark (Table 3 of their paper). In this experiment, the count model actually outperforms the best predictive approach. Instead, in a word-similarity-in-context task (Table 5), the best predict model outperforms the count model, albeit not by a large margin.

Blacoe and Lapata (2012) compare count and predict representations as input to composition functions. Count vectors make for better inputs in a phrase similarity task, whereas the two representations are comparable in a paraphrase classification experiment.[3]

Finally, Mikolov et al. (2013d) compare their predict models to "Latent Semantic Analysis" (LSA) count vectors on syntactic and semantic analogy tasks, finding that the predict models are highly superior. However, they provide very little details about the LSA count vectors they use.[4]

In this paper, we overcome the comparison scarcity problem by providing a direct evaluation of count and predict DSMs across many parameter settings and on a large variety of mostly standard lexical semantics benchmarks. Our title already gave away what we discovered.

2 Distributional semantic models

Both count and predict models are extracted from a corpus of about 2.8 billion tokens constructed by concatenating ukWaC,[5] the English Wikipedia[6] and the British National Corpus.[7] For both model types, we consider the top 300K most frequent words in the corpus both as target and context elements.

2.1 Count models

We prepared the count models using the DISSECT toolkit.[8] We extracted count vectors from symmetric context windows of two and five words to either side of target. We considered two weighting schemes: positive Pointwise Mutual Information and Local Mutual Information (akin to the widely used Log-Likelihood Ratio scheme) (Evert, 2005). We used both full and compressed vectors. The latter were obtained by applying the Singular Value Decomposition (Golub and Van Loan, 1996) or Non-negative Matrix Factorization (Lee and Seung, 2000), Lin (2007) algorithm, with reduced sizes ranging from 200 to 500 in steps of 100. In total, 36 count models were evaluated.

Count models have such a long and rich history that we can only explore a small subset of the counting, weighting and compressing methods proposed in the literature. However, it is worth pointing out that the evaluated parameter subset encompasses settings (narrow context window, positive PMI, SVD reduction) that have been

Allocation (LDA) models (Blei et al., 2003; Griffiths et al., 2007), where parameters are set to optimize the joint probability distribution of words and documents. However, the fully probabilistic LDA models have problems scaling up to large data sets.

[2] We owe the first term to Hinrich Schütze (p.c.). Predictive DSMs are also called neural language models, because their supervised context prediction training is performed with neural networks, or, more cryptically, "embeddings".

[3] We refer here to the updated results reported in the erratum at `http://homepages.inf.ed.ac.uk/s1066731/pdf/emnlp2012erratum.pdf`

[4] Chen et al. (2013) present an extended empirical evaluation, that is however limited to alternative context-predictive models, and does not include the word2vec variant we use here.

[5] `http://wacky.sslmit.unibo.it`

[6] `http://en.wikipedia.org`

[7] `http://www.natcorp.ox.ac.uk`

[8] `http://clic.cimec.unitn.it/composes/toolkit/`

found to be most effective in the systematic explorations of the parameter space conducted by Bullinaria and Levy (2007; 2012).

2.2 Predict models

We trained our predict models with the word2vec toolkit.[9] The toolkit implements both the skip-gram and CBOW approaches of Mikolov et al. (2013a; 2013c). We experimented only with the latter, which is also the more computationally-efficient model of the two, following Mikolov et al. (2013b) which recommends CBOW as more suitable for larger datasets.

