Language Resources and Evaluation manuscript No.
(will be inserted by the editor)

Is Singular Value Decomposition Useful for Word Similarity Extraction?

Pablo Gamallo Otero · Stefan Bordag

Received: date / Accepted: date

Abstract In this paper, we analyze the behaviour of Singular Value Decomposition in a number of word similarity extraction tasks, namely acquisition of translation equivalents from comparable corpora. Special attention is paid to two different aspects: computational efficiency and extraction quality.

The main objective of the paper is to describe several experiments comparing methods based on Singular Value Decomposition (SVD) to other strategies. The results lead us to conclude that SVD makes the extraction less computationally efficient and much less precise than other more basic models for the task of extracting translation equivalents from comparable corpora.

Keywords Information Extraction · Word Similarity · Comparable Corpora · Singular Value Decomposition

1 Introduction

Singular value decomposition (SVD) is a matrix algebra operation that can be used to reduce matrix dimensionality yielding a new high dimensional abstract space, in a similar way as *principal component analysis*. There is a large family of word space models applying SVD to reduce the co-occurrence matrix extracted from the input corpus. Arguably, the most popular word space model based on SVD to extract semantic information from raw text is Latent Semantic Analysis [27], which represents the vector space as a word-by-document co-occurrence matrix. If the main application is information retrieval, that model is also referred to as Latent Semantic Indexing

F. Author
Departamento de Língua Espanhola, Faculdade de Filologia
Universidade de Santiago de Compostela, Galiza, Spain
E-mail: pablo.gamallo@usc.es

S. Author
Max Planck Institute for Social Anthropology in Halle/Saale
Germany
E-mail: bordag@eth.mpg.de

(LSI) [11]. Usually it is being used for various kinds of similarity computations, such as (semantic) word similarity. There are also derivations of the initial model, such as probabilistic LSA, [22,23], and many different applications ranging from writing style discovery [49] to video categorization [47], with citation counts of the original LSA publication ranging in the several thousands.

Proponents of SVD in LSI (for example [45,31,34,6]) and LSA (for example [38, 48,32,22]) argue that this technique provides the word space model with two positive features: on the one hand, matrix reduction allows engineering applications to be faster and less memory demanding with optimizations such as the Hebbian algorithm [20] or QUIC-SVD [24], see also next section for more details. On the other hand, such a high dimensional abstract model is claimed to capture many human cognitive abilities, ranging from acquisition recognition vocabulary to sentence-word semantic priming and judgments of essay quality, as evidenced in many LSA online tutorials. In short, SVD is supposed to make information extraction applications more computationally efficient and more similar to human semantic acquisition. However, experiments and evaluations comparing SVD-based techniques with other models have not always shown that such a dimensionality reduction improves the quality of similarity computations. In some experiments, SVD-based approaches performed better than baseline strategies, but there are cases where they produced worse results and there is a body of work describing these mixed results [33,29,42]. As far as the efficiency is concerned, no comparison among different word space models has been reported in previous work. It seems as if it is simply assumed that SVD reduces the computational cost of semantic extraction algorithms.

In this paper, we first analyse the computational efficiency of SVD matrix reduction before computing vector similarity. We describe more efficient ways of representing a sparse matrix without previously computing SVD (see Section 2). Then, in Section 3, we discuss some problems underlying previous experiments where SVD-based methods were compared to other approaches in a particular semantic extraction task, namely synonymy detection. Then, Section 4 briefly introduces a number of methods to extract translation equivalents from comparable corpora. Some of them rely on dimensionality reduction by SVD. In Section 5, we describe an experiment to compare and evaluate the methods introduced in the previous section. We propose a non-ambiguous, robust, and large-scale evaluation method.

Such an evaluation allows us to conclude that SVD-based methods are less computationally efficient and much less precise than other word space models for the task of extracting translation equivalents from comparable corpora. While we do not draw conclusions about tasks we did not evaluate explicitly, the described procedure allows us to formulate hypotheses about which other tasks would obtain similar results, such as for instance word similarity extraction from monolingual corpora.

2 Is SVD Faster for Similarity Computation and More Economical for Storage?

It is assumed by SVD proponents that a matrix reduced by SVD "has the advantage that all subsequent similarity computations are much faster"[37], since the final matrix representation "is more economical, in the sense that N original dimensions have been replaced by the $k < N$ best surrogates by which they can be approximated" [11]. Other voices state that "The surprisingly small requirements of SVD dimensions

resolve the computational restrictions" [45] or that "The use of SVD results in a very large space and processing time advantage by drastically reducing the size of the representation space. If we took LSA without SVD as the original basis for comparison, and then discovered the advantages of SVD with its ability to 'do more with less', it would clearly be judged superior to the non-SVD LSA model." [48], or "In order to solve these [high-dimensionality and sparseness] problems, the original n-dimensional vector space is converted into a condensed, lower-dimensional, real-valued matrix using Singular Value Decomposition" [31]. Sometimes even server-costs are put directly into connection with running SVD: "In these [commercial] applications, the processing time and RAM required for SVD computation, and the processing time and RAM required during LSI retrieval operations are all roughly linear in the number of dimensions, k, chosen for the LSI representation space. In large-scale commercial LSI applications, reducing k values could be of significant value in reducing server costs." [6].

We claim that the gain in computational efficiency by operating on the reduced set of matrix dimensions can be easily outperformed by an efficient data representation making use of the fact that the co-occurrence matrix built from any linguistic corpus is sparse. In order to compute the similarity between all words it is not necessary to reduce the dimensionality of the entire vector space. Instead of representing the whole word space model as a full matrix (with n^2 required storage space, or trading space for time by using something like the Hebbian algorithm [20]), it could be represented in such a way that a vector uses only as much memory as there are non-zero entries in it. Zero values are easily induced, or rather assumed, later by the algorithm used to compute vector similarity.

2.1 Data Structures for Storing Matrices

Given the power-law distribution of words in a corpus, all co-occurrence matrices representing lexical knowledge are sparse. When storing and manipulating large sparse matrices on a computer, it is beneficial and often necessary to use specialized data structures that take advantage of the sparseness. Many if not most entries of a sparse matrix are zeros that do not need to be stored explicitly. Ignoring the sparseness, a matrix can be stored as a two-dimensional array, including both its zero and non-zero elements. Alternatively such a matrix can be stored in various packed storage modes, only including non-zero elements.

Tables 1 and 2, respresent co-occurrences that were extracted from the following sentence with a window size of 1 and taking only content words, recognizable by reference numbers:

"The man_1 who works_2 in the office_3 likes_4 Chinese_5 food_6."

Table 1 shows a word-by-word matrix where target words are represented in rows while word contexts are in columns. Hence, "man" represents both the first row and the first column, "works" is the second row and the second column, and so on. The result is a 6-by-6 matrix with 26 zeros and 10 non-zero values. The only non-zero appearing in the first row ("man") represents its co-occurrence with "works" (second column). Such a data structure is considered to be a naive representation of a sparse matrix and it has a storage space complexity of $O(n^2)$ and an access complexity of $O(1)$ (one computational step needed to access an element).

Table 1 Sparse co-occurrence matrix

0	1	0	0	0	0
1	0	1	0	0	0
0	1	0	1	0	0
0	0	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0

Table 2 Packed storage with hash table

$1- > \{2- > 1\}$
$2- > \{1- > 1, 3- > 1\}$
$3- > \{2- > 1, 4- > 1\}$
$4- > \{3- > 1, 5- > 1\}$
$5- > \{4- > 1, 6- > 1\}$
$6- > \{5- > 1\}$

Table 3 Dense matrix reduced with SVD

0.122	0.845	0.067
0.396	0.190	0.810
0.067	1.060	0.037
0.414	-0.084	0.967
-0.152	0.010	-0.084
0.156	-0.152	0.414

Table 4 Hash table representation of the SVD-reduced matrix

$1- > \{1- > 0.122, 2- > 0.845, 3- > 0.067\}$
$2- > \{1- > 0.396, 2- > 0.190, 3- > 0.810\}$
$3- > \{1- > 0.067, 2- > 0.060, 3- > 0.037\}$
$4- > \{1- > 0.414, 2- > -0.084, 3- > 0.037\}$
$5- > \{1- > 512, 2- > 0.010, 3- > -0.084\}$
$6- > \{1- > 156, 2- > -0.152, 3- > -0.414\}$

Another possible storage mode for a sparse matrix is the hash table depicted in Table 2, with a key-value representation. Keys are structured as a two-dimensional array containing only those row-column pairs with non-zero values. Like in a matrix structure, hashes also allow to access any arbitrary element in a constant amount of time by means of using a hash function that, given a key, computes the address of the value stored for that key. Hashtable implementations usually have a memory overhead of 20% over the actual amount of data points to be stored. However, this is a constant overhead that is much better than the quadratic overhead of a matrix representation. Hence hashtables have a storage complexity of $O(3 \times n \times m + 0.2 \times n)$ where m is the average number of non-zero co-occurrence observations per word and an access complexity of $O(1)$.

