Improving ESA with Document Similarity

Tamara Polajnar[1], Nitish Aggarwal[1], Kartik Asooja[2], and Paul Buitelaar[1]

[1] Unit for Natural Language Processing
Digital Enterprise Research Institute
National University of Ireland
Galway, Ireland
firstname.lastname@deri.org
[2] Ontology Engineering Group
Universidad Politecnica de Madrid
Madrid, Spain
asooja@gmail.com

Abstract. Explicit semantic analysis (ESA) is a technique for computing semantic relatedness between natural language texts. It is a document-based distributional model similar to latent semantic analysis (LSA), which is often built on the Wikipedia database when it is required for general English usage. Unlike LSA, however, ESA does not use dimensionality reduction, and therefore it is sometimes unable to account for similarity between words that do not co-occur with same concepts, even if their concepts themselves cover similar subjects. In the Wikipedia implementation ESA concepts are Wikipedia articles, and the Wikilinks between the articles are used to overcome the concept-similarity problem. In this paper, we provide two general solutions for integration of concept-concept similarities into the ESA model, ones that do not rely on a particular corpus structure and do not alter the explicit concept-mapping properties that distinguish ESA from models like LSA and latent Dirichlet allocation (LDA).

1 Introduction

Explicit semantic analysis (ESA) computes the relatedness between two words, or bags-of-words, based on concepts in which they occur [9, 10], where the concepts are defined as high quality documents, such as Wikipedia pages. ESA can be used to enrich information retrieval (IR) tasks such as query expansion [1] or document clustering [13], by providing words that can be substituted for one another based on their co-occurrence in the documents of the *model collection*. This which is used to calculate text relatedness is often different from the *query collection* where these *query documents* come from. If a query word does not occur in the model collection, ESA will have a null vector for that word, for this reason it is important to use a model collection that is within the same domain as the query collection. Likewise, if a word is infrequent within the model collection it may not co-occur within same concepts with its related words, therefore leading to a null relatedness value. In the Wikipedia-based ESA models, the annotated links between the documents are used to boost similarity between words that occur within highly-linked documents; however, a high-quality structure, such as this, may not be available in every domain-specific collection.

Latent topic models, such as LSA [8] or LDA [6], innately incorporate some degree of cross-document smoothing; but, the resulting topics are not as clearly interpretable as the human-labelled concepts of the ESA model. In this paper we introduce two new ways of integrating document similarity into ESA, both of which preserve this key interpretability aspect of ESA. The first method can be run *ad-hoc* at retrieval time and expands the number of concepts representing the original query text by querying the model collection with the key-words from the original concepts. In the second, we use ESA with semantic kernel functions and the mathematical framework associated with these functions to produce new kernel matrices and combinations of these matrices. Our primary intention in this paper is not to apply the kernels to a specific classification or clustering task, but to investigate the effects that different kernel transformations and combinations have on the quality of the relatedness between query documents modelled by ESA, while preserving the original concept mapping.

The rest of this paper describes the methodology and experiments undertaken to investigate these two approaches. In the next section we discuss relevant related work. We then describe ESA and our implementation. In Sect. 4 we will describe the two methods, and in Sect. 5 we describe the experiments and provide the results. Finally, we conclude this work with a discussion of these methods and results within the wider context of recent investigations into ESA.

2 Related Work

Originally, it was thought that ESA performs well in word sense and textual similarity tasks because of the perceived orthogonality of the documents in the model collection [10], that is, in an encyclopedia each document covers a specific concept. This, however, has since been shown to be incorrect [11], and dimensions formed from mixtures of Wikipedia documents, the Reuters corpus, and even random indexing [4] produce acceptable results. Moreover, although documents in Wikipedia describe different concepts, many of these concepts cover related subjects, and as such are not truly orthogonal.

Several methods, which exploit this subject overlap between documents, been proposed. Most of these use the linking structure between the Wikipedia documents in order to enrich the word-word similarities [9, 15]. This structure can also be used to augment the matrix representation of ESA, for example by re-weighting the concepts using PageRank [13]. Interest in calculating inter-document similarities also extends to other topic models and has lead to revised LDA algorithms [5, 12].