The CBOW model learns to predict the word in the middle of a symmetric window based on the sum of the vector representations of the words in the window. We considered context windows of 2 and 5 words to either side of the central element. We vary vector dimensionality within the 200 to 500 range in steps of 100. The word2vec toolkit implements two efficient alternatives to the standard computation of the output word probability distributions by a softmax classifier. Hierarchical softmax is a computationally efficient way to estimate the overall probability distribution using an output layer that is proportional to $log(unigram.perplexity(W))$ instead of W (for W the vocabulary size). As an alternative, negative sampling estimates the probability of an output word by learning to distinguish it from draws from a noise distribution. The number of these draws (number of *negative samples*) is given by a parameter k. We test both hierarchical softmax and negative sampling with k values of 5 and 10. Very frequent words such as *the* or *a* are not very informative as context features. The word2vec toolkit implements a method to downsize their effect (and simultaneously improve speed performance). More precisely, words in the training data are discarded with a probability that is proportional to their frequency (capturing the same intuition that motivates traditional count vector weighting measures such as PMI). This is controlled by a parameter t and words that occur with higher frequency than t are aggressively subsampled. We train models without subsampling and with subsampling at $t = 1e^{-5}$ (the toolkit page suggests $1e^{-3} - 1e^{-5}$ as a useful range based on empirical observations).

In total, we evaluate 48 predict models, a number comparable to that of the count models we consider.

2.3 Out-of-the-box models

Baroni and Lenci (2010) make the vectors of their best-performing *Distributional Memory* (dm) model available.[10] This model, based on the same input corpus we use, exemplifies a "linguistically rich" count-based DSM, that relies on lemmas instead or raw word forms, and has dimensions that encode the syntactic relations and/or lexico-syntactic patterns linking targets and contexts. Baroni and Lenci showed, in a large scale evaluation, that dm reaches near-state-of-the-art performance in a variety of semantic tasks.

We also experiment with the popular predict vectors made available by Ronan Collobert.[11] Following the earlier literature, with refer to them as *Collobert and Weston* (cw) vectors. These are 100-dimensional vectors trained for two months (!) on the Wikipedia. In particular, the vectors were trained to optimize the task of choosing the right word over a random alternative in the middle of an 11-word context window (Collobert et al., 2011).

3 Evaluation materials

We test our models on a variety of benchmarks, most of them already widely used to test and compare DSMs. The following benchmark descriptions also explain the figures of merit and state-of-the-art results reported in Table 2.

Semantic relatedness A first set of semantic benchmarks was constructed by asking human subjects to rate the degree of semantic similarity or relatedness between two words on a numerical scale. The performance of a computational model is assessed in terms of correlation between the average scores that subjects assigned to the pairs and the cosines between the corresponding vectors in the model space (following the previous art, we use Pearson correlation for rg, Spearman in all other cases). The classic data set of Rubenstein and Goodenough (1965) (rg) consists of 65 noun pairs. State of the art performance on this set has been reported by Hassan and Mihalcea (2011) using a technique that exploits the Wikipedia linking structure and word sense disambiguation techniques. Finkelstein et al. (2002)

[9] https://code.google.com/p/word2vec/

[10] http://clic.cimec.unitn.it/dm/

[11] http://ronan.collobert.com/senna/

introduced the widely used WordSim353 set (ws) that, as the name suggests, consists of 353 pairs. The current state of the art is reached by Halawi et al. (2012) with a method that is in the spirit of the predict models, but lets synonymy information from WordNet constrain the learning process (by favoring solutions in which WordNet synonyms are near in semantic space). Agirre et al. (2009) split the ws set into similarity (wss) and relatedness (wsr) subsets. The first contains tighter taxonomic relations, such as synonymy and cohyponymy (*king/queen*) whereas the second encompasses broader, possibly topical or syntagmatic relations (*family/planning*). We report state-of-the-art performance on the two subsets from the work of Agirre and colleagues, who used different kinds of count vectors extracted from a very large corpus (orders of magnitude larger than ours). Finally, we use (the test section of) MEN (men), that comprises 1,000 word pairs. Bruni et al. (2013), the developers of this benchmark, achieve state-of-the-art performance by extensive tuning on *ad-hoc* training data, and by using both textual and image-extracted features to represent word meaning.

Synonym detection The classic TOEFL (toefl) set was introduced by Landauer and Dumais (1997). It contains 80 multiple-choice questions that pair a target term with 4 synonym candidates. For example, for the target *levied* one must choose between *imposed* (correct), *believed*, *requested* and *correlated*. The DSMs compute cosines of each candidate vector with the target, and pick the candidate with largest cosine as their answer. Performance is evaluated in terms of correct-answer accuracy. Bullinaria and Levy (2012) achieved 100% accuracy by a very thorough exploration of the count model parameter space.