2.2 SVD and Dimensionality Reduction

The matrix resulting of applying SVD on a sparse matrix is not sparse any more. It is a condensed and smoothed representation capturing indirect word associations that were not observed in the input sparse matrix. It seems obvious that any dimensional

reduction yields a smaller structure. However, that is true only if we consider the full matrix storage mode. Table 3 represents a new condensed 6-by-3 matrix after having applied SVD and having retained the 3 most important dimensions (those with the greatest variance in the orginal matrix). This condensed structure is smaller than that depicted in Table 1. Yet, as it does not contain zero values, no packed storage would help saving memory. In fact, the hash table depicted in Table 4, built from Table 3, even requires much more memory to be stored than the packed representation (Table 2) of the original sparse matrix. The hash table generated after SVD contains 18 values while that built from the original matrix only contains 10. With larger corpora and more sparse matrices, such a size difference tends to be bigger (see Section 5). So, whereas the claim that SVD saves storage space is true (for instance in [37]), the savings pale in comparison with other, much simpler methods.

Moreover, a problem arises when we try to compute word similarity taking as input matrices reduced with SVD. This is discussed in the next subsection.

2.3 SVD and Algorithms to Compute Word Similarity

In a non-reduced vector space, in order to compute the pairwise similarity between words it is not necessary to compare each target word with each other. Two words will have non-zero similarity if and only if there is at least one third word with which they both co-occur. This can be expressed as an algorithm that only compares word pairs sharing at least one word context.

Due to the power-law distribution of word frequency, for most target words the list of comparable words is very short (on the order of less than 100 words in a corpus of more than 10 million words). Hence, assuming a fixed threshold of at most 100 other words to be compared with, the complexity of the entire algorithm becomes linear $O(100 \times n)$, instead of quadratic. For the few words that have significantly more than 100 words to be compared with we ignore those other words and accept the risk of unprecise results. In the experiments below we provide results from both ignoring those other words or also taking them.

For those words that do have more than 100 other words to be compared with, the selection criterion is set to take those with the highest significance with either the input word or with the intermediate word (the co-occurrence they share with the input word).

In our example, the algorithm should select only 4 word pairs to be compared ("office" with "man" , "like" with "works" and "food", "office" with "Chinese"), out of 15 possible word pair candidates ($(6!/(6-2)!)/2$). The comparable word pairs share at least one word context, that is, they are associated by means of second-order co-occurrences through a third word [28].

Such an algorithm turns out to be difficult to be implemented if the matrix has been previously reduced by SVD. Indeed, the reduced matrix does not contain explicit information on word-context co-occurrences. All words contain non-zero values in all dimensions. Hence, unless the original sparse matrix also remains accessible, a SVD-reduced matrix does not allow the algorithm to restrict the list of comparable word pairs.

However, proponents of latent semantic analysis could claim that this naive algorithm prevents us from comparing words with higher-order associations. According to this, a SVD-reduced matrix is supposed to represent a more abstract and generic word

space since it tries to capture higher-order associations between words. More precisely, it tries to induce a latent high order similarity structure that does not rely only on word co-occurrences attested in the corpus (i.e., first-order and second-order co-occurrences). If that assumption is true, SVD should allow to infer similarity between two words that have never been observed with the same word contexts, but that could be linked by third-order (or more) co-occurrences. The experiments described later on translation equivalent acquisition will show that this assumption is supported by observable data. Almost 10% of the correct translations proposed by the system are, in fact, bilingual pairs of words that did not co-occur with a common context in the corpus. So, it seems that the generalization performed by SVD is useful to search for latent semantics since it enables finding third-or-more-order similarities. However, these benefits are entirely outruled by the much stronger decrease in precision of second-order similarity.

In sum, this section led us to conclude that SVD does not help to reduce the computational complexity of word similarity algorithms. Rather, compared with other possible matters it even increases computational complexity. On the one hand, the reduced matrix needs more memory space to be stored than existing data structures (e.g., hash tables) and, on the other hand, efficient heuristics restricting the search for similar candidates cannot be easily applied. In fact, hash-tables and the similarity search space restrictions allow an algorithm that is completely linear in its time-complexity, something that is impossible with SVD. Additionally, even the comparisons themselves are cheaper than in SVD, because on average less than 100 co-occurrences are being compared, as opposed to the typical 300 dimensions or more of a SVD matrix.

The next section discusses quality aspects of SVD: how much improvement can be achieved in semantic extraction from reducing dimensionality?

3 Do SVD-based Space Models Improve the Quality of Similarity Extraction?

It has been claimed that SVD provides a significant qualitative improvement in several NLP tasks, namely IR, automatic synonymy extraction, or sense discrimination, for example: "The experiments reported in Schütze [40, 41] give evidence that reduction to this dimensionality does not decrease accuracy of sense discrimination. Space requirements for context vectors are reduced to about 1/10 and 1/20 for a 1,000-dimensional and a 2,000-dimensional Word Space, respectively." [42]. Such an improvement relies on the assumption that SVD is a useful technique to emulate some human learning processes [27], such as the acquisition of lexical information. This situation led some authors to state that LSA is not only a practical strategy to obtain approximate estimates of word similarities, but also a *model* of the human representations underlying substantial portions of the acquisition and utilization of linguistic knowledge. In the foundational paper, Landauer and Dumais [27] proposed that Latent Semantic Analysis (LSA) constitutes a fundamental computational theory of the acquision and representation of knowledge. According to these authors, a statistical technique such as SVD is both psychologically motivated and computationally appropriate to improve results on semantic extraction.

We will not discuss the first statement on psychological motivation. This section will be focused on whether or not it has been clearly demonstrated SVD helps improving semantic extraction.

There exist many tests comparing SVD-based methods with either human judgements or other automatic techniques. One of the most popular tests is to choose the most appropriate synonym for a given word given a restricted list of four candidates. We are interested in those evaluations comparing SVD, not with humans, but with other automatic techniques. To compare the accuracy of two (or more) methods, it is assumed that the system makes the right decision if the correct word is ranked highest among the four alternatives.

We found two drawbacks with this kind of test: one is motivated by the heterogeneity of the elements involved in each experiment, and the other derives from the size of the test itself.

3.1 Hetereogenous Tests

The experiments required to measure the precision of a synonym test involving several variables with many possible instantiations:

- The type of questions. The most used are 80 synonym test questions selected from the Test Of English as a Foreign Language (TOEFL). There are also 50 questions selected from a collection of tests of English as a Second Language (ESL), and 300 Reader's Digest questions.
- The training corpora. Many different corpora were used: some are traditional such as British National Corpus (BNC), containing 100 million tokens, TASA corpus, with 17 million tokens, Grolier's Academic American Encyclopedia (GAAE), with 4,6 million tokens. Other corpora were gathered by web crawling: Stanford Corpus (SC), with 30 million word types, English Gigaword collection (LDC), with a selection of 1,1 million articles of the New York Times, and a terabyte of web data crawled from the general web containing over 55 millions words (TERA). There is also a corpus containing a set of documents retrieved from queries to Altavista (ALTA).
- The type of similarity. Two general types of similarities can be distiguished: one based on plain co-occurrences, using a significance measure such as pointwise mutual information (PMI), the other relying on second-order co-occurrences (e.g, where the association between two words is computed through a number of other words, taken as word contexts).
- The type of context used to define co-occurrences. Some use documents or paragraphs, others windows of size N, and others syntax-based patterns.
- The use or not of SVD. It is possible to separate the methods performing SVD from those using the original sparse matrix. In fact, this is the parameter we would like to evaluate.
- Other parameters: e.g., co-occurrence significance measure, type of similarity measure, use or not of an initial vocabulary (e.g., most frequent words) to define a restricted list of seed contexts, etc.