Both of our approaches are different from the ones mentioned above. In particular, we are attempting to induce the similarities from within the document space itself, without relying on Wiki-specific features. This would allow these ESA extensions to be applied with any model collection, not just Wikipedia.

The related work mainly comes from kernel method literature. We are applying semantic kernels, which were introduced in order to incorporate latent semantic analysis into the kernel space for use with the support vector machine [7]. These kernels, however, were not used to incorporate external knowledge, but to transform the training data matrix using LSA. LSA kernels have been applied in this way to reduce the ESA document space when used in conjunction of kernel-based text classification and clustering algorithms [13] on the Reuters collection.

3 ESA Algorithm and Implementation

The ESA algorithm can be described mathematically as a series of matrix transformations. In practice, it is implemented as an inverted index accessed through a search engine, which improves the speed of retrieval of the term-concept vectors. In this section, we first describe ESA in a way that will be consistent with the kernel-based experiments further on. We then provide the implementation details that are necessary for reproducibility.

3.1 ESA

Given a model collection, let $\boldsymbol{W_F}$ be an inverted index where each word in the collection is mapped to the documents (concepts) in which it occurs. $\boldsymbol{W_F}$ is a very sparse $M \times N$ matrix, where M is the number of unique terms in the collection and N is the number of documents. Each non-empty cell $w_{Fij} = \text{freq}(i, j)$ in $\boldsymbol{W_F}$ indicates the number of times term m_i occurs in the document n_j. For each word, we can use a search engine to retrieve the top K most representative documents from this index. The term-document pairings can be weighted using standard retrieval functions, such as *tf.idf* or *BM25F* [14], to produce an augmented concept matrix, for example, $\boldsymbol{W_C}$, where $w_{Cij} = \text{bm25f}(i, j)$.

A query document can be represented as a $1 \times M$ vector $\boldsymbol{x}$, where the non-zero elements x_i indicate the number of times word m_i occurs in the query. In its simplest form the ESA algorithm takes a query document and transforms this bag-of-words representation into a bag-of-concepts representation based on the top K retrieved concepts for each of the words that occur in the query. This can be described as the multiplication between the query vector and an augmented index matrix $\boldsymbol{x_C} = \boldsymbol{x}\boldsymbol{W_C}$, where the resulting concept vector $\boldsymbol{x_C}$ is $1 \times N$. The semantic relatedness or similarity between two documents is calculated using some distance metric, such as cosine similarity, between their augmented vectors:

$$\text{sim}_{ESA}(\boldsymbol{x_C}, \boldsymbol{y_C}) = \frac{\boldsymbol{x_C} \times \boldsymbol{y_C^T}}{|\boldsymbol{x_C}||\boldsymbol{y_C}|} = \frac{\boldsymbol{x}(\boldsymbol{W_C}\boldsymbol{W_C^T})\boldsymbol{y^T}}{|\boldsymbol{x_C}||\boldsymbol{y_C}|} \tag{1}$$

This is equivalent to a normalised generalised vector space model (GVSM) [11, 13, 17], GSVM $= \boldsymbol{x}\boldsymbol{G}\boldsymbol{y^T}$, where $\boldsymbol{G}$ is a square matrix of word-word similarity values.

3.2 Implementation

We implemented ESA with the preprocessed Wikipedia2005 collection[1] [9]. We indexed this collection using Whoosh[2], a search engine implemented in Python. We initially indexed all of the files that complied with the standard ESA quality specifications. We achieve a top performance of 0.72 on the full dataset (without using Wikilinks-based document smoothing), as measured by the Spearman correlation, on the WordSim-353 dataset.[3] This dataset consists of 353 pairs of words with a gold standard indicating their perceived similarity as an average of scores given by 13 to 16 annotators. The original ESA performance (with Wikilink smoothing) reported on this dataset is 0.74 [9].