Concept categorization Given a set of nominal concepts, the task is to group them into natural categories (e.g., *helicopters* and *motorcycles* should go to the *vehicle* class, *dogs* and *elephants* into the *mammal* class). Following previous art, we tackle categorization as an unsupervised clustering task. The vectors produced by a model are clustered into n groups (with n determined by the gold standard partition) using the CLUTO toolkit (Karypis, 2003), with the repeated bisections with global optimization method and CLUTO's default settings otherwise (these are standard choices in the literature). Performance is evaluated in terms of *purity*, a measure of the extent to which each cluster contains concepts from a single gold category. If the gold partition is reproduced perfectly, purity reaches 100%; it approaches 0 as cluster quality deteriorates. The Almuhareb-Poesio (ap) benchmark contains 402 concepts organized into 21 categories (Almuhareb, 2006). State-of-the-art purity was reached by Rothenhäusler and Schütze (2009) with a count model based on carefully crafted syntactic links. The ESSLLI 2008 Distributional Semantic Workshop shared-task set (esslli) contains 44 concepts to be clustered into 6 categories (Baroni et al., 2008) (we ignore here the 3- and 2-way higher-level partitions coming with this set). Katrenko and Adriaans (2008) reached top performance on this set using the full Web as a corpus and manually crafted, linguistically motivated patterns. Finally, the Battig (battig) test set introduced by Baroni et al. (2010) includes 83 concepts from 10 categories. Current state of the art was reached by the window-based count model of Baroni and Lenci (2010).

Selectional preferences We experiment with two data sets that contain verb-noun pairs that were rated by subjects for the typicality of the noun as a subject or object of the verb (e.g., *people* received a high average score as subject of *to eat*, and a low score as object of the same verb). We follow the procedure proposed by Baroni and Lenci (2010) to tackle this challenge: For each verb, we use the corpus-based tuples they make available to select the 20 nouns that are most strongly associated to the verb as subjects or objects, and we average the vectors of these nouns to obtain a "prototype" vector for the relevant argument slot. We then measure the cosine of the vector for a target noun with the relevant prototype vector (e.g., the cosine of *people* with the *eating* subject prototype vector). Systems are evaluated by Spearman correlation of these cosines with the averaged human typicality ratings. Our first data set was introduced by Ulrike Padó (2007) and includes 211 pairs (up). Top-performance was reached by the supervised count vector system of Herdağdelen and Baroni (2009) (supervised in the sense that they directly trained a classifier on gold data, as opposed to the 0-cost supervision of the context-learning methods). The mcrae set (McRae et al., 1998) consists of 100 noun–verb pairs, with top performance reached by the DepDM system of Baroni and Lenci (2010), a count DSM relying on

syntactic information.

Analogy While all the previous data sets are relatively standard in the DSM field to test traditional count models, our last benchmark was introduced in Mikolov et al. (2013a) specifically to test predict models. The data-set contains about 9K semantic and 10.5K syntactic analogy questions. A semantic question gives an example pair (*brother-sister*), a test word (*grandson*) and asks to find another word that instantiates the relation illustrated by the example with respect to the test word (*granddaughter*). A syntactic question is similar, but in this case the relationship is of a grammatical nature (*work–works*, *speak… speaks*). Mikolov and colleagues tackle the challenge by subtracting the second example term vector from the first, adding the test term, and looking for the nearest neighbour of the resulting vector (what is the nearest neighbour of $\vec{brother} - \vec{sister} + \vec{grandson}$?). Systems are evaluated in terms of proportion of questions where the nearest neighbour from the whole semantic space is the correct answer (the given example and test vector triples are excluded from the nearest neighbour search). Mikolov et al. (2013a) reach top accuracy on the syntactic subset (ansyn) with a CBOW predict model akin to ours (but trained on a corpus twice as large). Top accuracy on the entire data set (an) and on the semantic subset (ansem) was reached by Mikolov et al. (2013c) using a skip-gram predict model. Note however that, because of the way the task is framed, performance also depends on the size of the vocabulary to be searched: Mikolov et al. (2013a) pick the nearest neighbour among vectors for 1M words, Mikolov et al. (2013c) among 700K words, and we among 300K words.

