# Large-Scale Learning of Word Relatedness with Constraints

Guy Halawi[§], Gideon Dror[†], Evgeniy Gabrilovich[‡], Yehuda Koren[†],
[§]Tel Aviv University, Tel Aviv, Israel; [†]Yahoo! Research, Haifa, Israel; [‡]Yahoo! Research, Santa Clara, CA, USA
ghalawi@gmail.com,{gideondr|gabr|yehuda}@yahoo-inc.com

## ABSTRACT

Prior work on computing semantic relatedness of words focused on representing their meaning *in isolation*, effectively disregarding inter-word affinities. We propose a large-scale data mining approach to *learning* word-word relatedness, where known *pairs* of related words impose *constraints* on the learning process. Our method, called CLEAR, is shown to significantly outperform previously published approaches. The proposed method is based on first principles, and is generic enough to exploit diverse types of text corpora, while having the flexibility to impose constraints on the derived word similarities. We also make publicly available a new labeled dataset for evaluating word relatedness algorithms, which we believe to be the largest such dataset to date.

## Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

## General Terms

Algorithms, Experimentation

## Keywords

Semantic similarity, word relatedness

## 1. INTRODUCTION

Computing semantic relatedness of words is an enabling technology for many natural language applications [5]. In document clustering, assessment of word relatedness can be used for formulating a better distance metric. In word sense disambiguation, the correct sense in which a word appears in a given text can be determined by computing the relatedness of the different candidate senses to the context. Consider also correction of typing errors or speech recognition errors, where a word is mistakenly replaced by another valid (i.e., not misspelled) word. Estimating word relatedness in context can help us determine the user most likely meant "Web site" rather than "Web sight".[1]

Conventional approaches to modeling semantic relatedness represent the meaning of individual words in a multidimensional space, and then compute the distance between the resultant word vectors. However, prior works focused on representation that is completely agnostic of the task of assessing relatedness, whereas the meanings of words is determined *in isolation*. Research in human judgment of similarity shows that it crucially depends on the context and frame of reference [25]. Quoting Tversky's example [25], "when subjects are asked to assess the similarity between the USA and the USSR, they usually assume that the relevant context is the set of countries and that the relevant frame of reference includes all political, geographical, and cultural features". Such context is completely lost when each word is considered in isolation from others.

Here we propose to *learn* a suitable word representation—in a latent factor space—with the particular task (of computing word relatedness) in mind. Specifically, we learn to represent word meanings by having observed word co-occurrences within sentences, as well as numerous *pairs of words* with different degrees of relatedness. This is done by incorporating word pairs of known relatedness as *constraints* on the learning process. We call our method CLEAR, which stands for Constrained LEArning of Relatedness.

Computing semantic relatedness using the newly learned representation yields statistically significant improvements on a range of datasets. Importantly, explicit modeling of word relatedness allows us to learn a signal that is *complementary* to that learned from large-scale world knowledge repositories, which were used in the previous state of the art method (TSA) [17]. Indeed, combining CLEAR with TSA yields significantly better results than either method alone. On the WS-353 dataset [10], which is the standard reference in the field, the combination improves the correlation with human judgments from $r = 0.8$ to 0.85.

The contributions of this paper are threefold. First, we introduce CLEAR, a novel approach to learning a word relatedness metric through a latent space embedding of the words, which directly incorporates the relatedness of training pairs of words used as constraints. Second, the results of using CLEAR for computing semantic relatedness are superior to the previous state of the art on three different datasets. Finally, we make publicly available a new dataset of 771 word pairs with human relatedness judgments, which was acquired as part of this research. To the best of our knowledge, this is the largest such dataset to date.


*KDD'12,* August 12–16, 2012, Beijing, China.


[1]Albeit one can argue that the last example can also be resolved by simple co-occurrence statistics (e.g., bi-grams), this would only be true for adjacent words. On the other hand, computing word relatedness to the rest of the text can easily handle long-term dependencies, which are very common in natural language (e.g., "Many people liked my *site* [or *sight* ?], which I have recently uploaded to the Web").

## 2. RELATED WORK

Prior work on semantic relatedness of words pursued two main directions, using purely statistical techniques or using repositories of human knowledge.

Early work on word relatedness formulated the notion of distributional similarity [6, 15], judging the relatedness of words by the similarity of contexts in which they occur. Later, Latent Semantic Analysis (LSA) [7] was proposed, which also leverages word co-occurrence information from a large unlabeled corpus of text. LSA is essentially a dimensionality reduction technique that identifies a number of most prominent dimensions in the data, which are assumed to correspond to "latent concepts". This is done by applying Singular Value Decomposition to the words-by-documents co-occurrence matrix. Latent Dirichlet Allocation (LDA) [3] can also be used for computing word relatedness by representing words as vectors of probabilities over each topic. Sun et al. [23] used LDA in a related application of text segmentation, using a Fisher kernel.

Lexical databases such as the WordNet electronic dictionary [8] or Roget's Thesaurus [19] encode relations between words such as synonymy, hypernymy, and meronymy. Quite a few metrics have been defined that compute word relatedness using various properties of the underlying graph structure of these resources (see [5] for an extensive survey). An obvious drawback of these approaches is that creation of lexical resources requires lexicographic expertise as well as considerable time and effort, and consequently such resources cover only a small fraction of the language lexicon. Specifically, such resources contain few proper names, neologisms, slang, and domain-specific technical terms. Indeed, Zesch et al. [29] showed that using a collaboratively constructed dictionary (Wiktionary) is superior to both WordNet and GermaNet (a German version of WordNet, [14]).

Dictionaries and thesauri mainly contain information about individual words but little world knowledge in general. Explicit Semantic Analysis (ESA) [11] was proposed to incorporate substantially large amounts of human knowledge — such as the entire Open Directory or Wikipedia — into word relatedness computation. Zesch and Gurevych [28] showed that it is possible to devise measures of relatedness using lexical resources that are competitive with those based on collaboratively constructed resources such as Wikipedia. Later, Temporal Semantic Analysis (TSA) [17] was proposed, which further improved the performance of ESA by extending it with a temporal dimension. This way, the computation of word relatedness was augmented with patterns of word occurrence over time (e.g., in an archive of a 100 years worth of New York Times articles).