We then re-indexed only the preprocessed text and full titles of the 10,000 longest articles. This was done for two reasons. Firstly, by choosing the longest documents we ensure wide word coverage, while by limiting the number of documents we ensure that the words in our test datasets occur with lower frequency in the model collection. This leads to the conditions where we have non-null but very sparse vectors for the words in the test collection, *i.e.* the exact conditions where we expect document smoothing to be effective. Secondly, it is computationally faster to manipulate these smaller matrices for experimental purposes. On this reduced collection we get a baseline ESA performance of 0.65.

Indexing. We only indexed the files had at least 5 incoming or 5 outgoing links and at least 100 unique terms, after stopword removal. Indexing was done without stemming, but words of length less than 2 and numbers were ignored.

Retrieval Settings. We used the *BM25F* retrieval function with the standard settings ($B = 0.75$, $K_1 = 1.2$) because the performance was significantly better than with tf.idf (0.72 vs. 0.65 on the whole dataset). We did not apply stemming at indexing time, but approximated it at query time, by constructing a compound query from all the terms in the index that conflate to the same stem as the query word. For 10,000 document index, the best performance was achieved when maximum 250 results were retrieved per word. All of the experiments in Sect. 5 use the reduced index and these retrieval settings.

4 Methods

In this section we will describe the semantic kernel (ESA-SK) and pseudo-query expansion (ESA-PQE) approaches for introducing concept similarities into ESA.

[1] `http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/wikiprep.html`
[2] `http://pypi.python.org/pypi/Whoosh/`
[3] `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`

4.1 Semantic Kernel (SK)

The kernel function transforms the $M \times N$ input data to a square, positive semi-definite, $M \times M$ matrix, called the *kernel* [16]. A matrix $\boldsymbol{K}$ is positive definite when for any vector $\boldsymbol{\alpha} \in \mathcal{R}^M$, $\boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \geq 0$. Kernel construction is conducted using a kernel function $\kappa = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle$, which is any function that can be reformulated as an inner product given some transformation function ϕ. The functions are also governed by a set of closure properties [16, Chap. 3]. One of these properties, which is key for the construction of semantic kernels, is that a new kernel can be constructed by embedding a positive definite matrix into a linear product:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \boldsymbol{K} \boldsymbol{y}^{\boldsymbol{T}} \tag{2}$$

The fact that ESA is a GSVM [11, 13, 17], which corresponds closely to (2), leads to an intuitive way to examine kernel extensions of ESA. The other key properties are that the sum and the product of two kernels, and the product of a kernel and a scalar, are also valid kernel matrices. Therefore a document-feature frequency matrix can be transformed by a number of available kernel functions, and the resulting kernel matrices can be combined in a number of ways, in order to produce a new representation of the data. In fact, given feature alignment between several different matrices, we can combine multiple models through this framework.

Some of the common kernel functions include the cosine similarity used in the original ESA formulation, as well as the Gaussian and polynomial kernels. More recently, [18] have introduced a set of kernels derived from standard IR ad-hoc retrieval functions, including an asymmetric kernel derived from the *BM25F* function. This provides the ability to treat query texts differently from the documents that are being retrieved, but also allows for ways to integrate some of the IR-based aspects of the ESA model.

Method. With this method, we are investigating whether the ESA model can be improved through introduction of concept correlations, without distortion of the one-to-one mapping of the concepts to easily interpretable Wikipedia articles. We are able to do this in a principled way within the kernel algebra framework. Let $\boldsymbol{W}$ be the term-concept matrix produced by ESA, then $\boldsymbol{K_C} = \kappa(\boldsymbol{W^T}, \boldsymbol{W^T})$ is an $N \times N$ kernel matrix representing concept-concept similarity, as defined by the particular kernel function used. We can then integrate this into the ESA model, and in order to calculate similarity between any two documents $\boldsymbol{x}$ and $\boldsymbol{y}$ within the concept space, we have:

$$\kappa_{\text{ESA}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x_C} \boldsymbol{K_C} \boldsymbol{y_C^T} = \boldsymbol{x}(\boldsymbol{W} \boldsymbol{K_C} \boldsymbol{W^T}) \boldsymbol{y^T} \tag{3}$$

The functions we use are the standard cosine kernel, semantic Gaussian kernel, and semantic LSA kernel:[4]