To evaluate whether or not SVD improves the quality of the extraction, we would need to define an experiment comparing two identical strategies except for the use or not of SVD. However, the experiments performed to measure the precision in synonym tests have differences with regard to, not only the use or not of SVD, but the use of other significant variables: training corpus, type of similarity, etc. Table 5 depicts a brief description of different experiments performed on the TOEFL synonym test

Table 5 Description of several experiments using TOEFL synonym test

	Corpus	Simil	Context	SVD	Other	Prec.	ref.
Rapp1	BNC	2-order	window=2	+	weight:entropy	92.5%	[38]
Baroni	BNC	2-order	window=5	+	seed contexts	91.3%	[2]
Rapp2	BNC	2-order	syntax-based	-		90.9%	[38]
GLSA1	LDC	2-order	window=16	+	PMI and seeds	86%	[32]
PMI1	TERA	1-order	window=16	-		81.25%	[43]
GLSA2	SC	2-order	window=16	+	PMI and seeds	76%	[7]
PMI2	ALTA	1-order	window=10	-		73.75%	[44]
GLSA3	TASA	2-order	window=16	+	PMI and seeds	72%	[7]
Rapp3	BNC	2-order	window=1	-	sim:cityblock	69%	[38]
LSA1	GAAE	2-order	document	+		64.5%	[27]
LSA2	TASA	2-order	document	+		60%	[27]
PMI3	SC	1-order	window=10	-		51%	[7]

questions. Each experiment is described making use of the variables introduced above, i.e., type of questions (here, we only selected experiments using TOEFL), corpus, type of similarity, type of context, use of SVD, other properties, and precision achieved. The table also assigns a bibliographic reference to each experiment. Experiments are ranked by precision rates.

Table 5 lets us observe a number of experimental "families" that will be described in more detail. The experiments with the name LSA are those that follow the standard method defined by Latent Semantic Analysis: they make use of plain occurrence frequency and then second-order similarity based on a word-by-document matrix reduced with SVD. The family of GLSA experiments follows the method based on General Latent Semantic Analysis. They introduce 3 different elements with regard to standard LSA: contexts are not documents but smaller windows; a small vocabulary with frequent words is used to define the contextual dimensions of the matrix (which is then less sparse); this matrix is weighted with pointwise mutual information (PMI). A very similar method is defined in the Baroni experiment. The main difference is that Baroni does not make use of PMI. The experiment performed in Rapp1 can be situated between LSA and GLSA: it uses a small window as in GLSA, but, as in LSA, it does not require a seed vocabulary to restrict the number of word contexts. So far, all experiments relied on SVD. By contrast, the PMI family of experiments is based on a simpler method with first-order similarity (i.e., only direct co-occurrences are needed to compute word similarity), and without SVD. As the computational complexity of such a method is not so high, it can be used to exploit huge document collections gathered by crawlers or retrieved by web search engines. Note that experiment PMI1 uses a huge corpus (TERA) with a terabyte of data. There are two remaining experiments without SVD: Rapp2 and Rapp3. They both use standard 2-order similarity, but they differ in the context definition: syntax-based and window-based, respectively.

As the experiments differ in more than one property, we claim that they do not allow to compare the efficiency of using or not SVD in the task of extracting correct synonyms. For instance, experiments *Rapp1* and *Rapp3* are perhaps those that are more comparable for measuring the accuracy of SVD. They were applied on the same corpus (BNC), they used both second-order similarity, and the window size was very short in both cases: 2 and 1, respectively. It seems that they differ only in whether they use SVD, or not. However, there are more significant differences: *Rapp1* uses

an entropy-based weight to smooth simple co-occurrence frequencies, whereas *Rapp3* seems to use log-likelihood for the same purpose. This is not explicitly said in [38] but it can be inferred from a reference to a previous paper. Moreover, while *Rapp3* uses as similarity coefficient cityblock, *Rapp1* makes use of a more usual metric: cosine. This way, we cannot know whether the differences regarding precision (92% against 69%) are mainly due to the use of SVD or to those other variables. We follow the same reasoning when other pairs of experiments are compared, in particular, when they also differ in the training corpus.

Besides, there is a further problem underlying these experiments: the test sets are very small and no results of significance tests were reported which would allow to induce confidence in the significance of the observed differences. And the fact that some of the observed differences in our own experiments turn out to be not significant is a good argument against trusting the significance of differences in much smaller experiments.

3.2 Small Test

A positive argument in favor of this type of test is that it allows an automatic evaluation. A drawback is that it is too small. We consider it small with regard to two different properties. On the one hand, the list of selected test questions is insufficient: the questions selected from TOEFL and ESL are only 80 and 50, respectively. One single error decreases the total precision of the system by almost 2 percentage points. Additionally, each question has only 4 alternative answers, that is, a baseline method using random choice achieves 25% precision. To select a synonym candidate, the system does not have to compare the target word with all words of the vocabulary but only with a reduced list containing 4 candidates. These two drawbacks make the test not very reliable to compare the performance of extraction methods.

Contrary to this test, we provide another test method with a robust, large-scale automatic evaluation. On the one hand, we use a list of several thousands of test words and, on the other hand, the number of candidates for each choice is the entire available vocabulary. To define such an evaluation protocol, we modify the extraction method to be compared. More precisely, we evaluate methods to extract translation equivalents from comparable corpora. Notice that a translation equivalent of a source word could be conceived as its best synonym in another language. So, as in the synonym tests analyzed above, the goal of our extraction is also to identify "synonyms" of a word in a different language from a sparse matrix with bilingual contexts. The main reason for this change is that evaluation can now be provided with a large-scale gold standard: a bilingual dictionary with thousands of test words and their correct translations.

To compare the efficiency of SVD in such a task, the evaluated methods differ only in whether they use SVD, or not. The remaining parameters (i.e., corpus, type of similarity, measures, and contexts) are constant. This allows us to remove noisy and unpredictable variables when observing the specific behaviour of SVD. This means that at least for the scenario of translingual word similarity extraction, any conclusions obtained from these experiments are likely to be universal, especially if the results remain the same for two different language pairs. These conclusions are not, of course, necessarily generalizable to other scenarios, such as information retrieval even though they do give hints on how a similar comparison might fare there.

4 Methods to Extract Translation Equivalents From Comparable Corpora

The methods we evaluate in the experiments described later rely on a well-known strategy to identify bilingual lexicons from comparable corpora [14,15,36,9,12,25,16,39]. The procedure works as follows: a word w_2 in the target language is a candidate translation of w_1 in the source language if the context expressions with which w_2 co-occurs tend to be translations of the context expressions with which w_1 co-occurs. The basis of the method is to find the target words that have the most similar distributions with a given source word. The starting point of this strategy is a list of bilingual expressions that are used to build the context vectors defining all words in both languages. This list is usually provided by an external bilingual dictionary.

This strategy is very similar to those used to extract similar words from monolingual corpora. It also requires a sparse matrix to compute word similarity. There are, however, two slight differences: on the one hand, the matrix dimensions are constituted by bilingual contexts (seed words or lexical patterns taken from a dictionary) and, on the other, the words to be compared need to belong to two different languages. Given a word in the source language, the most similar ones in the target language are considered as its candidate translations.

The methods briefly described in the next subsections are based on this standard strategy. However, they differ in two specific elements: context definition and matrix construction.

4.1 Window and Syntax-based strategies

The experiments are performed using two different types of word contexts: both window-based and syntax-based contexts.