Our approach effectively bridges the above two directions in a principled way. Similarly to purely statistical approaches, it learns word relatedness from word occurrence statistics in large text corpora. However, it *constrains* the learning process using word pairs that are *known* to be related, thus incorporating human knowledge. In Section 6 we compare CLEAR to the best performing statistical and knowledge-based approaches, and show that it outperforms all of them.

In this work, we used word pairs from WordNet, but in general they can come from a variety sources. For example, we can use lists of entities from Wikipedia (thus implicitly assuming the entities in the list are related) or Freebase, or lists automatically mined from the World Wide Web [13]. Domain-specific pairs of related words can be naturally incorporated in CLEAR.

In the present study we focused on computing semantic relatedness of individual words. However, in many cases a need arises to quantify semantic relatedness of longer fragments of text, for instance, in applications such as information retrieval or text clustering. The conventional approach to comparing fragments of text in information retrieval represents them as vectors of words, using the vector space model [20]. A major limitation of this model is that it assumes words to be independent, which is an oversimplification of the richness of natural language. Generalized vector space model [26] no longer makes this assumption, and explicitly manipulates a matrix $G$ of term dependencies. Under this model, the similarity of a pair of documents, $a$ and $b$, is computed as $sim_{GVSM}(a, b) = \mathbf{a}^T \cdot G \cdot \mathbf{b}$. Our method for estimating word relatedness can be immediately used for computing the matrix $G$, hence the generalized vector space model provides a principled way for incorporating CLEAR to refine existing text similarity models.

It should be noted that in this paper we deal with "semantic relatedness" rather than "semantic similarity" or "semantic distance", which are also often used in the literature. In their extensive survey of relatedness measures, Budanitsky and Hirst [5] argued that the notion of relatedness is more general than that of similarity, as the former subsumes many different kinds of specific relations, including meronymy, antonymy, functional association, and others. They further maintained that computational linguistics applications often require measures of relatedness rather than the more narrowly defined measures of similarity. For example, word sense disambiguation can use any *related* words from the context, and not merely *similar* words.

## 3. MODELING RELATEDNESS OF WORDS

We learn relatedness between words by embedding them in a low-dimensional space (e.g., a 100-D space), which strives to capture their semantic relations. Positions of words in the latent space reflect their meaning within sentences. This embedding is constructed while modeling the probability of observing a word given its adjacent context (e.g., the rest of its host sentence).

Before delving into the details of the method, we provide some necessary notational conventions. We observe words arranged within sequences. Each sequence annotates a pivot word. The sequence can be a short sentence containing the pivot word, or when the pivot word is placed within a long sentence, we limit the sequence to the words close to the pivot word (e.g., up to 5 words away). In order to distinguish words from sequences, we index words by $i, j$ and sequences by $s$.

Therefore, from the text corpus we extract sequences of at most $2K + 1$ consecutive words, which do not exceed single sentence boundaries. A typical value of $K$ would be in the range 2–8. Each sequence characterizes its middle pivot word denoted by $i_s$. The pivot word is removed from the sequence, that is $i_s \notin s$. Note that the extracted sequences can overlap because multiple pivot words are extracted from the same sentence. The sequences are arranged within a train set $\mathcal{S}$. After the sequences are identified, the word order within each sequence is no longer important, and we treat a sequence as a set of words.

A key component underlying our model is a mapping of words, and indirectly also of sequences, into a joint low dimensional space (of dimension $\ell$), which we refer to as the latent factor representation, as follows:

- Each word $i$ is mapped into a vector $q_i \in \mathbb{R}^\ell$.
- Each sequence $s$ is mapped into a vector $p_s \in \mathbb{R}^\ell$, defined by $p_s = \frac{1}{|s|} \sum_{i \in s} q_i$.

In addition, we introduce biases for words, such that the bias of word $i$ is denoted by $b_i$. These biases reflect the varying frequencies of words, independent of the surrounding sequence.

We denote by $r_{si}$ the affinity of word $i$ to sequence $s$, defined as

$$r_{si} = b_i + q_i^T p_s \quad (1)$$

Common words will generally have high bias values, reflecting their tendency to frequently co-occur with many other words (i.e., similar to many words in a distributional sense). However, words that are "strongly explained" by their context get bias values much lower than what is "merited" by their frequency. One such example is the word "versa", which almost always appears in the phrase "vice versa", hence its learned bias value is very low (actually negative, -1.25). On the other hand, the word "vice" can occur with many other words (e.g., "vice president"), hence its bias is much higher (+1.89).

Both the latent factor representation and the biases are parameters that we need to learn. Henceforth, we denote the model parameters by $\Theta$.

The latent space representation strives to capture the semantics of the words, such that affinities in the latent space reflect semantic relations. Typically, we use a latent space dimensionality of $\ell = 100-200$, which gives a good trade off between time and accuracy. We report the results of experimenting with other possible numbers of dimensions in Section 6.2.

The likelihood of observing word $i$ within sequence $s$ is modeled by the multinomial distribution

$$P(i|s;\Theta) = \frac{\exp(r_{si})}{\sum_j \exp(r_{sj})} \quad (2)$$

We seek model parameters ($\Theta$) in a way that maximizes the log-likelihood of the training set

$$L(\mathcal{S};\Theta) \stackrel{\text{def}}{=} \sum_{s\in\mathcal{S}} \log P(i_s|s;\Theta) \quad (3)$$

This modeling follows *multinomial logistic regression* [12], albeit with latent factors that make the function non-convex.

In the following, we often omit $\Theta$ from the probability definition, using the notation $P(i|s)$.

## 3.1 Optimization process

Learning proceeds by stochastic gradient ascent [4, 22]. Given a training sequence $s$ we update each parameter $\theta \in \Theta$ by

$$\Delta\theta = \eta \frac{\partial \log P(i_s|s)}{\partial\theta} = \eta\left(\frac{\partial r_{si_s}}{\partial\theta} - \sum_j P(j|s)\frac{\partial r_{sj}}{\partial\theta}\right) \quad (4)$$

where $\eta$ is the learning rate.

However, such a training scheme would be too slow to be practical, as each update rule requires summing over all words. Thus, we resort to sampling the weighted sum in (4), based on the importance sampling idea proposed by Bengio and Senécal [2].

