array instead of the entire data. This run-time complexity is faster than other competing methods, for example, Meltzer and Hovy's (2011) method was $\sim 200X$ slower.

Model Elements

We propose a new conditional model to construct a probabilistic combination function, essentially measuring similarity between two entities based on a function over shared (common) set of features, as discussed below:

Contextual Query Phrase Relevance (CQR): Contextual Query Phrase Relevance (CQR) is a measure of how important the query phrase is to its contexts as compared to other phrases occurring together with them:

$$CQR(x,q) = P(X = x|Q = q) = \frac{p(x,q)}{p(q)} = \frac{f(x,q)}{f(q)}$$

where p(•) and $f(\bullet)$ are probability and frequency points, respectively, in the distribution.

Candidate Contextual Strength (CCS): Candidate Contextual Strength (CCS) is a measure of how strongly related the query phrase contexts are to the potential near synonym candidate phrases as compared to other local contexts surrounding them:

$$CCS(y,x) = P(Y = y|X = x) = \frac{p(y,x)}{p(x)} = \frac{f(y,x)}{f(x)}$$

Normalization: In order to address base-level frequency variation among candidate phrases we introduce a normalization factor: $k(y) = \big(f(y)\big)^{-d}$, where d is a constant.

Contextual Information (Inf): Some contexts still carry more semantic information than others based on their content (e.g., type and/or number of words) and our model tries to take that into account. Therefore, $Inf(x) = a * w(x) + b * l(x) + c$, where $w(x)$ is the number of content words in context x, $l(x)$ is the length of x, and a, b and c are coefficients.

Shared Feature Gain Scoring Fuction

Combining the concepts described above, we compute the score, first for left contexts ($L(q)$):

$$S_L(y,q) = \sum_{x \in L(q)} CQR(x,q)CCS(y,x)k(y)Inf(x)$$

The model also accounts for *displaced contextual matches*, that are essentially cradle matches but with the left and right matching at different instances of the query:

$$S'_L(y,q) = \sum_{x \in L(q)} \begin{cases} 2 * CQR.CCS.k.Inf & if\ x \in DL(q) \\ CQR.CCS.k.Inf & otherwise \end{cases}$$

where $DL(q)$ is a subset of $L(q)$ which qualifies as displaces lefts. Similarly, we compute scores for rights and cradles and combine the three to get the final score:

$$S(y,q) = HM + C_{cf}.S_C(y,q) \qquad (1)$$

$$HM = 2\frac{S'_L S'_R}{S'_L + S'_R}$$

where $C_{cf} > 1$ to boost the score for cradle matches S_C.

Kullback-Leibler Divergence Scoring Function

KL divergence (Cover and Thomas 1991) is measure of the difference between two probability distributions. We use it to measure the information lost when the contextual distribution given a candidate is used to approximate the same contextual distribution given the query phrase:

$$S_L(y,q) = \sum_{x \in L(q)} p(x|q) \log\left(\frac{p(x|q)}{r(x|y)}\right)$$

$$p(x|q) = \frac{p(x,q) + c'}{p(q) + c'|L(q)|} \;\&\; r(x|y) = \frac{r(x,y) + c''}{r(y) + c''|L(q)|}$$

$$c' = p(q)\frac{0.001}{|L(q)|} \;\&\; c'' = r(y)\frac{0.001}{|L(q)|}$$

where $L(q)$ represents the combined set of lefts for the query phrase and the candidate. As before, the ratio of the probabilities $p(\bullet)$ and $r(\bullet)$, can be interpreted as the ratio of frequencies. We apply smoothing and also compute the scores for the combined rights and combined cradles, then combine the three to get the final score:

$$S(y,q) = -S_L(y,q)S_R(y,q)S_C(y,q)^2 \qquad (2)$$

We re-score and re-rank the top 1000 scoring candidates generated by the shared feature gain using Equation 2.