The window-based technique does not perform any kind of syntactic analysis, but simply considers some window of words as forming the context of the compared words. We follow the method described in [36]. Texts in both languages are lemmatized and POS tagged, and function words are removed. Window size is 2 and word order is taken into account.

The syntactic strategy relies on dependency-based partial parsing. Dependencies are generated by means of *DepPattern*[1], a rule-based partial parser which can process 5 languages: English, Spanish, Galician, Portuguese, and French. To extract syntax-based contexts from dependencies, we used the co-compositional methodology defined in [19], which was inspired by Pustejovsky [35]. Co-compositionality is defined as following: Two words related by a syntactic dependency are mutually constrained and impose linguistic requirements on each other. In particular, we consider that in a Head-Dependent syntactic dependency, not only the Head imposes constraints on the Dependent, but the latter also imposes linguistic requirements on the Head in return. The *DepPattern* toolkit also includes a script aimed to extract co-compositional contexts from the dependencies generated by the parser.

[1] *DepPattern* is a linguistic toolkit, with GPL licence, which is available at: `http://gramatica.usc.es/pln/tools/deppattern.html`

4.2 Different Ways of Building Co-occurrence Matrices

In our experiments, we evaluate the performance of several different types of co-occurrence matrices (see Figure 1). First, we call *baseline* the simplest method that takes as input a sparse matrix containing word-by-context raw co-occurrences. No further operation was applied on the baseline matrix before computing word similarity. Then, some association measures, namely log-likelihood (*log*) and two entropy-based coefficients (*entropyLSA* and *entropyRapp*), are applied on the baseline matrix turning simple co-occurrences into weighted values. Our interpretation of the formula of the original entropy-based coefficient [27], where it is given only in textual form, is the following. The entropy H of a word is:

$$H(word) = -\sum_{i}^{n} p_i log_2 p_i \tag{1}$$

where n is the number of contexts, and probability p_i is:

$$p_i = \frac{freq\,of\,word\,in\,context_i}{freq\,of\,word} \tag{2}$$

Given that the formula is described in the cited article as "ln (1+cell frequency)/entropy of the word over all contexts", the weight assigned to each cell (whose value is 'freq of word in context') is computed as:

$$Weight_{LSA} = \frac{ln(1 + (freq\,of\,word\,in\,context))}{H(word)} \tag{3}$$

Rapp [38] proposed to multiply the local weight and word entropy instead of dividing it, claiming that the results become better from this. Hence, *entropyRapp* is the following formula:

$$Weight_{Rapp} = ln(1 + (freq\,of\,word\,in\,context)) \times H(word) \tag{4}$$

On the basis of these three weighted matrices, we define four reduced matrices: three generated by SVD reduction (*svdLSA*, *svdRapp*, and *svdLog*) and one built with a different filtering-based strategy (*filter*). The filtering method was defined in [5], and consists of the following tasks: the input matrix, which contains log-likelihood values is ranked by decreasing significance. Then, only the N best ones are selected (where $N = 200$ in our experiments). This way, each word is associated, at most, with 200 non-zero weighted values. Given that corpus frequency follows the power-law distribution, only very frequent words co-occur with more than 200 other words. Even if such a filtering strategy only affects very frequent words, it allows us to reduce the number of pairwise comparisons (and thus runtime) significantly, while hopefully not deacreasing accuracy. As it turns out, it even increases accuracy, though insignificantly.

The 8 types of matrices shown in Figure 1 are organized in a hierarchical tree with three levels of complexity. At the top node, we find *baseline* matrix. Then, at the second level, *entropyLSA*, *entropyRapp*, and *log* are weighted matrices that directly depends on *baseline*. Their values are the result of three different association measures, but no reduction is applied: they still contain the same number of objects as *baseline*. Finally, at the third level, *svdLSA*, *svdRapp*, and *svdLog* are the result of two different reduction operations (SVD and filtering) on the weighted matrices.

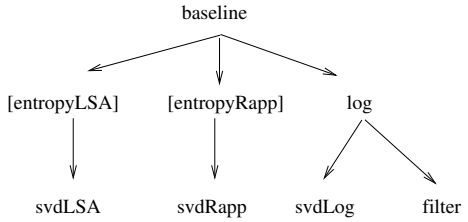

Fig. 1 Tree organization of 4 types of matrices

Two of these matrices, *entropyLSA* and *entropyRapp*, which are noted with brackets in the figure, will not be directly evaluated. They are only used to build SVD reduced matrices. The extraction methods we will evaluate are based on the 6 remaining matrices.

If we combine the 6 evaluable matrices with the two types of contexts defined above, we will be able to evaluate 12 different methods to extract translation equivalents, differing only in two elements, namely the type of context and the type of input matrix. This allows us to easily compare the performance of SVD-based methods with regard to other slightly different strategies. The 12 methods are noted and described as follows:

W-baseline: window-based contexts and raw co-occurrences,
W-log: window-based contexts and log-likelihood values,
W-svdLSA: window-based contexts, entropy-based weight as described in formula 3, and svd reduced matrix,
W-svdRapp: window-based contexts, entropy-based weight as described in formula 4, and svd reduced matrix,
W-svdLog: window-based contexts, loglikelihood weight, and svd reduced matrix,
W-filter: window-based contexts and filtered matrix as described in [5],
S-baseline: syntax-based contexts and raw co-occurrences,
S-log: syntax-based contexts and log-likelihood values,
S-svdLSA: syntax-based contexts, entropy-based weight as described in formula 3, and svd reduced matrix,
S-svdRapp: syntax-based contexts, entropy-based weight as described in formula 4, and svd reduced matrix,
S-svdLog: syntax-based contexts, loglikelihood weight, and svd reduced matrix,
S-filter: syntax-based contexts and filtered matrix as described in [5]

Notations such as *W-svd* and *S-svd* will also be used when either the distinction among diferent SVD-based methods is not relevant or one of them is taken as our by-default SVD strategy.

The other main parameters are not modified: we use the same training corpus and the same type of similarity (second-order). However, it is still possible to compare the behaviour of each one of these 12 methods considering different similarity measures. Finally, we use the Scheffé test to measure which groups of matrix-measure combinations differ statistically significantly from each other.

4.3 Ten Similarity Measures

Each method will be tested against 10 similarity coefficients (see Table 6). Some of them transform all vectors into binary values: binary Baseline (Base), binary Dice (DiceBin), binary Jaccard (JaccardBin), and binary Cosine (CosineBin). By contrast, Cosine, Euclidian distance (Eucl), City-Block (City), Dice (DiceMin), and Jaccard (JaccardMin) use vectors with co-occurrence (or weighted) values. These different similarity metrics between two words, w_1 and w_2, are defined in Table 6, where $BIN(w_1)$ stands for a set representation of the binary vector defining word w_1. This vector is the result of transforming the real-valued vector with co-occurrences or log-likelihood scores into a vector with binary values. The length $\parallel BIN(w_1) \parallel$ of a binary vector $BIN(w_1)$ is the number of non-zero values.

On the other hand, $A(w_1, c_j)$ is an association value of a vector of length n, with j, i, and k ranging from 1 to n. In our experiments, the association value stands for either the simple co-occurrences of word w_1 with a contextual seed expression c_j, or the weight computed using the log-likelihood ratio between the word and its context. For Cosine, the association values of two words with the same context are joined using their product, while for JaccardMin [21,26] and DiceMin [10,46,16] only the smallest association weight is considered. For the Lin coefficient, the association values of common contexts are summed [30], where $c_j \in C_{1,2}$ if and only if $A(w_1, c_j) > 0$ and $A(w_2, c_j) > 0$. Finally, in City, $|x - y|$ represents an absolute value. In sum, we use two types of similarity coefficients: those based on binary vectors and those relying on association values.