With importance sampling we draw words according to a *proposal distribution*. In our case we assign each word a probability proportional to its empirical frequency in the train set, and denote this proposal distribution by $P(i|\mathcal{S})$. Consequently, words are sampled with replacement from $P(i|\mathcal{S})$ into a list $\mathcal{J}$. Thus, the expensive-to-compute $P(i|s)$ probabilities are approximated with the weighting scheme

$$w(i|s) = \frac{\exp(r_{si})/P(i|\mathcal{S})}{\sum_{j\in\mathcal{J}} \exp(r_{sj})/P(j|\mathcal{S})} \quad (5)$$

Consequently, the approximated gradient ascent step given a training sequence $s$ will be

$$\Delta\theta = \eta\left(\frac{\partial r_{si_s}}{\partial\theta} - \sum_{j\in\mathcal{J}} w(j|s)\frac{\partial r_{sj}}{\partial\theta}\right) \quad (6)$$

As mentioned in [2], it is desirable that the size of set $\mathcal{J}$ grows as the training process proceeds, because at later training phases more delicate parameter adjustments are needed. We employ a simple rule for controlling the sample size ($|\mathcal{J}|$) based on the fitness of current estimate. Given a training sequence $s$, we keep sampling words into $\mathcal{J}$ until the following condition is satisfied:

$$\sum_{j\in\mathcal{J}} P(j|s) > P(i_s|s) \Leftrightarrow \sum_{j\in\mathcal{J}} \exp(r_{sj}) > \exp(r_{si_s}) \quad (7)$$

The adaptive sampling automatically lets the sample size grow when parameters are nearing final values and the correct word is getting a relatively high probability. We impose a minimal size of 50 on the sample size. For efficiency we also limit the maximal sample size to 1500.

We now provide a few finer details on the stochastic gradient ascent algorithm. We cycle through random permutations of the training examples (pivot words) in order to achieve convergence. In our implementation we ran the process for 5 sweeps over the training data, though not much improvement was observed beyond 3 sweeps. Weight decay was not found to be helpful. When training on the $t$-th example, we set the learning rate to $\eta_t = \frac{\eta_0}{t+\tau}$. This learning rate schedule satisfies the Robbins-Monro conditions [18], $\sum \eta_t = \infty$ and $\sum \eta_t^2 < \infty$. Specifically, in our implementation we set $\eta_0 = 10^6$ and $\tau = 10^7$. All model parameters were randomly initialized by drawing each from $\mathcal{N}(0, 0.01)$.

## 3.2 Incorporating similarity constraints

Frequently we have external knowledge relating pairs of words. For example, WordNet [8] can supply us with numerous pairs of related words, including synonyms, hypernyms/hyponyms, and meronyms/holonyms. Other sources of such information can be domain-specific, expert-driven, or Web-based (see Section 2 for additional examples). We incorporate this knowledge into our approach through pairwise regularization terms. It should be emphasized that it is this step that allows CLEAR to incorporate world knowledge. This way, we no longer represent words in isolation, but rather induce a representation where related words are located close to one another.

Specifically, let us denote by $\mathcal{P}$ the set of word pairs known to be related. We encourage closeness between such pairs by enhancing our optimization goal (3) to be

$$L(\mathcal{S};\Theta) \stackrel{\text{def}}{=} \sum_{s\in\mathcal{S}} \log P(i_s|s,\Theta) - \lambda \sum_{(i_1,i_2)\in\mathcal{P}} \|q_{i_1} - q_{i_2}\|^2 \quad (8)$$

We used the value 0.01 for the constant $\lambda$, which controls the extent of pulling known related words towards each other. We report the results of experimenting with other possible values of this parameter in Section 6.2.

Note that we only use the fact that particular word pairs are related, but do not use the actual degree of their relatedness. In practice, automatic acquisition of related word pairs is fairly easy from electronic dictionaries, thesauri, or world knowledge repositories such as Wikipedia. Consider, for example, the list of mathematical functions in Wikipedia (`http://en.wikipedia.org/wiki/List_of_mathematical_functions`). Pairing all the entries in this list can easily create many pairs of related concepts, yet requiring each such pair to be accompanied with a human judgment of relatedness would adversely affect large-scale data acquisition.

In our future work, we plan to experiment with annotating constraints with automatic assessment of relatedness.

Prior work on semantic relatedness represented individual words in isolation, effectively disregarding known affinities among the words in the language. Thus, we view the easiness of incorporating such information as a core component of our method, which augments statistical processing with human world knowledge in a principled way.

## 3.3 Deriving word relatedness

The latent factors yield a compact representation of the words that allows their direct comparison. Given two words $i$ and $j$, we estimate their relatedness by comparing their respective factor vectors $q_i$ and $q_j$. In this work, the word-word relatedness is defined by the cosine function:

$$\mathrm{rel}_{ij} = \frac{q_i^T q_j}{\|q_i\|\|q_j\|} \quad (9)$$

# 4. DATA CREATION AND PROCESSING

In this section we discuss the data sources used in this work, including the corpora used for model training, word relatedness constraints, and test datasets. The next section motivates the choice of evaluation metric and the baselines we used. Then, we report the experimental results in Section 6.

## 4.1 Text corpora

As explained in Section 3, our method learns word relatedness from text corpora. In this work, we used three text corpora from very different domains. With the intent of modeling the word relatedness judgments of a layman, we refrained from using more formal sources such as Reuters' RCV1 or Wikipedia[2], but focused on corpora that reflect informal language use:

• **Yahoo! Answers**: This dataset is available through the Yahoo! Webscope program[3]. The dataset contains questions and answers posted to the Yahoo! Answers service prior to October 2007. The corpus contains a wide spectrum of questions, such as factual, advice seeking, and social, on topics ranging from Arts & Entertainment to Sports, Travel, Health, or Politics. We used the first half of the corpus comprising of 2,215,396 questions. For the purpose of this work, the concatenation of all the text related to a single question, namely, the question title, body, and the best answer, was considered as a single document.

• **UKWaC** [1] is a corpus built by crawling of the .uk Web domain. It contains more than a billion words, and is considered one of the largest Web resources for British English. It is part-of-speech tagged and lemmatized. We used the first file of the UKWaC corpus distribution version 1.0, which contains 110,165 documents.

• **Subtitles**: We extracted English subtitles for 9,950 random movies and television series from the OpenSubtitle API[4].

## 4.2 Processing

Each corpus was separately processed using the following preprocessing steps:

1. Parsing, tokenization and stopword removal.

2. Lemmatization of tokens, which amounts to reducing each token to its morphological stem. For example, this step identifies as equivalent different tense forms of the same verb (e.g., 'climbed' and 'climb'), including irregular verbs (e.g., 'go' and 'went'). For this task we used the LemmaGen tool[5].