Some characteristics of the benchmarks we use are summarized in Table 1.

4 Results

Table 2 summarizes the evaluation results. The first block of the table reports the maximum per-task performance (across all considered parameter settings) for count and predict vectors. The latter emerge as clear winners, with a large margin over count vectors in most tasks. Indeed, the predictive models achieve an impressive overall performance, beating the current state of the art in several cases, and approaching it in many more. It is worth stressing that, as reviewed in Section 3, the state-of-the-art results were obtained in almost all cases using specialized approaches that rely on external knowledge, manually-crafted rules, parsing, larger corpora and/or task-specific tuning. Our predict results were instead achieved by simply downloading the word2vec toolkit and running it with a range of parameter choices recommended by the toolkit developers.

The success of the predict models cannot be blamed on poor performance of the count models. Besides the fact that this would not explain the near-state-of-the-art performance of the predict vectors, the count model results are actually quite good in absolute terms. Indeed, in several cases they are close, or even better than those attained by dm, a linguistically-sophisticated count-based approach that was shown to reach top performance across a variety of tasks by Baroni and Lenci (2010).

Interestingly, count vectors achieve performance comparable to that of predict vectors only on the selectional preference tasks. The up task in particular is also the only benchmark on which predict models are seriously lagging behind state-of-the-art and dm performance. Recall from Section 3 that we tackle selectional preference by creating average vectors representing typical verb arguments. We conjecture that this averaging approach, that worked well for dm vectors, might be problematic for prediction-trained vectors, and we plan to explore alternative methods to build the prototypes in future research.

Are our results robust to parameter choices, or are they due to very specific and brittle settings? The next few blocks of Table 2 address this question. The second block reports results obtained with single count and predict models that are best in terms of average performance rank across tasks (these are the models on the top rows of tables 3 and 4, respectively). We see that, for both approaches, performance is not seriously affected by using the single best setup rather than task-specific settings, except for a considerable drop in performance for the best predict model on esslli (due to the small size of this data set?), and an even more dramatic drop of the count model on ansem. A more cogent and interesting evaluation is reported in the third block of Table 2, where we see what happens if we use the single models with *worst* performance across tasks (recall from Section 2 above that, in any case, we are exploring a space of reasonable parameter settings, of the sort that an

name	task	measure	source	soa
rg	relatedness	Pearson	Rubenstein and Goodenough (1965)	Hassan and Mihalcea (2011)
ws	relatedness	Spearman	Finkelstein et al. (2002)	Halawi et al. (2012)
wss	relatedness	Spearman	Agirre et al. (2009)	Agirre et al. (2009)
wsr	relatedness	Spearman	Agirre et al. (2009)	Agirre et al. (2009)
men	relatedness	Spearman	Bruni et al. (2013)	Bruni et al. (2013)
toefl	synonyms	accuracy	Landauer and Dumais (1997)	Bullinaria and Levy (2012)
ap	categorization	purity	Almuhareb (2006)	Rothenhäusler and Schütze (2009)
esslli	categorization	purity	Baroni et al. (2008)	Katrenko and Adriaans (2008)
battig	categorization	purity	Baroni et al. (2010)	Baroni and Lenci (2010)
up	sel pref	Spearman	Padó (2007)	Herdağdelen and Baroni (2009)
mcrae	sel pref	Spearman	McRae et al. (1998)	Baroni and Lenci (2010)
an	analogy	accuracy	Mikolov et al. (2013a)	Mikolov et al. (2013c)
ansyn	analogy	accuracy	Mikolov et al. (2013a)	Mikolov et al. (2013a)
ansem	analogy	accuracy	Mikolov et al. (2013a)	Mikolov et al. (2013c)