5 Experiments and Large-Scale Evaluation

5.1 Corpora and Dictionaries

The experiments were performed on two comparable corpora. First, a Spanish and Galician comparable corpus comprised of news from on-line journals published between 2005 and 2006. As the Spanish corpus, we used 10 million words of two newspapers: *La Voz de Galicia* and *El Correo Gallego*, and as Galician corpus 10 million words from *Galicia-Hoxe*, *Vieiros* and *A Nosa Terra*. In sum the bilingual corpus consists of 2×10 million words. The second comparable corpus has the same size and consists of English and Galician texts. The Galician part is the same as in the previous corpus, while the English part consists of news from Reuters published in 2006. The Galician, Spanish, and English texts were lemmatized and POS tagged using a multilingual free software: Freeling [8]. Since the orientation of the newspapers is quite similar, the three monolingual texts can be considered as more or less comparable.

The bilingual dictionaries used to select the seed words required by the acquisition algorithm are the lexical resources integrated in OpenTrad, an open source machine translation system for Spanish-Galician and English-Galician [1]. The Spanish-Galician dictionary contains about 25,000 entries, and the English-Galician about 12.000.

Table 6 10 similarity measures

$$\mathsf{Base}(w_1, w_2) = \parallel \mathsf{BIN}(w_1) \cap \mathsf{BIN}(w_2) \parallel$$

$$\mathsf{DiceBin}(w_1, w_2) = \frac{2 \parallel \mathsf{BIN}(w_1) \cap \mathsf{BIN}(w_2) \parallel}{\parallel \mathsf{BIN}(w_1) \parallel + \parallel \mathsf{BIN}(w_2) \parallel}$$

$$\mathsf{JaccardBin}(w_1, w_2) = \frac{\parallel \mathsf{BIN}(w_1) \cap \mathsf{BIN}(w_2) \parallel}{\parallel \mathsf{BIN}(w_1) \cup \mathsf{BIN}(w_2) \parallel}$$

$$\mathsf{CosineBin}(w_1, w_2) = \frac{\parallel \mathsf{BIN}(w_1) \cap \mathsf{BIN}(w_2) \parallel}{\sqrt{\parallel \mathsf{BIN}(w_1) \parallel}\sqrt{\parallel \mathsf{BIN}(w_2) \parallel}}$$

$$\mathsf{City}(w_1, w_2) = \sum_j |\mathsf{A}(w_1, c_j) - \mathsf{A}(w_2, c_j)|$$

$$\mathsf{Eucl}(w_1, w_2) = \sqrt{\sum_j (\mathsf{A}(w_1, c_j) - \mathsf{A}(w_2, c_j))^2}$$

$$\mathsf{Cosine}(w_1, w_2) = \frac{\sum_j \mathsf{A}(w_1, c_j)\mathsf{A}(w_2, c_j)}{\sqrt{\sum_j (\mathsf{A}(w_1, c_j))^2}\sqrt{\sum_k (\mathsf{A}(w_2, c_k))^2}}$$

$$\mathsf{DiceMin}(w_1, w_2) = \frac{2\sum_j \min(\mathsf{A}(w_1, c_j), \mathsf{A}(w_2, c_j))}{\sum_j \mathsf{A}(w_1, c_j) + \sum_k \mathsf{A}(w_2, c_k)}$$

$$\mathsf{JaccardMin}(w_1, w_2) = \frac{\sum_j \min(\mathsf{A}(w_1, c_j), \mathsf{A}(w_2, c_j))}{\sum_j \max(\mathsf{A}(w_1, c_j), \mathsf{A}(w_2, c_j))}$$

$$\mathsf{Lin}(w_1, w_2) = \frac{\sum_{c_i \in C_{1,2}} (\mathsf{A}(w_1, c_j) + \mathsf{A}(w_2, c_j))}{\sum_j \mathsf{A}(w_1, c_j) + \sum_k \mathsf{A}(w_2, c_k)}$$

5.2 Comparing Computational Efficiency

The basic word space we built from the Spanish-Galician comparable corpora and the syntax-based approach gave rise to a 17,000 words by 13,000 contexts sparse co-occurrence matrix. The 17,000 most frequent Spanish and Galician nouns (8,000 and 7,000, respectively) are treated as targets, that is, they are the objects of the matrix. The top 13,000 bilingual syntactic patterns are treated as word contexts (i.e., the matrix dimensions).

The window-based approach gave rise to a larger 17,000-by-16,000 matrix, where the target nouns are the same as in the syntactic space. Contexts are the 16,000 most frequent nouns, adjectives, verbs, and adverbs. In both cases, words were previously lemmatized and tagged. We did not work with larger matrices so as to allow the SVD software to be applied without running into RAM memory restrictions. Using SVD, we

Table 7 Hash tables size obtained from different methods

	baseline/log	**filter**	**svd(300)**
Window	4,629,609	1,840,174	5,303,700
Syntax	2,496,888	1,901,600	5,275,200

created two reduced matrices with 1000 and 300 dimensions. Dimensionality reduction was performed with SVDLIBC[2].

To compute word similarity, the input data contained in those matrices is stored in hash tables only containing non-zero values. This is necessary, because the optimizations of our similarity computation program are based on the usage of hash tables. As we expected, the largest hash tables correspond to the dense matrices produced by SVD (see Table 7). For instance, the *S-svd(300)* method (i.e., syntax-based contexts and SVD-reduced matrix with 300 dimensions) yielded a table with 5,275,200 non-zero entries. Note the three different SVD-reduced matrices (*svdLSA*, *svdRapp*, and *svdLog*) have the same size. By contrast, the *S-baseline* and *S-log* methods (without any matrix reduction) have 2,496,888 non-zero entries. The smallest table was built with the *S-filter* algorithm, with merely 1,901,600 values.

As far as runtime is concerned, the slowest process was run with the *W-svd* word space (i.e., window-based contexts and SVD-reduced matrix). The process of computing similarity took 115h10min, using a 2.33GHz CPU. By contrast, the same process with the *W-baseline* space took 3 times less time: 35h45min. And with *W-filter*, less than 24 hours. Notice that this includes only the time required by the final similarity process. Dimensionality factorization by SVD was not taken into account. To compute word similarity with *baseline* and *filter*, we applied the efficient heuristic described in Section 2, that is, we selected only those pairwise nouns sharing at least one context. This heuristic can not be applied to SVD-based matrices.

In sum, concerning storage and runtime, the methods based on SVD seem to be less efficient than baseline strategies by quite a large margin.

5.3 Qualitative Evaluation

5.3.1 Protocol

To evaluate the quality of all tested extraction methods for translation equivalents, we elaborated an automatic and large-scale evaluation protocol with the following characteristics. As far as the Spanish-Galician corpus is concerned, the test list conceived as gold standard contains about 14,000 bilingual nominal entries. The English-Galician test list consists of 3,300 nominal entries. Each test list is the result of selecting all words that appear in both the bilingual dictionary and the corpus. *Precision* is the number of correct translations proposed by the system, divided by the number of nouns appearing in the test list and for which the system has proposed a translation. Given a Spanish target word, for instance, a translation is considered as correct only if the correct Galician candidate was ranked among the top-10 most similar words to the target. *Recall* is the number of correct translations divided by the number of nouns in the test list. Finally, *f-score* is the harmonic mean of precision and recall.

[2] `http://tedlab.mit.edu/~dr/svdlibc/`

This evaluation protocol is provided with three positive properties: First, unlike evaluation tests based on small lists of words (e.g., TOEFL), our evaluation makes use of a big list of thousands of test nouns as gold standard. This makes it sound and reliable. Second, unlike the TOEFL or ESL test questions, the list of synonym candidates is not restricted to a small word set. Hence, the baseline of randomly choosing possible words is close to 0%. Third, as finding word translations is akin to identifying strong and well defined synonymy relations between two words (the source word and its translation), our evaluation has the positive aspect of those controlled tests, such as TOEFL, containing non-ambiguous questions elaborated by humans for a specific task. By contrast, other tests relying on general lexical resources such as WordNet are not suited to evaluate well-defined word synonymy.