[4] The semantic kernels were normalised using the kernel normalisation technique described in `http://qwone.com/~jason/writing/normalizeKernel.pdf`

$$\begin{aligned}
\kappa_{\cos}(\boldsymbol{x},\boldsymbol{y}) &= \frac{\boldsymbol{x}\boldsymbol{y}^{\boldsymbol{T}}}{|\boldsymbol{x}||\boldsymbol{y}|} \\
\kappa_{\mathrm{gauss}}(\boldsymbol{x},\boldsymbol{y}) &= \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{2\sigma^2}\right) \\
\kappa_{\mathrm{ESAgauss}}(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{x}(\boldsymbol{W_O}\kappa_{\mathrm{gauss}}(\boldsymbol{W_I^T},\boldsymbol{W_I^T})\boldsymbol{W_O^T})\boldsymbol{y}^T \\
\kappa_{\mathrm{ESAlsa}}(\boldsymbol{x},\boldsymbol{y}) &= \boldsymbol{x}(\boldsymbol{W_O}\kappa_{\cos}(\boldsymbol{V_p},\boldsymbol{V_p})\boldsymbol{W_O^T})\boldsymbol{y}^{\boldsymbol{T}}
\end{aligned} \tag{4}$$

In the LSA kernel, $\boldsymbol{V}$ is the $M \times M$ matrix produced by singular value decomposition of $\boldsymbol{W_I}$ and p is a number of singular vectors chosen by examining the diagonal matrix $\boldsymbol{\Sigma}$, $\boldsymbol{V_p}$ is a $M \times p$ of latent topics in the documents. The outer ($\boldsymbol{W_O}$) and the inner ESA matrices ($\boldsymbol{W_I}$) do not have to have the same number of words, provided there is a one-to-one correspondence between the document vectors.

4.2 Pseudo-Query Expansion (PQE)

The second approach we adopted is used to suggest more concepts for words that do not directly occur in at least K documents. For example, for the word *cup* we find $K = 250$ results including: *Stanley Cup, FA Cup, Coffee*, and *Sake*. However, for the word *tableware* we only get 4 hits: *Silver, Civilization, Plastic*, and *Blackface*. Consequently, the cosine similarity between the concept vectors corresponding to these related words is very small (0.0175). When we add $K-4$ documents which are related to these top 4 hits, including *Copper, Laser Engraving*, and *Chopsticks*, cosine similarity rises to to 0.0400.

Method. We call this method pseudo-query expansion because we create new queries from the top concepts related to the original query, and we then use these new queries to expand our pool of related concepts. Firstly we query the model collection with the original query text *e.g. tableware*. We take up to 100 top documents retrieved, and from each of these we create a query using top 100 key words.[5] With each of these queries we retrieve up to 100 documents (subhits). Throughout these 100 searches, each time we encounter a document we add their BM25F score to their overall rating. In the end we sort the documents according to their total retrieval score and by adding the top rated documents we ensure that each word is associated with up to K concepts. This approach has several constants that were chosen without any testing, and perhaps better performance could be gained with tuning.

5 Experiments and Results

In order to test whether there is an improvement in textual similarity when we employ document smoothing, we will compare the performance of the baseline

[5] They key words are chosen by using an inbuilt search engine function with the Bo1 model [3].

ESA with the **PQE** and **SK** methods, as well as combinations of both (**PQE-SK**). We compare the methods and tune their parameters on the WordSim-353 dataset and then apply the best performing methods to two sentence similarity datasets from the SemEval conference [2].

5.1 Data

ESA Matrices for Kernel Experiments. To construct the ESA matrices we use the subset of the Wikipedia2005 corpus as described in Sect. 3.2. We needed to construct a matrix representation of the index to use for the kernel experiments. The index contains over 200,000 words, so we needed an effective way of extracting a smaller matrix. In order to do this, we indexed the top 100 terms[7] for each document, providing that they occurred in at least 10 documents. We then created a matrix of term-document *BM25F* values by retrieving up to top 250 documents for each of these terms. We refer to this as the `BM25F` matrix. We also used a second approach which we refer to as `Freq`. Here we export the index vector for each of these terms to produce a term-document frequency matrix. We also experimented with using the top terms for the collection, but this did not deliver significantly different results. We conflate all the terms to their stemmed representation and the resulting matrices contain 10,000 documents and approximately 10,000 words. These matrices are used as the inner $\mathbf{W_I}$ matrices.