Parameter Training

Equation 1 contains the parameters a, b, c, and d separately for S'_L, S'_R and S_C each along with the cradle boosting parameter C_{cf}, for a total of 13 parameters. One possible parameter training scheme, is to generate training data consisting of query phrases (Q), and pick near-synonym candidates rated as highly synonymous by human judges. A natural optimization objective would then be:

$$\sum\nolimits_{q \in Q} \sum\nolimits_{y \in Y(q)} \|S(y_{best},q) - S(y,q)\|$$

with the constraint that all the parameters > 0. $S(y,q)$ is a product of two nonnegative convex functions, and is therefore convex. This makes the optimization objective a difference of two convex functions (DC class) and its direct optimization is reserved for future work. For the present we relied on multi-start coordinate ascent with binary search instead of increasing the linear step size increase. The parameters were trained on a set of 30 query phrases, separate from the ones used in the evaluation (see section Experiments).

Experiments

The Gigaword Corpus

We selected the very large English Gigaword Fifth Edition

Method	MRŞ(5)	MRŞ(10)	MRŞ(15)	MRŞ(20)
SF	2.35	2.19	2.12	2.00
KL	**2.45**	**2.28**	**2.17**	**2.08**
PPDB	1.97	1.80	1.62	1.48
Mavuno	2.04	1.83	1.75	1.64
Thesaurus	1.18	1.09	1.00	0.95

Table 1: Significant MRŞ improvements for both scoring functions (SF and KL) over PPDB, Mavuno and Roget's Thesaurus, for 23 two word query phrases

Method	MRŞ(5)	MRŞ(10)	MRŞ(15)	MRŞ(20)
SF	**2.15**	**1.99**	1.85	1.76
KL	2.10	1.99	**1.89**	**1.84**
PPDB	1.65	1.57	1.48	1.38
Mavuno	1.85	1.76	1.71	1.65
Thesaurus	0.50	0.47	0.43	0.43

Table 2: Significant MRŞ improvements for both scoring functions (SF and KL) over PPDB, Mavuno and Roget's Thesaurus, for 16 greater than two word query phrases

(Parker et al. 2011), a comprehensive archive of newswire text data, for our experiments. The corpus was split into 32 equal parts with a suffix array constructed from each split. Since, the server hardware can support up to 32 (16x2) threads in parallel, each suffix array operates on a separate thread of its own. We used 37.5% of the data (12 suffix arrays, ~1.51 billion words) for our experiments. The full Gigaword may have yielded better results, but would have run slower.

Rank-Sensitive Evaluation

For our experiments, we chose a set of 54 randomly selected query phrases including 15 single word, 23 two word phrases, and 16 longer phrases[1]. For each query phrase, 20 near-synonym candidates were generated using each of the two scoring functions and baselines. The annotators (6 human judges) were asked to provide ratings on each query phrase-synonym candidate combination. The ratings scaled from 0-3 (Rubenstein and Goodenough 1965), where 3 indicates absolute synonymy (Zgusta 1971), 2 indicates near-synonymy (Hirst 1995), 1 indicates some semantic correlation such as hypernymy, hyponymy or antonymy and 0 indicates no relationship. The inter-annotator agreement was measure to be $\kappa = 0.43$ (Fleiss 1971), using binary categories for ratings 2, 3, and 0, 1, respectively, which is moderate. When the two categories were modified to 1, 2, 3 vs 0, it measured $\kappa = 0.90$ which is almost perfect agreement (Landis and Koch 1977).

[1] The query phrases, annotations and other results can be downloaded at http://www.cs.cmu.edu/~dishang/.