Furthermore, we test the results with the Scheffé test, an anova post-hoc test [13] comparing the means (of the F-scores) of the various windowing and measure combinations (briefly called algorithms). This test checks all possible groupings of results for whether they do differ significantly or not, with a confidence interval of 99% (i.e. an error probability of less than 1%). Since this is considered to be a conservative significance test, differences found by it are with a very high probability not due to chance. However, this test might fail to find a difference in the performance of two algorithms which does not exclude the possibility of such a difference with more test samples or under different conditions.

We also measure correlation coefficients to obtain information about whether two algorithms perform well on the same words or not. If two algorithms would perform well on two mostly distinct sets of words, they could be combined to boost the overall performance.

5.3.2 Results of the Spanish-Galician Corpus

Before comparing all methods, we first evaluated the performance of the different SVD-based strategies. Table 8 depicts the F-scores of 6 methods with SVD reduction (300 dimensions) over the Spanish-Galician corpus. In particular, we compare the use of SVD with two types of contexts (syntax and window based) and three association measures: loglikelihood (Log), the entropy-based transformation proposed by LSA (equation 3), and the entropy-based version by Rapp (equation 4).

Table 8 Spanish-Galician comparable corpus. F-score (in %) of different svd-based methods (300 dimensions)

Measures	**S-svd LSA**	**S-svd Rapp**	**S-svd Log**	**W-svd LSA**	**W-svd Rapp**	**W-svd Log**
City	6.32	5.92	8.75	7.52	5.93	1.03
Cosine	17.54	**23.39**	17.17	20.79	**25.23**	1.01
Euclidean	5.15	5.89	7.63	7.77	6.48	0.86

The results in Table 8 show that the best scores are those performed using the entropy-based formula defined by Reinhard Rapp. In particular, Rapp's formula gives good similarity estimates with cosine and window-based contexts. This is in accordance with the results depicted above in Table 5, where the window-based method combined

with Rapp's equation yielded the best results on the TOEFL test. In the following, *svdRapp* will be taken as our by-default SVD strategy. Notice that non-binary measures, namely City-Block, Cosine, and Euclidean are the only meaningful coefficients applicable to the dense matrices generated wit SVD. This is due to the fact that the reduced matrix consists of words sharing all high-dimensional contexts. In boolean terms, every word-context association is assigned value 1. So, the results obtained with binary metrics over SVD-reduced matrices are the same as any random technique: close to 0.

In the next experiments, we compare all possible combinations among methods and similarity measures described above in Section 4. Tables 9 and 10 show the F-scores obtained using respectively the windowing technique and the syntax-based strategy over the Spanish-Galician comparable corpus. The columns of each table represent the methods described in 4.2: *W-baseline, W-log, W-filter, W-svd* in Table 9, and *S-baseline, S-log, S-filter, S-svd* in Table 10. Rows represent the 10 similarity coefficients introduced above in 4.3. The SVD-based methods evaluated in these tables are our by-default strategies, namely *S-svdRapp* and *W-svdRapp*.

The evaluation shows that the SVD-based strategies (with either 300 or 1000 reduced dimensions) perform much worse than the other three methods (baseline, log, and filter), which, in fact, do not differ much from each other. We could observe a direct positive relationship between continually raising the number of dimensions and extraction quality without the peak at around 300 dimensions as reported by some of the previously cited researchers. According to proponents of LSA, "it is clear that there is a strong nonmonotonic relation between number of LSA dimensions and accuracy of simulation, with several hundred dimensions (300) needed for maximum performance, but still a small fraction of the dimensionality of the raw data" [27]. By contrast, our experiments show the relation between LSA dimensions and accuracy is monotonic: the more dimensions the matrix contains the higher the accuracy of extraction. The use of 300 and 1000 dimensions in the reported experiments allows us to visualize the linear improvement of performance: the peak is reached by the original non-reduced matrix with 16000 dimensions.

When analyzing the differences between the algorithms we both compute the Pearson correlation coefficient (see Table 11 for a selected number of coefficients) and a simple count statistic. For example, Cosine with *S-svd* finds the correct translation for 3316 target words whereas DiceMin with *S-filter* algorithm is correct for 7350 words. But there are only 65 cases where the *S-svd* method found the correct translation for which *S-filter* did not find the correct translation. In other words, the correct results of the SVD based method differ only by less than 2% from the other methods. The entire matrix of Pearson coefficients and simple count statistics between all algorithms exhibits the same behavior. That is, if algorithms are very distinct in their performance, then the worse of the two compared algorithms barely has correct translations where the better has not correct translations. It follows that the algorithms are not complementary. This is unfortunate because it means that their combination will not produce any gain in performance.

The results of both *S-svd* and *W-svd* cannot be compared with other related work on translation equivalents extraction, since as far as we know SVD has not been used for this specific purpose. On the other hand, we can observe that the syntax-based methods improve the results obtained by the windowing techniques slightly (except if we compare *S-svd* against *W-svd*). The same was observed in recent work [17]. However, the differences between the two methods are not statistically significant.

Table 9 Spanish-Galician comparable corpus. F-score (in %) of window-based methods and 10 similarity measures

Measures	W-baseline	W-log	W-filter	W-svd(1000)	W-svd(300)
Base	0.66	0.66	43.25		
City	0.63	0.58	0.58	5.22	5.93
CosineBin	43.18	43.18	**48.82**		
Cosine	11.26	10.84	11.22	25.53	25.23
DiceBin	**48.49**	**48.49**	**48.82**		
DiceMin	33.88	26.79	26.21		
euclidean	2.63	2.35	2.29	7.16	6.48
JaccardBin	**48.49**	**48.49**	**48.82**		
JaccardMin	33.88	26.79	26.22		
Lin	7.62	6.35	8.20		

Table 10 Spanish-Galician comparable corpus. F-score (in %) of syntax-based methods and 10 similarity measures

Measures	S-baseline	S-log	S-filter	S-svd(1000)	S-svd(300)
Base	5.26	5.26	40.72		
City	1.73	1.75	4.51	5.60	5.92
CosineBin	**48.62**	**48.62**	**48.99**		
Cosine	39.92	42.45	42.57	30.02	23.39
DiceBin	**48.15**	**48.15**	**48.71**		
DiceMin	**47.61**	**50.25**	**50.22**		
Euclidean	6.22	6.21	18.34	5.91	5.89
JaccardBin	**48.15**	**48.15**	**48.71**		
JaccardMin	**47.61**	**50.25**	**50.22**		
Lin	40.89	40.11	40.64		

Table 11 Pearson Coefficient for a selected number of algorithms

	S-Filter DiceMin	S-Filter JaccardMax	S-Log Cosine	S-svd(1000) Cosine	W-Filter JaccardBin
S-Filter DiceMin	1	1,00	0,77	0,47	0,70
S-Filter JaccardMin	1,00	1	0,77	0,47	0,70
S-Log Cosine	0,77	0,77	1	0,53	0,60
S-svd(1000) Cosine	0,47	0,47	0,53	1	0,40
W-Filter JaccardBin	0,70	0,70	0,60	0,40	1

The statistical significance tests reveal that with a significance of 0.973 (below 0.01 would be insignificant) there is a large group of algorithms that do not differ significantly from each other, marked with bold face in the tables. But they all perform significantly better than the rest of the algorithms (which also form groups, but are of less interest). This is in accordance with findings in related literature [4] where precisely the combination of the baseline approach with binary measures performed best.

In sum, these results show that it is not easy to overcome the two baseline strategies: both *W-baseline* and *S-baseline*. Even if *(WS)-filter* and *(WS)-log* perform slightly better than the baseline, differences are actually very small. So, to compute word similarity, the simple co-occurrence sparse matrix represented as a hash table with non-zero values and an appropriate similarity measure behaves, at least, as good as other more elaborated methods.

Table 12 English-Galician Comparable corpus. F-score (in %) of three syntax-based methods and 10 similarity measures

Measures	S-baseline	S-filter	S-svd(300)
Base	1.25	**22.54**	
City	0.43	0.43	1.09
CosineBin	11.37	**21.50**	
Cosine	11.38	8.57	2.93
DiceBin	18.89	**22.19**	
DiceMin	**22.80**	17.55	
Euclidean	1.61	1.21	1.66
JaccardBin	18.89	**22.19**	
JaccardMin	**22.80**	17.55	
Lin	15.08	10.61	

Besides the general comments made so far, results depicted in tables 9 and 10 also let us observe the following phenomena:

As it was expected, binary metrics yieldied the same results with both baseline and log matrices. These two matrices are indeed identical if they are represented in boolean terms.