3. Removal of rare tokens: tokens occurring fewer than 30 times were removed from the dictionary as well as from the text. For the Subtitles dataset, which is considerably smaller, we reduced the dictionary size by simply taking the 40,000 most frequent tokens.

4. The corpus was split into sentences, and sentences with fewer than 4 tokens were discarded. We used the SharpNLP Project[6] for sentence boundary detection.

| Corpus | # Sentences | Dictionary size | #Tokens |
|---|---|---|---|
| Y! Answers | 4,778,784 | 41,102 | 41,170,886 |
| UKWaC | 3,499,361 | 42,717 | 34,382,199 |
| Subtitles | 695,482 | 40,000 | 4,927,282 |

**Table 1: Number of sentences, dictionary size and total number of word tokens comprising the final corpora**

Table 1 details the characteristics of the processed datasets. Additionally, to give the reader a glimpse into the different corpora we used, Figure 1 pictorially depicts the most frequent tokens in each corpus using word clouds.

## 4.3 WordNet synonyms as similarity constraints

As explained in Section 3.2, our method can be constrained with a collection of word pairs, which are known to be related. In Section 2 we discussed a variety of suitable resources that can provide lists of related word pairs. In this work, we used the WordNet electronic dictionary [8], and the constraints were based on the synonymy relation.

The basic unit of information in WordNet is a synset, which is a set of synonyms that share the same word sense. For each word in our datasets, we queried WordNet for synonyms. The most common synset was picked if the word was ambiguous. Overall, 10,148 tokens were found to be related through the synonymy relations. These provided a total of 21,929 synonym pairs, which were used by method as relatedness constraints.

## 4.4 Evaluation datasets

The standard practice of evaluating word relatedness models is to compute the correlation between their predictions and human judgments [5]. In this study, we used three evaluation datasets.

The WordSimilarity-353 (WS-353) is the standard reference dataset in the field, which contains 353 word pairs. Each word pair was judged by 13–16 human annotators. This dataset, which was so far the largest publicly available collection of this kind, was extensively used for word relatedness evaluation [29, 11, 28, 27, 17].

In addition, we used the dataset constructed by Radinsky et al. [17], comprising of 287 word pairs. In this dataset, word relatedness was evaluated by Amazon's Mechanical Turk workers, with an average of 23 worker ratings for each word pair. We call this dataset MTURK-287.

To enhance the statistical strength of our findings, we also constructed a third, larger dataset, which we make publicly available at `http://www2.mta.ac.il/~gideon/datasets/`. We discuss the process of its construction in the next section.

## 4.5 Constructing a new evaluation dataset

Word pairs were constructed to get a comprehensive sampling of both related and unrelated words, and we specifically aimed at

[2]Recent research shows that Wikipedia—despite being collaboratively authored by numerous contributors around the world—can hardly be viewed as easily readable text [24].

[3]`http://webscope.sandbox.yahoo.com/`

[4]`http://trac.opensubtitles.org/projects/opensubtitles`

[5]http://lemmatise.ijs.si/

[6]`http://sharpnlp.codeplex.com/`

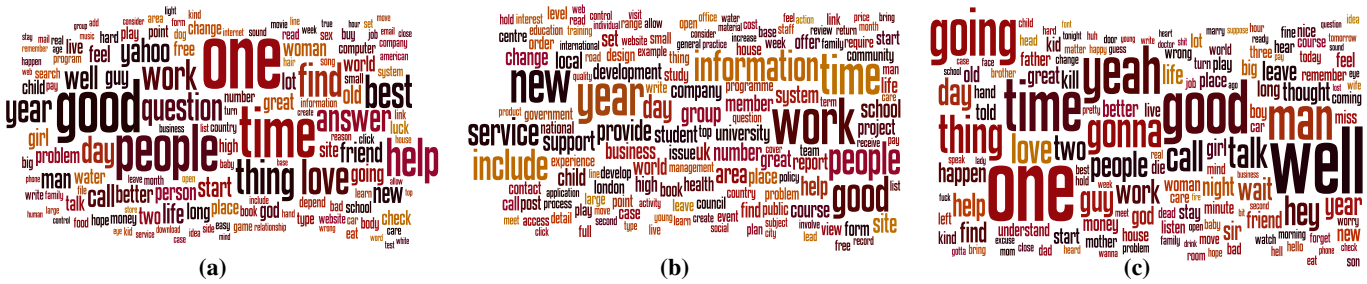


**Figure 1: Word clouds depicting most frequent tokens in (a) Yahoo! Answers dataset; (b) UKWaC dataset; (c) Subtitles dataset. Displayed token size is proportional to token frequency. Pictures produced by `wordle.net`.**

various types of relatedness. To this end we considered WordNet [8] as a graph linking nouns. For two nouns $i$ and $j$, the following relations (graph edges) are possible:

- $i$ and $j$ are synonyms (synonymy relation).
- $i$ is a part of $j$ (meronymy relation), for example 'leg' and 'table'.
- $i$ is a type of $j$ (holonymy relation), for example 'table' and 'furniture'.

*Construction of word pairs.*

From this graph we uniformly sampled noun pairs with (undirected) graph distances between 1 and 4. In order to ensure quality of collected human evaluations, we eliminated rare words. To this end, we used the Yahoo! Web search engine to count the number of search results when a word is used as a query. We discarded all pairs containing a word with fewer than 40 million search results.

*Human estimation of word relatedness.*

Word relatedness was evaluated by Amazon's Mechanical Turk workers, with an average of 20 ratings for each word pair, where each judgment task consisted of a batch of 50 word pairs. Ratings were collected on a 1–5 scale, where 5 stands for "highly related" and 1 stands for "not related". In order to discard poor-quality work, each batch contained 10 trap word pairs with known extreme relatedness values (WordNet synonyms or extremely unrelated words as indicated by WS-353), serving as binary indicators. A batch that failed on more than one of the binary indicators was discarded. This guarantees an over 98% probability for detecting random workers (e.g., bots). The relatedness value of each word-pair was taken as the mean score given by the workers. This dataset has 771 word-pairs, and is accordingly named MTURK-771.