Method	MRŞ(5)	MRŞ(10)	MRŞ(15)	MRŞ(20)
SF	2.22	2.00	1.90	1.79
KL	1.98	1.84	1.76	1.65
PPDB	1.42	1.30	1.23	1.16
Mavuno	2.00	1.79	1.64	1.55
Thesaurus	**2.88**	**2.83**	**2.81**	**2.80**
H&S	0.27	0.29	0.28	0.26

Table 3: Significant MRŞ improvements for both scoring functions (SF and KL) over PPDB, Mavuno and H&S Model, for 15 single word query phrases

We extended the standard performance measures: Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (nDCG). We did not use MAP directly because it is rank-insensitive, and is valid only for binary (0 or 1, relevant or irrelevant) rating scale. In the case of nDCG, even though it does take ordering into account it does not penalize for inferior results. For example, in our experiments the rating sequence 2, 2, 2 for the top 3 retrieval results of a query phrase would get a higher score as compared to the sequence 3, 2, 3, whereas the latter is clearly superior in quality. Besides this, nDCG does not penalize for missing results (recall) either. Our normalized metric, the mean rank-sensitive score ($MR\S$), which devalues the annotated scores for lower ranks (further from top rank) is:

$$MR\S(n) = \frac{1}{|A|}\frac{1}{|Q|}\sum_{a \in A}\sum_{q \in Q}\frac{\sum_{r=1}^{n} S_r[\log_2(r+1)]^{-1}}{\sum_{r=1}^{n}[\log_2(r+1)]^{-1}}$$

where S_r is the annotated score, n is the cutoff at the n^{th} rank, r is the rank of the candidate and A is the set of raters. $MR\S$ takes into account missing results by padding the rating sequence with zeros for the missing values. Also, due to normalization $MR\S$ is insensitive to the length of the rating sequence, i.e., $MR\S(3)$ for 2, 2, 2 is equal to $MR\S(5)$ for 2, 2, 2, 2, 2.

Multi-Word and Single Word Comparisons

Can NeSS really perform better than thesauri, at least for multi-word phrases, and other systems in the literature?

Roget's Thesaurus: To show the inadequacy of thesauri lookup for phrasal synonyms, we compare our model to a baseline from the Roget's Thesaurus. Since, like all other thesauri it primarily contains single words, we combine elements in the synonym sets of individual words in the query phrase to construct candidates for each of the 54 query phrases. For instance, in *"strike a balance"* we randomly select "hammer" and "harmony" as synonyms for "strike" and "balance", respectively, to form "hammer a harmony"

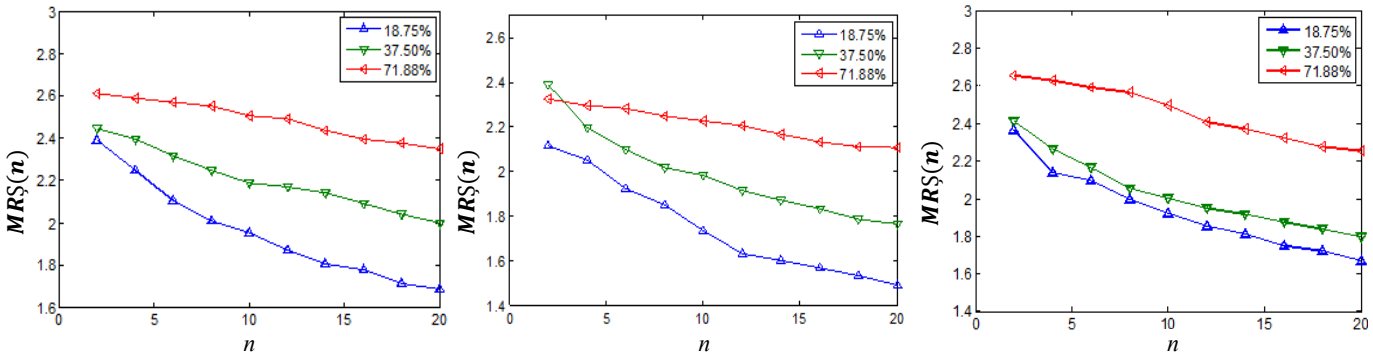

*Figure 2: **MRŞ(n)** plots for 2 word (left), > 2 word (middle) and single word (right) phrases, using shared feature gain function, for 18.75%, 37.50% and 71.88% of the Gigaword corpus. NeSS's retrieval quality improves with increasing corpus size.*