Word Similarity Data. To test the word-word similarities we use the 353 word pairs from the WordSim-353 dataset (Sect. 3.2). For each of the words in the dataset we extract a vector of term-document *BM25F* values, with and without document extensions.

Sentence Similarity Data. To calculate similarity between two pieces of text we use two of the sentence datasets used in the SemEval 2012 Task 6 [2], specifically the MSRvid and MSRpar test collections. Each of the collections contains 750 pairs of sentences and an averaged gold standard similarity value. The MSRpar collection is more difficult as it contains longer sentences related to current events, while the MSRvid collection contains highly-similar short sentences. For each sentence we extract the term vectors. We also extract the term-document *BM25F* vector, with and without document extensions for each of these terms. This is our outer $\mathbf{W}_O$ matrix.

5.2 Word Similarity – Training

In this section we perform the following experiments on individual words in order to find the settings that lead to the largest increase over the baseline ESA method.

ESA (Baseline). We apply the simple cosine kernel to the ESA $\mathbf{W}_O$ matrix containing the weighted top-K document values for each of the words in the test data. The Spearman (**0.6481**) and Pearson (**0.5233**) correlations between the similarities in this kernel matrix and the gold standard are our baseline.

ESA-SK. For this experiment we apply the $\kappa_{ESAgauss}$ and κ_{ESAlsa} kernel functions to both the frequency and weighted inner matrices. We use the above baseline data as the outer matrix. In the LSA kernel experiments we apply singular value decomposition (SVD) to the inner matrix and then we chose top p vectors from the $N \times N$ matrix $\mathbf{V}$. For the `BM25F` matrix we calculate p by choosing all the right singular vectors with singular values larger than 79 (values on the diagonal of matrix $\mathbf{\Sigma}$) and for `Freq` we get the best results by choosing the vectors with values larger than 75.

We find that the Gaussian kernel, at best, makes no difference. Using LSA kernel with the `Freq` matrix, on the other hand, statistically improves classification over the baseline.

ESA-PQE. In this experiment we use the query-expansion-based document extension method to add more relevant documents to each term and potentially reweigh some of the concepts. We find that using this in the cosine kernel provides an apparent improvement in both correlation matrix, however the t-test finds that the similarities are not statistically different from the baseline.

ESA-SK-PQE. Using the same inner matrices as in the ESA-SK experiments, we apply the extended document outer matrix. We find that using LSA with the `BM25F` matrix improves the performance over the baseline, but not over the simple cosine kernel from ESA-PQE.

Kernel Combination. Finally, one of the advantages of using the kernel approach is the ability to use various kernels as building blocks of new kernels. Here we take the two best performing LSA kernels from Table 1. Let $\mathbf{K}_1$ be the best performing kernel from the ESA-SK experiment, and $\mathbf{K}_2$ be the best performing LSA kernel from the ESA-PQE-SK experiment. We can then construct a new kernel $\mathbf{K} = \alpha\mathbf{K}_1 + (1 - \alpha)\mathbf{K}_2$. Empirically, we find that $\alpha = 0.3$ gives the best value. The combined kernel result is **0.6735** for Spearman correlation and **0.5621** for Pearson.

Results. The results are shown in Table 1, with statistically significant improvement over the baseline marked in bold. We find that, without applying document extensions, the best results are achieved by using ESA with an LSA kernel constructed from the `Freq` matrix. With document extensions, the best results are when using LSA with the `BM25F` matrix. Using document extensions on their own improves the Spearman correlation more than the Pearson correlation, and the t-test shows that the difference in predictions is not statistically significantly better than without PQE. Likewise, using LSA on top of document