Figure 2 depicts the word-relatedness scores, sorted by their mean value. The error bars represent a one standard deviation confidence interval of each value. The horizontal axis represents the rank of each word-pair when sorted by relatedness in ascending order, such that the least related word-pair is ranked 1 and the most related word-pair is ranked 771. Importantly, our dataset includes the full range of word relatedness, with a clear signal, demonstrated by the relatively small error bars. In particular we observe that word-pairs which are with extremal value of relatedness have significantly lower variance in human judgements, as evident from the shorter error bars on both extremes of the relatedness range.

*Consistency assessment.*

To verify the agreement between raters, following Snow et al. [21], we randomly split the raters into two groups, each including 10 Mechanical Turk workers. We then averaged the numeric judgements for each word pair among the raters in each of the two sets, thus

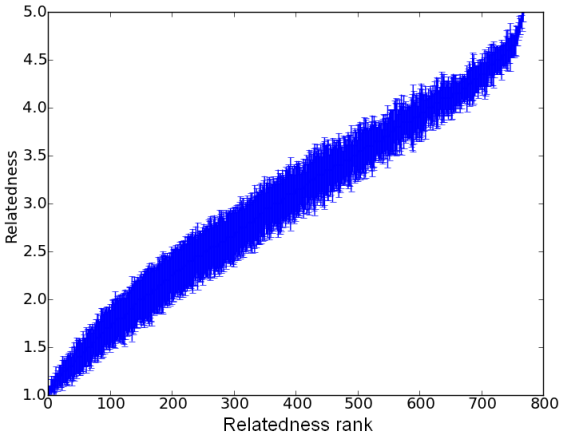


**Figure 2: Word-pair relatedness (with error bars), sorted by value**

yielding a (771 element long) vector of average judgments for each set. Finally, we computed the correlation between the vectors of the two sets. We repeated this process 1000 times, and over these 1000 random splits the mean correlation between the two sets of raters was 0.8957, with extremely small variance, attesting to the quality of the collected data.

## 5. EVALUATION SETUP

We now discuss the evaluation metric we employed, as well as the procedure for assessing the significance of the results observed.

Following prior literature [11, 17], we compute Spearman's rank-order correlation coefficient between the algorithm's predictions and the average human judgments on a list of word pairs. Being non-parametric, Spearman's correlation coefficient is considered to be much more robust than Pearson's linear correlation. We report Spearman's correlation separately for each test dataset, because the datasets have been collected by different methods.

### 5.1 Significance testing

We now discuss our methodology for testing the significance of the obtained results. Given two algorithms for computing word relatedness, let $r_1$ and $r_2$ be the values of their (Spearman rank) correlation with human judgments, obtained on a set of $n$ word pairs. Let $z_i$ be their Fisher transformed values [9]:

$$z_i = \sqrt{\frac{n-3}{1.06}} \frac{1}{2} \ln \frac{1+r_i}{1-r_i} \tag{10}$$

The $z_i$ scores approximately follow a standard normal distribution, and therefore it is straightforward to determine whether the difference $z_1 - z_2$ is significantly different from zero. Specifically, to show the superiority of CLEAR we use a one-sided t-test.

## 5.2 Baseline methods

In what follows, we compare CLEAR to the two methods that showed best reported results to date, namely, ESA [11] and TSA [17] (see Section 2). We also compare CLEAR to two additional strong baselines, namely, distributional similarity and LDA, which share certain aspects with CLEAR. These two baselines are described below.

### *Distributional similarity (DS).*

The notion of distributional similarity of words was formulated by Lee [15] and Dagan et al. [6].

Let $n$ be the number of unique words in the dataset. For each word we maintain an $n$-dimensional vector, which reflects the frequency with which it co-occurs with every other word in the dataset. We consider two words to co-occur if they are collocated within the same sentence, no more than $K$ tokens apart. We used $K = 3$, which yielded the best results. Word frequency was normalized by dividing it by the overall word count (standard IDF normalization was also tried, but yielded inferior results). Formally, let $n_{ij}$ be the number of co-occurrences of words $i$ and $j$, and let $n_j$ be the overall frequency of word $j$. Then, word $i$ is represented via $v_i \in \mathbb{R}^n$, where $v_{ij} = n_{ij}/n_j$. Finally, the distributional similarity (DS) between words $i$ and $j$ is defined by $\cos(i, j)$.

Distributional similarity shares with CLEAR the principle of defining the word semantics using the words that frequently co-occur with it. However, unlike CLEAR, it is not based on an optimization process, and does not use a latent factor representation for compactly capturing the words semantics.

### *Latent Dirichlet allocation (LDA).*

LDA [3] is a generative model that explains word occurrences in documents through latent topics underlying the document and the word distributions. We used the open source LDA implementation called Mallet [16]. We report results using 200 topics (best performing setting). For each word $i$, we compute its probability distribution over topics, denoted by $p_i$. Semantic relatedness between words $i$ and $j$ is then computed as the Jensen-Shannon divergence between their respective probability distributions.

CLEAR shares resemblance with LDA in terms of modeling the words through latent factor vectors. However, CLEAR differs in modeling short word sequences, or sentences, whereas LDA requires longer documents for good performance. Furthermore, CLEAR incorporates external-supplied similarity relations, which are given as constraints.

## 5.3 New evaluation scenario: finding word synonyms

We also used a new evaluation scenario, where we employed CLEAR to identify word synonyms (which are obviously related in meaning), and used WordNet to validate our predictions. When doing so, we did not use WordNet synonyms as learning constraints (cf. Section 4.3). Note that WordNet synonyms comprise a larger dataset than any of the three word relatedness datasets (cf. Sections 4.4 and 4.5); however, this particular dataset lacks negative or weak positive examples.

# 6. EXPERIMENTAL RESULTS

We trained CLEAR on the three text corpora described in Section 4.1, and a combination thereof. The method was implemented in C#, and was run on a machine with Intel Dual Core E840 3GHz processor and 2GB RAM. Typical training time for 5 sweeps over the Yahoo! Answers corpus was 5 hours. To give the reader a flavor of the results, we show in Table 2 examples of related words found by our method.

In what follows, we first report the performance of our method on the three datasets described in Sections 4.4 and 4.5, and then on the task of identifying word synonyms described in Section 5.3.

## 6.1 Quantifying word relatedness with CLEAR

We compare the performance of CLEAR with that of prior methods (discussed in Sections 2 and 5.2). The choices of parameter values used for the experiments in this subsection, as well as the results of experimentation with other possible parameter settings, are discussed in Section 6.2.