as a candidate. We assume 100% thesaurus precision for single word thesaurus entries (it is a published thesaurus), and for the rest we again employ 3 human judges. Tables 1, 2 and 3 compare the *MRŞ* scores for shared feature gain, KL divergence and the thesaurus for two word, greater than two word and single word query phrases, separately. We can clearly see the performance advantage of our approach increases in query phrase length. Like the rating scale, *MRŞ* ranges from 0 to 3 and thus, a difference of more than 1 and 1.3 at each cutoff for two word and greater than two word query phrases, respectively, further signifies the considerable superiority of our methods over thesauri composition. Note that both functions peak at the two word level, and shared feature gain performs stronger for single word queries whereas KL divergence takes the lead for longer ones.

Since *MRŞ* is insensitive to cutoff point due to normalization, the observation that both our scoring functions produce greater scores at stricter cutoffs (i.e. lower values of n) implies that our model is able to discriminate stronger semantic matches from relatively weaker ones and ranks the highly synonymous candidates higher.

The Paraphrase Database: We also compare our methods to the machine translation technique by Ganitkevitch et al. (2013), PPDB 1.0. The English portion of PPDB 1.0 contains over 220 million paraphrases. We extracted the top 20 near-synonyms for our 54 query phrases from the 73 million phrasal and 8 million lexical paraphrase pairs, using the Annotated Gigaword distributional similarity scores provided in the database for ranking the candidates. Again, 6 human judges provided the ratings. Again, from Tables 1, 2 and 3, it is clear that our methods are better at ranking and recall at every cutoff point as well as phrase length. Considering the fact that NeSS operates on a monolingual corpus, does not require any NLP specific resources, and is a live retrieval system, as compared to PPDB which is none of the above, this is quite a significant result.

Mavuno: We also compare with Mavuno, an open-source Hadoop-based scalable paraphrase acquisition toolkit developed by Meltzer and Hovy (2011). Specifically, they define the context of a phrase as the concatenation of the n-grams to the immediate left and right of the phrase, and set the minimum and maximum lengths of an n-gram context to be 2 and 3, respectively, but they use point-wise mutual information weighted (Lin and Pantel 2001) phrase vectors, to compute cosine similarity as a measure of relatedness between two phrases. That is,

$$\cos(p,p') = \sum_{c \in C(p)} \frac{pmi(p,c) * pmi(p',c)}{\sqrt{\sum_{c' \in C(p)} pmi^2(p,c') * \sum_{c'' \in C(p')} pmi^2(p',c'')}}$$

where $C(p)$ represents the context vector of phrase p.

We re-implemented the above scoring function in NeSS on our data (37.5% of the preprocessed English Gigaword Fifth Edition). The results shown in Tables 1, 2 and 3 demonstrate that both our scoring functions are superior.

Word embedding: Recently, "word embedding" neural network approaches have been quite popular in building vector representations that capture semantic word relationships. To gauge their effectiveness, we compare with the single prototype word embeddings trained by Huang and Socher (2012). From the *MRŞ* comparisons in Table 3, it is clear that the H&S model is inadequate for the task of synonym extraction. We are unable to make comparisons to their multi-prototype model because they trained it only for 6162 most frequent words in the vocabulary. We also tried to train the word embedding model on 10% of the Gigaword corpus, but the task proved to be infeasible, since it would take about 2-years.

Concluding Remarks

We introduced a new unsupervised method for discovering phrasal near synonyms from large monolingual unannotated corpora and an evaluation method that generalizes precision@k for ranked lists of results based on multiple human judgments, weighing more heavily the top of the ranked list. Our methods in NeSS are based on combining elements of frequentist statistics, information theory, and scalable algorithms. NeSS significantly outperforms previous automated synonym finding methods on both the lexical and phrasal level, and outperforms thesaurus-based