Table 1: Benchmarks used in experiments, with type of task, figure of merit (measure), original reference (source) and reference to current state-of-the-art system (soa).

	rg	ws	wss	wsr	men	toefl	ap	esslli	battig	up	mcrae	an	ansyn	ansem
best setup on each task														
cnt	74	62	70	59	72	76	66	84	98	41	27	49	43	60
pre	84	75	**80**	**70**	**80**	91	75	86	**99**	41	28	**68**	**71**	**66**
best setup across tasks														
cnt	70	62	70	57	72	76	64	84	98	37	27	43	41	44
pre	83	73	78	68	**80**	86	71	77	98	41	26	67	69	64
worst setup across tasks														
cnt	11	16	23	4	21	49	24	43	38	-6	-10	1	0	1
pre	74	60	73	48	68	71	65	82	88	33	20	27	40	10
best setup on rg														
cnt	(74)	59	66	52	71	64	64	84	98	37	20	35	42	26
pre	(84)	71	76	64	79	85	72	84	98	39	25	66	70	61
other models														
soa	**86**	**81**	77	62	76	**100**	**79**	**91**	96	**60**	**32**	61	64	61
dm	82	35	60	13	42	77	76	84	94	51	29	NA	NA	NA
cw	48	48	61	38	57	56	58	61	70	28	15	11	12	9

Table 2: Performance of count (cnt), predict (pre), dm and cw models on all tasks. See Section 3 and Table 1 for figures of merit and state-of-the-art results (soa). Since dm has very low coverage of the an* data sets, we do not report its performance there.

experimenter might be tempted to choose without tuning). The count model performance is severely affected by this unlucky choice (2-word window, Local Mutual Information, NMF, 400 dimensions, mean performance rank: 83), whereas the predict approach is much more robust: To put its worst instantiation (2-word window, hierarchical softmax, no subsampling, 200 dimensions, mean rank: 51) into perspective, its performance is more than 10% below the *best* count model only for the an and ansem tasks, and actually higher than it in 3 cases (note how on esslli the worst predict models performs much better than the best one, confirming our suspicion about the brittleness of this small data set). The fourth block reports performance in what might be the most realistic scenario, namely by tuning the parameters on a development task. Specifically, we pick the models that work best on the small rg set, and report their performance on all tasks (we obtained similar results by picking other tuning sets). The selected count model is the third best overall model of its class as reported in Table 3. The selected predict model is the fourth best model in Table 4. The overall count performance is not greatly affected by this choice. Again, predict models confirm their robustness, in that their rg-tuned performance is always close (and in 3 cases better) than the one achieved by the best overall setup.

Tables 3 and 4 let us take a closer look at the most important count and predict parameters, by reporting the characteristics of the best models (in terms of average performance-based ranking across tasks) from both classes. For the count models, PMI is clearly the better weighting scheme, and SVD outperforms NMF as a dimensionality reduction technique. However, no compression at all (using all 300K original dimensions) works best. Compare this to the best overall predict vectors, that have 400 dimensions only, making them much more practical to use. For the predict models, we observe in Table 4 that negative sampling, where the task is to distinguish the target output word from samples drawn from the noise distribution, outperforms the more costly hierarchical softmax method. Subsampling frequent words, which downsizes the importance of these words similarly to PMI weighting in count models, is also bringing significant improvements.

Finally, we go back to Table 2 to point out the poor performance of the out-of-the-box cw model.

window	weight	compress	dim.	mean rank
2	PMI	no	300K	35
5	PMI	no	300K	38
2	PMI	SVD	500	42
2	PMI	SVD	400	46
5	PMI	SVD	500	47
2	PMI	SVD	300	50
5	PMI	SVD	400	51
2	PMI	NMF	300	52
2	PMI	NMF	400	53
5	PMI	SVD	300	53

Table 3: Top count models in terms of mean performance-based model ranking across all tasks. The first row states that the window-2, PMI, 300K count model was the best count model, and, across all tasks, its average rank, when ALL models are decreasingly ordered by performance, was 35. See Section 2.1 for explanation of the parameters.