Several variants of CLEAR are reported, which differ by the training corpus. First, we trained CLEAR separately on Y!Answers, UKWaC, and Subtitles. As can be seen in Table 3, models trained on Y!Answers deliver the best performance on all datasets, followed closely by UKWaC-trained models. Training on Subtitles yields significantly poorer results, which might be explained by the significantly smaller size of this corpus.

Clearly, there is no need to limit the training to only one of the corpora. Therefore, we also present the results obtained with the two more informative training corpora (Yahoo! Answers and UKWaC) combined, as well as the results due to the combination of all three corpora. When combining these corpora, we let the training process run on each, and then employ the average of the cosine similarities computed on each corpus separately. As we can see in Table 3, the performance of CLEAR is obviously improving when it learns from diverse corpora. Interestingly, even the smaller Subtitles corpus, which delivered poorer results on its own, adds a non-negligible value when integrated with the others. This probably reflects on its unique nature of representing spoken, rather than written, language.

In the remainder of this subsection (6.1), we report CLEAR results with training on all the three training corpora.

| Method | WS-353 | MTURK-287 | MTURK-771 |
|---|---|---|---|
| CLEAR on Y!Answers | 0.771 | 0.697 | 0.690 |
| CLEAR on UKWaC | 0.748 | 0.653 | 0.673 |
| CLEAR on Subtitles | 0.635 | 0.636 | 0.610 |
| CLEAR on Y!Answers+ UKWaC | 0.795 | 0.712 | 0.716 |
| CLEAR on Y!Answers+ UKWaC+Subtitles | **0.810** | **0.737** | **0.727** |

**Table 3: Performance of CLEAR on different training corpora and their combinations.**

Table 4 compares CLEAR with previously published methods. The results of prior methods (ESA [11] and TSA [17]) were achieved using their original code through their respective authors. As we can readily see, CLEAR outperforms these state of the art methods with a notable margin, especially on the two MTURK test sets. In almost all the cases, the difference between CLEAR and each of the other methods is statistically significant. Table 4 shows the statistical significance of the difference between the performance of each method and that of CLEAR; we use the symbols § and † to represent significance levels of $\alpha = 0.0001$ and $\alpha = 0.01$, respectively.

| coke | mile | religion | sheep | university | webcam |
|---|---|---|---|---|---|
| pepsi (0.813) | mi (0.826) | relgion (0.779) | ox (0.648) | suny (0.809) | cam (0.636) |
| cola (0.567) | kilometer (0.690) | abrahamic (0.771) | ewe (0.620) | nyu (0.806) | logitech (0.581) |
| sprite (0.491) | nautical (0.607) | spirituality (0.717) | goat (0.614) | devry (0.804) | messenge (0.519) |
| coca (0.477) | kilometre (0.589) | religon (0.715) | herd (0.612) | polytechnic (0.801) | messenger (0.484) |
| pepsico (0.471) | furlong (0.563) | secularism (0.674) | cattle (0.596) | univeristy (0.795) | camera (0.474) |

**Table 2: Word relatedness examples: top-5 lists of most related for six sample words, obtained by running our method on the Yahoo! Answers corpus; cosine similarity values are reported in parentheses. (Spelling errors are in original data.)**

We remark that unlike ESA and TSA, CLEAR follows more general principles, as it does not require elaborate knowledge resource or text corpora spanning long periods of time. In fact, CLEAR can be trained on virtually any text corpus. Hence, we would expect the CLEAR advantage to grow even further when it is trained on more diverse corpora.

We further analyze the performance of CLEAR by comparing it to two other methods, which isolate two of its founding principles. First, we compare it to distributional similarity (DS), which models words based on their contextual neighborhood. The best results for this method were obtained when trained on the three corpora combined. Second, we compare CLEAR to LDA, which demonstrates the importance of mapping words to a latent factor space, thus capturing their semantic relations. Again, combining the three training text corpora improved LDA accuracy. We observe that both DS and LDA cannot match the performance of CLEAR. Yet, LDA delivers results much stronger than DS, indicating the greater importance of latent factor modeling.

| Method | WS-353 | MTURK-287 | MTURK-771 |
|---|---|---|---|
| CLEAR | **0.810** | **0.737** | **0.727** |
| TSA | 0.8 | $0.61^{\S}$ | $0.606^{\S}$ |
| ESA | $0.75^{\S}$ | $0.607^{\S}$ | $0.603^{\S}$ |
| LDA | $0.736^{\S}$ | $0.677^{\dagger}$ | $0.619^{\S}$ |
| DS | $0.61^{\S}$ | $0.625^{\S}$ | $0.578^{\S}$ |

**Table 4: Comparing the performance of CLEAR with competing methods. The symbols § and † represent statistically significant differences (vs. CLEAR) at the levels of 0.0001 and 0.01, respectively.**

We also evaluated whether other methods can add value on top of CLEAR. To this end, we formed linear combinations of CLEAR values with those produced by other methods. The linear combination coefficients were learned by linear regression on the human judgments. When evaluating on each one of the datasets, we used the other two datasets for training the regressor.

The results are reported in Table 5. We can see that the accuracy is further improved by combining CLEAR with other methods, implying that CLEAR learns signals that are complementary to the other methods. We also tried to combine the previous methods, without involving CLEAR (the bottom two rows of Table 5). We found that the linear combination of ESA and TSA could not improve performance. This is possible because linear regression is suboptimal when optimizing Spearman's correlation. We also tried a simple average, but it did not improve the performance either. Even though some smarter combinations of the two methods may be successful, we suspect that the signals captured by these two methods are not sufficiently different to justify their combination. Indeed, TSA extends the ESA model with temporal analysis, therefore, the results of the two methods are correlated. As for DS and LDA, their linear combination does improve the performance; this is not surprising given their very different nature. However, their combination is still inferior to the standalone CLEAR.

| Method | WS-353 | MTURK-287 | MTURK-771 |
|---|---|---|---|
| CLEAR + ESA | 0.826 | 0.746 | 0.732 |
| CLEAR + TSA | **0.850** | **0.751** | **0.742** |
| CLEAR + DS | 0.809 | 0.738 | 0.729 |
| CLEAR + LDA | 0.818 | 0.749 | 0.731 |
| ESA+TSA | 0.772 | 0.601 | 0.604 |
| DS + LDA | 0.757 | 0.694 | 0.638 |

**Table 5: Performance of pairs of methods combined**

To further compare CLEAR with TSA, we examined the ranking of word pairs by their relatedness according to each algorithm. To this end, for each dataset we ranked all the words by their relatedness according to CLEAR, TSA, and humans. We denote the resultant rankings as $\mathcal{R}_{CLEAR}$, $\mathcal{R}_{TSA}$, and $\mathcal{R}_{human}$, respectively. We found $\mathcal{R}_{CLEAR}$ to be consistently closer to $\mathcal{R}_{human}$ than $\mathcal{R}_{TSA}$.