Among the non-binary metrics, the coefficient providing the best results using the SVD-reduced matrix is Cosine. However, in this context Cosine metric behaves better with the other approaches: baseline, log, and filter. Besides, City and Euclidean tend to behave better with both *S-svd* and *W-svd* than with the other three methods. This means that City and Euclidean are not suited at all to deal with sparse matrices. Concerning the remaining non-binary measures (DiceMin, JaccardMin, and Lin), their application to SVD-reduced matrix is not meaningful, since they only distinguish between shared and not shared word contexts.

It is worth noting that Jaccard and Dice metrics are equivalent. They provide the same scores in 7 out of 8 methods (there is only a small difference within *W-filter*). This is in accordance with the fact that Jaccard and Dice coefficients should always yield the same similarity rankings for any word [4]. Hereafter, we'll use the term "Dice-Jaccard".

Tables 9 and 10 also show that each method has its preferred similarity measure. The favorite coefficient for *S-log* and *S-filter* is Dice-JaccardMin, which achieves the best scores (50.25% and 50.22%, respectively) of all experiments. For the windowing techniques without SVD, i.e., *W-log*, *W-filter*, and *W-baseline*, the best measures are the binary ones, namely CosineBin and Dice-JaccardBin. Concerning the SVD-based methods, Cosine achieved the highest scores with both *W-svd(1000)* (25.53%) and *s-svd(1000)* (30.02%). Finally, Euclidean seems to be slightly better that City, which turned out to be the worst metric in our experiments. However, it is City-Block the only metric that improves results using SVD-based methods, that is, it performs better with dense matrices.

These results are in accordance with those obtained in [18] where analogous large-scale experiments were performed to acquire word similarity from a monolingual corpus.

5.3.3 Results of the English-Galician Corpus

Table 12 shows the F-scores obtained using the syntax-based strategy and three types of matrices (baseline, filter, and svd(300)) over the English-Galician comparable corpus. Results are analogous to those obtained in the previous syntax-based experiment:

the best coefficients are Dice-JaccardMin, but there are no significant differences with regard to binary measures. Concerning SVD dense matrices, Cosine is the best coefficient, even if its score is still far from the results achieved with sparse matrices. By contrast, the behaviour of City-Block and Euclidean is better with SVD dense matrices than with the sparse ones (baseline and filter).

Notice that F-scores are much worse than in the previous experiments. There are, at least, two reasons: First, the language pair Spanish-Galician is more closely related than English-Galician. Second, the English-Galician dictionary used to built the list of seed words is much smaller.

5.3.4 Corpus Partition

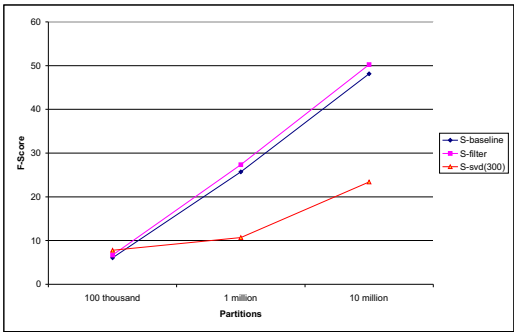

Fig. 2 F-Score of *S-baseline*, *S-filter*, and *S-svd* over 3 corpus partitions

Figure 2 additionally depicts how or whether three syntactic-based methods (*S-baseline*, *S-filter*, and *S-svd(300)*) benefit from a larger corpus with sizes from 100,000 to 10 million word tokens, taken from the Spanish-Galician corpus (the same described above in subsection 5.1). Figure 2 only shows one similarity score (the best one) by method: Dice-JaccardBin with *S-baseline*, Dice-JaccardMin with *S-filter*, and Cosine with *S-svd*. Notice that *S-svd* starts with the best score, but it improves very slowly as the corpus grows. This seems to mean that SVD-based strategy works well on small corpora but the precision gain with larger input is actually very poor compared to the other two approaches. This could explain some of the optimistic findings reported in the literature (see introduction above).

One possible reason is the following: a small corpus provides very few direct context-word co-occurrences, which are the only source of information required by *S-baseline* and *S-filter*. By constrast, the amount of information available for *S-svd* is considerably larger since it uses factor analysis and multi-dimensional scaling to generate more abstract word spaces with higher-order co-occurrences. So, SVD-methods work slightly better with small corpora because they are able to capture more information

before computing word similarity. As the corpus size grows, the number of direct co-occurrences also grows, and then the source of information required by *S-baseline* and *S-filter* becomes more reliable for similarity extraction. By contrast, such a reliable information (only direct co-occurrences) is transformed by SVD into an abstract word space with latent information that becomes hard to interpret, not only by humans but also by the most straightforward similarity coefficients.

5.3.5 Third-Or-More-Order Similarities

As it has been said above, SVD-methods should theoretically be able to find not only second-order similarity but also higher-order relatedness. A high-order similarity is based on comparing words that do not co-occur in the corpus with the same words (or lexical-syntactic contexts) but with words that can be related to the two compared words through further indirect co-occurrences. Using the smallest corpus size, we found that almost 10% (31 out of 338) correct translations proposed by the *S-svd* strategy are bilingual pairs of words that did not co-occur in the corpus with any common context. This means that the generalization performed by SVD does indeed find some latent semantics since it enables finding third-or-more-order similarities. However, the benefits of such a gain come at a dramatic decrease in precision in larger corpora (from 50.22% precision with the *S-filter* algorithm to 30.02% precision with *S-svd*) of second-order similarity, whose contribution for the overall similarity is crucial. Additionally, using the same simple baseline comparison methods to compare words based on their similar words (instead of their co-occurrences) would also yield higher-order relatedness as shown in [3].

6 Conclusions

While the main goal was to find the best method that computes translations, one of the main contributions of this paper is to compare SVD-methods with other models under controlled circumstances by means of a large-scale evaluation, and by taking a large bilingual dictionary as gold standard.

The results of the experiments leave no doubt that at least for the task of extracting translation equivalents from comparable corpora, SVD-based methods are both computationally more costly and effectively less precise in their results. On the one hand, given that the sparse matrix reduced with SVD produces more non-zero values than those contained in the original matrix, SVD-methods turn out to compute similarity in a much more time-consuming manner than baseline strategies. On the other hand, latent semantic information as a result of factorization by SVD, such as high-order co-occurrences, does not help to improve the task of extracting candidate translations. Especially for larger corpora, the precision gain is far less than expected if compared with baseline strategies.

It is also clear that the arguments about computational efficiency can be transferred to all other tasks involving any kind of SVD procedure. While it is not as clear whether the poor precision values would also be observed in other tasks, such as LSA or plain similar word computations, these experiments at least give a strong hypothesis about how SVD-based methods compare with baseline methods there.

However, it is also obvious that the results reported are not fully conclusive. In order to reach a more reliable degree of certainty, it is required to perform further tests

with both monolingual and bilingual copora. Nevertheless, the results reported in this paper provide evidence for the following statement: In the normal vector space where initially words constitute dimensions it is really difficult to overcome baseline methods to extract semantic information. There is no evidence yet that elaborate extracting techniques, such as those relying on SVD, are better than those based on the original co-occurrence matrix and boolean similarity coefficients.

Acknowledgements This work has been supported by the Galician Government (projects with reference: PGIDIT07PXIB204015PR and 2008/101), and by the Natural Language Engineering Department at the University of Leipzig.