Table 1. Word similarity experimental results

	ESA*	**ESA-SK**				**ESA-PQE**	**ESA-PQE-SK**			
Kernel	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAgauss}}$		$\kappa_{\mathbf{ESAlsa}}$		$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAgauss}}$		$\kappa_{\mathbf{ESAlsa}}$	
Inner Mat.		`Freq`	`BM25F`	`Freq`	`BM25F`		`Freq`	`BM25F`	`Freq`	`BM25F`
Spearman	0.6481	0.6483	0.6476	**0.6648**	0.6533	0.6635	0.6635	0.6635	0.6636	**0.6679**
Pearson	0.5233	0.5233	0.5233	**0.5854**	0.5434	0.5311	0.5311	0.5311	**0.5719**	0.5402

extensions does not significantly improve the similarities over the PQE experiment. The statistically significant improvement comes from applying the LSA kernel to the plain ESA (0.06 increase in Pearson correlation) and by using this kernel in a combined kernel (0.025 increase in Spearman).

5.3 Sentence Similarity – Testing

From the previous experiment we found that LSA kernels and kernels combined with 70-30% ratio provide the best results. We also found the right number of dimensions for the LSA kernel. In this experiment we will apply these settings to compare similarity between pairs of sentences. Because there might be some lexical overlap between the sentences, we provide a baseline calculated from this overlap, we examine the effects that applying ESA to all of the words that occur in the sentences has, as well as, the effect of applying ESA only to those words whose stems do not match. We performed the following experiments.

Text. For each of the datasets we computed the sentence-sentence similarity based solely on the lexical features that occur within the sentences. We removed stop words and stemmed the remaining words, in order to be consistent with the conflated representation in the ESA matrices. This provided a baseline slightly higher than without stemming. The baseline for MSRpar is consistent with the previously published baseline [2], but our MSRvid baseline is significantly higher (0.80 compared to 0.30), indicating that perhaps the published baseline was too low.

ESA and ESA-SK. In the result tables we combine these experiments under the heading ESA for ease of readability. We use ESA settings from Sect. 5.2. For ESA-SK we use the settings from the best performing kernel which was LSA with the `BM25F` matrix and the number of topics that was found to lead to best performance in the word similarity experiments.

ESA-PQE and ESA-PQE-SK. In the result tables we combine these experiments under the heading ESA-PQE for ease of readability and use the same settings for the LSA kernel as indicated in the ESA-SK experiments.

Text+ESA and Text+ESA-PQE. In these experiments we isolate only the words that occur in either of the test sentences, but not in both. This is done as a simple operation on the document vectors. We apply the document smoothing to these words only to produce a very sparse kernel (**K1**) of similarities. This kernel is potentially negatively correlated with the gold standard. We combine this kernel with the baseline textual similarity kernel (**K2**), from the **Text** experiment, above. We generate a new similarity kernel using the combination function (Sect. 5.2), with $\alpha = 0.3$, as found in training.

Table 2. Sentence similarity experimental results - MSRpar

	Text*	**ESA**		**Text+ESA**		**ESA-PQE**		**Text+ESA-PQE**	
Kernel	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$
Spearman	0.4200	0.2982	0.4219	**0.4407**	**0.4450**	0.3389	0.4144	**0.4381**	**0.4377**
Pearson	0.4348	0.2810	0.4320	**0.4540**	**0.4589**	0.3222	0.4231	**0.4489**	**0.4496**

Table 3. Sentence similarity experimental results - MSRvid

	Text*	**ESA**		**Text+ESA**		**ESA-PQE**		**Text+ESA-PQE**	
Kernel	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$	$\kappa_{\mathbf{cos}}$	$\kappa_{\mathbf{ESAlsa}}$
Spearman	0.7921	0.7025	**0.8125**	0.7971	0.7981	0.7324	**0.8148**	0.7982	0.7983
Pearson	0.8002	0.6998	**0.8182**	0.8036	0.8047	0.7340	**0.8193**	0.8032	0.8048

Results. The results are shown in Tables 2 and 3. All of the results are statistically different according to the t-test, so we only highlighted the highest in each table. We find that using semantic kernels (SK) or document extension (PQE) consistently improves over baseline ESA; however, the baseline ESA and ESA-PQE are consistently much worse than just simple bag-of-words similarity between stemmed words that occur in the test sentences. There is no consistency in the best performing settings across the two datasets. On MSRpar, we find that applying ESA with or without the LSA kernel smoothing and/or document extensions only to the words that are mismatched between the documents, and using this in a combined kernel with the textual similarity kernel, provides the best performance. On MSRvid, these kernels likewise provide an improvement over the baseline text similarity, but the best performance comes from using the LSA kernel both on ESA and ESA-PQE data.