Indeed, on WS-353, the average improvement in rank from $\mathcal{R}_{TSA}$ to $\mathcal{R}_{CLEAR}$ was 5.8 positions (median 3.5). For MTURK-287, the average improvement was 10.2 positions (median 3). Finally, for MTURK-771, the average improvement was 29.8 positions (median 19).

## 6.2 Parameter setting study

We now explore the dependency of CLEAR performance on the following parameters:

$(a)$ The parameter $K$ controls the context size, i.e., the maximum distance between the pivot word and other words in its context (Section 3).

$(b)$ The parameter $\lambda$ controls the extent of using the similarity constraints (Section 3.2).

$(c)$ The parameter $\ell$ represents the dimensionality of the latent factor space (Section 3).

The results reported here were obtained by training on the Yahoo! Answers corpus (the results using the other two training corpora exhibited very similar patterns).

Table 6 analyzes the effect of different context sizes. We see a mild variation of performance as $K$ changes, with the best results around $K = 6$. Hence we set $K = 6$ throughout the other experiments. We also observe that the results on the WS-353 test set are consistently higher than those on the two MTURK datasets, which remains true through all our evaluations.

Table 7 studies the effect of the similarity constraints (see Section 3.2). For the WS-353 and MTURK-771 datasets, optimal performances is obtained when similarity constraints are taken with the weight of $\lambda = 0.01$; however, further emphasis of these constraints with $\lambda = 0.1$ becomes counter-productive. As for the smaller MTURK-287 test set, even $\lambda = 0.01$ was too large with a negative impact on performance. Therefore, we set $\lambda = 0.01$ for all our reported results.

Finally, we report in Table 8 the effect of the dimensionality of

| Context size | WS-353 | MTURK-287 | MTURK-771 |
|---|---|---|---|
| $K = 2$ | 0.744 | 0.664 | 0.682 |
| $K = 4$ | 0.756 | 0.670 | **0.700** |
| $K = 6$ | **0.767** | **0.687** | 0.695 |
| $K = 7$ | 0.761 | 0.686 | 0.688 |
| $K = 8$ | 0.747 | 0.678 | 0.686 |

**Table 6: Impact of context size: CLEAR performance for various values of $K$ (measured by Spearman correlation; higher is better). Here $\lambda = 0.01$ and $\ell = 100$.**

| $\lambda$ | WS-353 | MTURK-287 | MTURK-771 |
|---|---|---|---|
| 0 | 0.744 | **0.692** | 0.674 |
| 0.01 | **0.767** | 0.687 | **0.695** |
| 0.1 | 0.742 | 0.684 | 0.666 |

**Table 7: Impact of similarity constraints: CLEAR performance for various values of $\lambda$. Here $K = 6$ and $\ell = 100$.**

the latent factor space. We see steady accuracy gains until reaching $\ell = 100 - 150$, demonstrating the added expressive power of the increased dimensionality. However, as dimensionality further increases towards 500, the sparser latent space does not translate into a measurably better performance. Therefore, we set $\ell = 150$ for the remainder of the experiments.

| Latent dim. | WS-353 | MTURK-287 | MTURK-771 |
|---|---|---|---|
| $\ell = 20$ | 0.681 | 0.662 | 0.610 |
| $\ell = 50$ | 0.722 | 0.675 | 0.670 |
| $\ell = 100$ | 0.767 | 0.687 | 0.695 |
| $\ell = 150$ | **0.771** | **0.697** | 0.690 |
| $\ell = 200$ | 0.759 | 0.668 | 0.680 |
| $\ell = 500$ | 0.758 | 0.665 | **0.700** |

**Table 8: Impact of dimensionality: CLEAR performance for various values of $\ell$. Here $\lambda = 0.01$ and $K = 6$.**

## 6.3 Identifying word synonyms

Test sets based on human judgments are expensive to collect and hence small in size. Therefore, we also evaluated the performance of CLEAR on an additional task, namely, identifying word synonyms. We used WordNet as a gold-standard repository of synonyms. In this experiment, we used a version of our algorithm trained without similarity constraints (i.e., we set $\lambda = 0$), so that no explicit knowledge on the synonyms leaks into the training phase (cf. Section 4.3).

First, we computed similarities between 21,929 WordNet synonym pairs, henceforth called *SynPairs*. Second, for each word in *SynPairs*, we formed 100 test instances by pairing it with 100 random words; we refer to this collection as *RndPairs*. Naturally, one would expect the similarities computed on *SynPairs* (i.e., between true synonyms) to be significantly higher than those computed on *RndPairs* (i.e., random pairs of words). We repeated the same experiment with LDA. As explained in Section 5.2, we used the Jensen-Shannon distance for comparing the LDA word vectors, which gave there better results than cosine similarity.

Figure 3 shows the performance of CLEAR and LDA on identifying word synonyms. For both methods, the results visibly indicate that the distribution of pairwise similarity values for synonyms (*SynPairs*) is very different from that for random word pairs (*RndPairs*). We used the Jensen-Shannon divergence to compute the distance between the two distributions. For CLEAR, the JS divergence between the two distributions is 0.275, whereas for LDA it is 0.178. This implies better discriminative power of CLEAR compared to LDA.

Note also that LDA topic vectors tend to be sparse (sparseness factor of 97.87%), so both *SynPairs* and *RndPairs* distributions are peaking near 1, indicating no overlap between the topic vectors of the respective words. In fact, we observed that the sparseness of LDA topic vectors, which is helpful for unsupervised exploratory analysis, harms its predictive abilities. As many topics are correlated, a sparse representation of words over topics masks important signals. At CLEAR, the sparseness factor is virtually zero, thus reducing the issues raised by correlative factors.