We must leave the investigation of the parameters that make our predict vectors so much better than cw (more varied training corpus? window size? objective function being used? subsampling? ...) to further work. Still, our results show that it's not just training by context prediction that ensures good performance. The cw approach is very popular (for example both Huang et al. (2012) and Blacoe and Lapata (2012) used it in the studies we discussed in Section 1). Had we also based our systematic comparison of count and predict vectors on the cw model, we would have reached opposite conclusions from the ones we can draw from our word2vec-trained vectors!

5 Conclusion

This paper has presented the first systematic comparative evaluation of count and predict vectors. As seasoned distributional semanticists with thorough experience in developing and using count vectors, we set out to conduct this study because we were annoyed by the triumphalist overtones often surrounding predict models, despite the almost complete lack of a proper comparison to count vectors.[12] Our secret wish was to discover that it is all hype, and count vectors are far superior to their predictive counterparts. A more realistic expec-

[12]Here is an example, where word2vec is called the crown jewel of natural language processing: `http://bit.ly/1ipv72M`

win.	hier. softm.	neg. samp.	subsamp.	dim	mean rank
5	no	10	yes	400	10
2	no	10	yes	300	13
5	no	5	yes	400	13
5	no	5	yes	300	13
5	no	10	yes	300	13
2	no	10	yes	400	13
2	no	5	yes	400	15
5	no	10	yes	200	15
2	no	10	yes	500	15
2	no	5	yes	300	16

Table 4: Top predict models in terms of mean performance-based model ranking across all tasks. See Section 2.2 for explanation of the parameters.

tation was that a complex picture would emerge, with predict and count vectors beating each other on different tasks. Instead, we found that the predict models are so good that, while the triumphalist overtones still sound excessive, there are very good reasons to switch to the new architecture. However, due to space limitations we have only focused here on quantitative measures: It remains to be seen whether the two types of models are complementary in the errors they make, in which case combined models could be an interesting avenue for further work.

The space of possible parameters of count DSMs is very large, and it's entirely possible that some options we did not consider would have improved count vector performance somewhat. Still, given that the predict vectors also outperformed the syntax-based dm model, and often approximated state-of-the-art performance, a more proficuous way forward might be to focus on parameters and extensions of the predict models instead: After all, we obtained our already excellent results by just trying a few variations of the word2vec defaults. Add to this that, beyond the standard lexical semantics challenges we tested here, predict models are currently been successfully applied in cutting-edge domains such as representing phrases (Mikolov et al., 2013c; Socher et al., 2012) or fusing language and vision in a common semantic space (Frome et al., 2013; Socher et al., 2013).

Based on the results reported here and the considerations we just made, we would certainly recommend anybody interested in using DSMs for theoretical or practical applications to go for the predict models, with the important caveat that they are not all created equal (cf. the big difference between word2vec and cw models). At the same time, given the large amount of work that has been carried out on count DSMs, we would like to explore, in the near future, how certain questions and methods that have been considered with respect to traditional DSMs will transfer to predict models. For example, the developers of Latent Semantic Analysis (Landauer and Dumais, 1997), Topic Models (Griffiths et al., 2007) and related DSMs have shown that the dimensions of these models can be interpreted as general "latent" semantic domains, which gives the corresponding models some *a priori* cognitive plausibility while paving the way for interesting applications. Another important line of DSM research concerns "context engineering": There has been for example much work on how to encode syntactic information into context features (Padó and Lapata, 2007), and more recent studies construct and combine feature spaces expressing topical vs. functional information (Turney, 2012). To give just one last example, distributional semanticists have looked at whether certain properties of vectors reflect semantic relations in the expected way: e.g., whether the vectors of hypernyms "distributionally include" the vectors of hyponyms in some mathematical precise sense.

Do the dimensions of predict models also encode latent semantic domains? Do these models afford the same flexibility of count vectors in capturing linguistically rich contexts? Does the structure of predict vectors mimic meaningful semantic relations? Does all of this even matter, or are we on the cusp of discovering radically new ways to tackle the same problems that have been approached as we just sketched in traditional distributional semantics?