6 Conclusion

In this paper we examined two approaches for including document similarity data into ESA, without altering the explicit concept mapping. One extends the number of concepts that represents the word, by querying the search engine for documents that are similar to the top documents that contain the word. The second approach builds a matrix of document-document similarities and integrates this matrix into the ESA word-word similarity calculation using kernel combination functions. Both approaches provide similarities that are more correlated with the gold standard assessment of word-word similarities. The most consistent approach uses a semantic LSA kernel, or combinations of such kernels.We found that using ESA on sentence data may require selective application only to mismatched words, but that using the kernel approach in most cases improves the correlation with the gold standard. Future work would include making this approach practical on the full scale Wikipedia-based ESA matrix.

Acknowledgements. This work has been funded in part by the European Union under Grant No. 248458 for the Monnet project and No. 258191 for the PROMISE project, as well as by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

1. Aggarwal, N., Buitelaar, P.: Query expansion using wikipedia and dbpedia. In: Forner, P., Karlgren, J., Womser-Hacker, C. (eds.) CLEF (2012)
2. Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: The First Joint Conference on Lexical and Computational Semantics, SEM 2012, Montréal, Canada, June 7-8, pp. 385–393. Association for Computational Linguistics (2012)
3. Amati, G.: Probability models for information retrieval based on divergence from randomness. PhD thesis, University of Glasgow (2003)
4. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 245–250. ACM, New York (2001)
5. Blei, D., Lafferty, J.: Correlated Topic Models. In: Advances in Neural Information Processing Systems, vol. 18, p. 147 (2006)
6. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022 (2003)
7. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent semantic kernels. Journal of Intelligent Information Systems 18(2), 127–152 (2002)
8. Deerwester, S., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.: Indexing by latent semantic analysis. Journal of the Society for Information Science 41(6), 391–407 (1990)
9. Gabrilovich, E.: Feature generation for textual information retrieval using world knowledge. PhD thesis, Technion - Israel Institute of Technology, Haifa, Israel (December 2006)
10. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI 2007, pp. 1606–1611. Morgan Kaufmann Publishers Inc., San Francisco (2007)
11. Gottron, T., Anderka, M., Stein, B.: Insights into explicit semantic analysis. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM 2011, pp. 1961–1964. ACM, New York (2011)

12. Li, W., McCallum, A.: Pachinko allocation: Dag-structured mixture models of topic correlations. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 577–584. ACM, New York (2006)
13. Minier, Z., Bodo, Z., Csato, L.: Wikipedia-based kernels for text categorization. In: Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2007, pp. 157–164. IEEE Computer Society, Washington, DC (2007)
14. Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 extension to multiple weighted fields. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM 2004, pp. 42–49. ACM, New York (2004)
15. Scholl, P., Böhnstedt, D., Domínguez García, R., Rensing, C., Steinmetz, R.: Extended Explicit Semantic Analysis for Calculating Semantic Relatedness of Web Resources. In: Wolpers, M., Kirschner, P.A., Scheffel, M., Lindstaedt, S., Dimitrova, V. (eds.) EC-TEL 2010. LNCS, vol. 6383, pp. 324–339. Springer, Heidelberg (2010)
16. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York (2004)
17. Wong, S.K.M., Ziarko, W., Wong, P.C.N.: Generalized vector spaces model in information retrieval. In: Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1985, pp. 18–25. ACM, New York (1985)
18. Xu, J., Li, H., Zhong, C.: Relevance Ranking Using Kernels. In: Cheng, P.-J., Kan, M.-Y., Lam, W., Nakov, P. (eds.) AIRS 2010. LNCS, vol. 6458, pp. 1–12. Springer, Heidelberg (2010)