References

1. Carme Armentano-Oller, Rafael C. Carrasco, Antonio M. Corbí-Bellot, Mikel L. Forcada, Mireia Ginestí-Rosell, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez, and Miriam A. Scalco. Open-source Portuguese-Spanish machine translation. In *Lecture Notes in Computer Science, 3960*, pages 50–59, 2006.
2. Marco Baroni and Alessandro Lenci. Concepts and Properties in Word Space. *Italian Journal of Linguistics*, 20(1), 2008.
3. C. Biemann, S. Bordag, and U. Quasthoff. Automatic Acquisition of Paradigmatic Relations using Iterated Co-occurrences. In *LREC2004*, Lisbon, Portugal, 2004.
4. Stefan Bordag. *Elements of Knowledge-free and Unsupervised Lexicon Acquisition*. PhD thesis, University of Leipzig, 2007.
5. Stefan Bordag. A Comparison of Co-occurrence and Similarity Measures as Simulations of Context. In *9th CICLing*, pages 52–63, 2008.
6. R. Bradford. An Empirical Study of Required Dimensionality for Large-scale Latent Semantic Indexing Applications. In *17th ACM Conference on Information and Knowledge Management*, pages 153–162, Napa Valley, California, 2008.
7. R. Budiu and P. Pirolli. Navigation in degree-of-interest trees. In *Advance Visual Interface Conference*, 2006.
8. X. Carreras, I. Chao, L. Padró, and M. Padró. An Open-Source Suite of Language Analyzers. In *4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, 2004.
9. Y-C. Chiao and P. Zweigenbaum. Looking for candidate translational equivalents in specialized, comparable corpora. In *19th COLING'02*, 2002.
10. James R. Curran and Marc Moens. Improvements in Automatic Thesaurus Extraction. In *ACL Workshop on Unsupervised Lexical Acquisition*, pages 59–66, Philadelphia, 2002.
11. S. Deerwester, S.T. Dumais, G.W. Furmas, T.K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
12. H. Dejean, E. Gaussier, and F. Sadat. Bilingual terminology extraction: an approach based on a multilingual thesaurus applicable to comparable corpora. In *COLING 2002*, Tapei, Taiwan, 2002.
13. George A. Ferguson and Yoshio Takane. *Statistical Analysis in Psychology and Education*. McGraw-Hill Ryerson Limited, Montreal, Quebec, 2005.
14. Pascale Fung and Kathleen McKeown. Finding terminology translation from non-parallel corpora. In *5th Annual Workshop on Very Large Corpora*, pages 192–202, Hong Kong, 1997.
15. Pascale Fung and Lo Yuen Yee. An IR Approach for Translating New Words from Nonparallel, Comparable Texts. In *Coling'98*, pages 414–420, Montreal, Canada, 1998.
16. Pablo Gamallo. Learning Bilingual Lexicons from Comparable English and Spanish Corpora. In *Machine Translation SUMMIT XI*, Copenhagen, Denmark, 2007.
17. Pablo Gamallo. Evaluating Two Different Methods for the Task of Extracting Bilingual Lexicons from Comparable Corpora. In *LREC 2008 Workshop on Comparable Corpora*, pages 19–26, Marrakech, Marroco, 2008.
18. Pablo Gamallo. Comparing different properties involved in word similarity extraction. In *14th Portuguese Conference on Artificial Intelligence (EPIA'09), LNCS, Vol. 5816*, pages 634–645, Aveiro, Portugal, 2009. Springer-Verlag.

19. Pablo Gamallo, Alexandre Agustini, and Gabriel Lopes. Clustering Syntactic Positions with Similar Semantic Requirements. *Computational Linguistics*, 31(1):107–146, 2005.
20. Genevieve Gorrel. Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing. In *EACL 2005*, 2005.
21. Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, USA, 1994.
22. Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, Berkeley, California, 1999.
23. Thomas Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42(1-2):177–196, 2001.
24. Michael P. Holmes, Alexander G. Gray, and Charles Lee Isbell Jr. QUIC-SVD: Fast SVD Using Cosine Trees. In *NIPS-2008*, pages 673–680, 2008.
25. Hiroyuki Kaji. Extracting Translation Equivalents from Bilingual Comparable Corpora. In *IEICE Transactions 88-D(2)*, pages 313–323, 2005.
26. Hiroyuki Kaji and Toshiko Aizono. Extracting Word Correspondences from Bilingual Corpora Based on Word Co-occurrence Information. In *16th Conference on Computational Linguistics (Coling'96)*, pages 23–28, Copenhagen, Danmark, 1996.
27. T.K. Landauer and S.T. Dumais. A solution to Plato's problem: The Latent Semantic Analysis theory of acquision, induction and representation of knowledge. *Psychological Review*, 10(2):211–240, 1997.
28. B. Lemaire and G. Denhière. Effects of High-Order Co-occurrences on Word Semantic Similarity. *Current Psychology Letters*, 18(1), 2006.
29. Esther Levin, Mehrbod Sharifi, and Jerry T. Ball. Evaluation of Utility of LSA for Word Sense Discrimination. In *HLT-NAACL*, 2006.
30. Dekang Lin. Automatic Retrieval and Clustering of Similar Words. In *COLING-ACL'98*, Montreal, 1998.
31. Hiroshi Masuichi, Raymond Flournoy, Stefan Kaufmann, , and Stanley Peters. Query translation method for cross language information retrieval. In *Proceedings of the Workshop on Machine Translation for Cross Language Information Retrieval, MT Summit VII*, pages 30–34, Singapore, 1999.
32. I. Matveeva, G. Levow, A. Farahat, and C. Royer. Terms representation with generalized latent semantic analysis. In *RANLP-2005*, 2005.
33. T. Pedersen and A. Kulkarni. Discovering Identities in Web Contexts with Unsupervised Clustering. In *IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data*, pages 23–30, Hyderabad, India, 2007.
34. R. Price and A. Zukas. Application of latent semantic indexing to processing of noisy text,. In *Intelligence and Security Informatics, LNCS 3495*, pages 602–603, 2005.
35. James Pustejovsky. *The Generative Lexicon*. MIT Press, Cambridge, 1995.
36. Reinhard Rapp. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *ACL'99*, pages 519–526, 1999.
37. Reinhard Rapp. Word sense discovery based on sense descriptor dissimilarity. In *9th Machine Translation Summit*, 2003.
38. Reinhard Rapp. A freely available automatically generated thesaurus of related words. In *LREC-2004*, pages 395–398, Lisbon, Portugal, 2004.
39. X. Saralegui, I. San Vicente, and A. Gurrutxaga. Automatic generation of bilingual lexicons from comparable corpora in a popular science domain. In *LREC 2008 Workshop on Building and Using Comparable Corpora*, 2008.
40. Hinrich Schütze. Dimensions of meaning. In *Proceedings of Supercomputing-92*, pages 787–796, Minneapolis, MN, 1992.
41. Hinrich Schütze. Ambiguity resolution in langugage learning. In *CSLI Publications*, Standford, CA, 1997.
42. Hinrich Schütze. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–124, 1998.
43. E. Terra and C.L. Clarke. Frequency estimates for statistical word similarity measures. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL'03)*, pages 165–172, NJ, USA, 2003.
44. P. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *12th European Conference of Machine Learning*, pages 491–502, 2001.

45. Yinghui Xu Kyoji Umemura. Very low-dimensional latent semantic indexing for local query regions. In *Annual Meeting of the ACL archive Proceedings of the sixth international workshop on Information retrieval with Asian languages*, pages 84–91, Saporo, Japan, 2003.
46. Lonneke van der Plas and Gosse Bouma. Syntactic Contexts for Finding Semantically Related Words. In *Meeting of Computational Linguistics in the Netherlands (CLIN2004)*, 2004.
47. Jinqiao Wang, Lingyu Duan, Lei Xu, Hanqing Lu, and Jesse S. Jin. TV a,d video categorization with probabilistic latent concept learning. In *Workshop on multimedia information retrieval*, pages 24–29, Augsburg, Bavaria, Germany, 2007.
48. P. Wiemer-Hastings, K. Wiemer-Hastings, and A. Graesser. Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. In S. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education*, pages 535–542, Amsterdam, 1999. IOS Press.
49. Yueting Zhuang, Weiming Lu, and Jiangqin Wu. Latent Style Model: Discovering writing styles for calligraphy works. *Journal of Visual Communication and Image Representation*, 20(2):84–96, 2009.