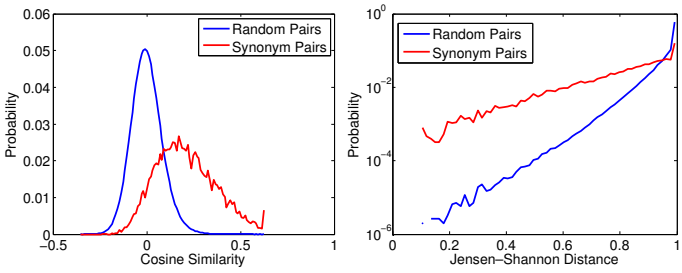


**Figure 3: The distributions of pairwise similarities (or, distances) for both WordNet synonyms and for random pairs.**

## 6.4 The effect of the consistency of human judgments

Interestingly, we found a dependency between the accuracy of CLEAR and the consistency of human judgments. For the MTURK-771 dataset, we ranked all word pairs by the CLEAR relatedness and by the human-judged relatedness. Then, for each word pair we calculated the difference in its ranks according to CLEAR and according to humans, denoted as $\mathcal{D}$. We also calculated the variance between human raters for each word pair, denoted by $\mathcal{V}$.

We found $\mathcal{D}$ and $\mathcal{V}$ to be significantly correlated, with Pearson's coefficient $r = 0.18$. In other words, when raters tend to disagree about a particular word pair, we find CLEAR to systematically err more about the relatedness of these words. This result is highly significant statistically and cannot be explained by chance, with $p$-value $p = 1.96 \cdot 10^{-7}$.

# 7. CONCLUDING REMARKS

This paper introduced CLEAR, a method for estimating relatedness between natural language words. Prior works represented the meaning of words in isolation, even though cognitive research proves humans represent word meanings in context. CLEAR circumvents this limitation by learning word semantics from actual contexts in which the word occurs. CLEAR can be trained on any text corpus, without limiting the nature of the analyzed text, and hence can learn from virtually all kinds of textual corpora. Indeed, we have shown that training the method on texts from very different domains and styles improves its accuracy.

The main principle behind CLEAR is learning to embed words in a low-dimensional latent space, thus explaining the composition of short word sequences. Furthermore, CLEAR can leverage existing knowledge on word relatedness, which is directly encoded using similarity constraints.

We believe that the following ingredients contribute to the success of CLEAR:

• Modeling informal language, such as that used in the Yahoo! Answers and the movie subtitles corpora, was found beneficial to modeling human judgments of word relatedness.

• An ability to anchor the model with existing lists of related words boosts the accuracy of the method.

• Analysis of short contexts, rather than full documents or paragraphs, assists in better identification of related words.

• Embedding the words in a latent semantic space is helpful for identifying related words, and also for treating rare words correctly based on models learned for their more frequent relatives.

• A model learned through global optimization captures large scale signals, and thus shows higher robustness against local artifacts.

In a comprehensive set of empirical tests, CLEAR was shown to outperform prior works, as it achieves the best known results on a host of highly competitive benchmarks. We also showed that the signal learned by CLEAR is complementary to that learned by other state of the art methods. Indeed, combining CLEAR with TSA yields even better results. Interestingly, we found a notable correlation between the agreement of human raters and the performance of CLEAR—the method shines on word pairs that are unambiguous for humans, while tends to err more on word pairs that are also difficult for humans to agree on.

In the future work, we plan to augment our method by incorporating new types of constraints. For instance, in addition to synonyms, we can use other word relations such as hypernyms, hyponyms, meronyms, and holonyms, to name but a few. These relations imply different degrees of word relatedness, and quantifying the degree of relatedness in each constraint can further refine our method.

As an additional contribution of this research, we collected a new human-judged dataset for evaluating word relatedness algorithms. We believe this dataset, called MTURK-771, is the largest such test collection to date. We make it publicly available at `http://www2.mta.ac.il/~gideon/datasets/`.

## Acknowledgments

We thank Kira Radinsky for running experiments on the TSA method. We also thank Greg Druck, Bo Pang, and Sujith Ravi for pointers to relevant literature.

# 8. REFERENCES

[1] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 2009.

[2] Y. Bengio and J.-S. Senécal. Quick training of probabilistic neural nets by sampling. In *Proc. 9th International Workshop on Artificial Intelligence and Statistics (AISTATS'03)*, 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] L. Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, LNAI 3176, pages 146–168. Springer Verlag, 2004.

[5] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

[6] I. Dagan, L. Lee, and F. C. N. Pereira. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1–3):43–69, 1999.

[7] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[8] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[9] E. Fieller, H. Hartley, and E. Pearson. Tests for rank correlation coefficients. *Biometrika*, 44:470–Ű481, 1957.

[10] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM TOIS*, 20(1):116–131, January 2002.

[11] E. Gabrilovich and S. Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34:443–498, 2009.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2009.

[13] R. Hoffmann, C. Zhang, and D. S. Weld. Learning 5000 relational extractors. In *ACL*, pages 286–295, 2010.

[14] C. Kunze. Computerlinguistik und sprachtechnologie. In *Lexikalisch-semantische Wortnetze*, pages 423–431. Spektrum Akademischer Verlag, 2004.

[15] L. Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 25–32, 1999.

[16] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[17] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *WWW*, 2011.

[18] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Math. Statistics*, 22:400–407, 1951.

[19] P. Roget. *Roget's Thesaurus of English Words and Phrases*. Longman Group Ltd., 1852.

[20] G. Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[21] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast — but is it good? Evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

[22] J. C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., 2003.

[23] Q. Sun, R. Li, D. Luo, and X. Wu. Text segmentation with LDA-based fisher kernel. In *ACL-HLT Short Papers*, pages 269–272, 2008.

[24] C. Tan, E. Gabrilovich, and B. Pang. To each his own: Personalized content selection based on text comprehensibility. In *WSDM*, 2012.

[25] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[26] S. K. M. Wong, W. Ziarko, and P. C. N. Wong. Generalized vector spaces model in information retrieval. In *SIGIR*, 1985.

[27] E. Yeh, D. Ramage, C. D. Manning, E. Agirre, and A. Soroa. Wikiwalk: Random walks on wikipedia for semantic relatedness. In *2009 TextGraphs-4 Workshop*, 2009.

[28] T. Zesch and I. Gurevych. Wisdom of crowds versus wisdom of linguists? measuring the semantic relatedness of words. *Natural Language Engineering*, 16(1):25–59, 2010.

[29] T. Zesch, C. Mueller, and I. Gurevych. Using Wiktionary for computing semantic relatedness. In *AAAI*, pages 861–866, 2008.