Either way, the results of the present investigation indicate that these are important directions for future research in computational semantics.

Acknowledgments

We acknowledge ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasça, and Aitor Soroa. 2009.

A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of HLT-NAACL*, pages 19–27, Boulder, CO.

Abdulrahman Almuhareb. 2006. *Attributes in Lexical Acquisition*. Phd thesis, University of Essex.

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni, Stefan Evert, and Alessandro Lenci, editors. 2008. *Bridging the Gap between Semantic Theory and Computational Simulations: Proceedings of the ESSLLI Workshop on Distributional Lexical Semantic*. FOLLI, Hamburg.

Marco Baroni, Eduard Barbu, Brian Murphy, and Massimo Poesio. 2010. Strudel: A distributional semantic model based on properties and types. *Cognitive Science*, 34(2):222–254.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2013. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*. In press; `http://clic.cimec.unitn.it/marco/publications/mmds-jair.pdf`.

John Bullinaria and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

John Bullinaria and Joseph Levy. 2012. Extracting semantic representations from word co-occurrence statistics: Stop-lists, stemming and SVD. *Behavior Research Methods*, 44:890–907.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou', and Steven Skiena. 2013. The expressive power of word embeddings. In *Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, GA. Published online: `https://sites.google.com/site/deeplearningicml2013/accepted_papers`.

Stephen Clark. 2013. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd ed.* Blackwell, Malden, MA. In press; `http://www.cl.cam.ac.uk/~sc609/pubs/sem_handbook.pdf`.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, Helsinki, Finland.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences*. Ph.D dissertation, Stuttgart University.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Andrea Frome, Greg Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A deep visual-semantic embedding model. In *Proceedings of NIPS*, pages 2121–2129, Lake Tahoe, Nevada.

Gene Golub and Charles Van Loan. 1996. *Matrix Computations (3rd ed.)*. JHU Press, Baltimore, MD.

Tom Griffiths, Mark Steyvers, and Josh Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:211–244.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of KDD*, pages 1406–1414.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *Proceedings of AAAI*, pages 884–889, San Francisco, CA.

Amaç Herdağdelen and Marco Baroni. 2009. BagPack: A general framework to represent semantic relations. In *Proceedings of GEMS*, pages 33–40, Athens, Greece.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

George Karypis. 2003. CLUTO: A clustering toolkit. Technical Report 02-017, University of Minnesota Department of Computer Science.

Sophia Katrenko and Pieter Adriaans. 2008. Qualia structures and their impact on the concrete noun categorization task. In *Proceedings of the ESSLLI Workshop on Distributional Lexical Semantics*, pages 17–24, Hamburg, Germany.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Daniel Lee and Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of NIPS*, pages 556–562.

Chih-Jen Lin. 2007. Projected gradient methods for Nonnegative Matrix Factorization. *Neural Computation*, 19(10):2756–2779.

Ken McRae, Michael Spivey-Knowlton, and Michael Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:283–312.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. `http://arxiv.org/abs/1301.3781/`.

Tomas Mikolov, Quoc Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for Machine Translation. `http://arxiv.org/abs/1309.4168`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, Lake Tahoe, Nevada.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*, pages 746–751, Atlanta, Georgia.

George Miller and Walter Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Ulrike Padó. 2007. *The Integration of Syntax and Semantic Plausibility in a Wide-Coverage Model of Sentence Processing*. Dissertation, Saarland University, Saarbrücken.

Klaus Rothenhäusler and Hinrich Schütze. 2009. Unsupervised classification with dependency based word spaces. In *Proceedings of GEMS*, pages 17–24, Athens, Greece.

Herbert Rubenstein and John Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D dissertation, Stockholm University.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Richard Socher, Milind Ganjoo, Christopher Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Proceedings of NIPS*, pages 935–943, Lake Tahoe, Nevada.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394, Uppsala, Sweden.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.