SVO triple based Latent Semantic Analysis for recognising textual entailment

Gaston Burek **Christian Pietsch** **Anne De Roeck**

Centre for Research in Computing
The Open University
Walton Hall, Milton Keynes, MK7 6AA, UK
{g.g.burek,c.pietsch,a.deroeck}@open.ac.uk

Abstract

Latent Semantic Analysis has only recently been applied to textual entailment recognition. However, these efforts have suffered from inadequate bag of words vector representations. Our prototype implementation for the Third Recognising Textual Entailment Challenge (RTE-3) improves the approach by applying it to vector representations that contain semi-structured representations of words. It uses variable size n-grams of word stems to model independently verbs, subjects and objects displayed in textual statements. The system performance shows positive results and provides insights about how to improve them further.

1 Introduction

The Third Recognising Textual Entailment Challenge (RTE-3) task consists in developing a system for automatically determining whether or not a hypothesis (H) can be inferred from a text (T), which could be up to a paragraph long.

Our entry to the RTE-3 challenge is a system that takes advantage of Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997). This numerical method for reducing noise generated by word choices within texts is extensively used for document indexing and word sense disambiguation. Recently, there have also been efforts to use techniques from LSA to recognise textual entailment (Clarke, 2006; de Marneffe et al., 2006). However, we argue that these efforts (like most LSA approaches in the past) suffer from an inadequate vector representation for textual contexts as bags of words. In contrast, we have applied LSA to vector representations of semi-structured text. Our representation takes into account the grammatical role (i.e. subject, verb or object) a word occurs in.

Within this system report, we describe and discuss our methodology in section 2, our current implementation in section 3, and system results in section 4. We conclude in section 5 with a discussion of the results obtained and with the presentation of possible steps to improve our system's performance.

2 Methodology for detecting Textual Entailment

2.1 Textual entailment formalisation

Our approach addresses the problem of the semantic gap that exists between low level linguistic entities (words) and concepts. Concepts can be described by means of predicate-argument structures or by a set of alternative natural language realisations. In this work we use terminology co-occurrence information to identify when different spans of text have common semantic content even if they do not share vocabulary. To achieve this, we use variable size n-grams to independently model subject, verb and object, and capture semantics derived from grammatical structure. In order to detect textual entailment we measure the semantic similarity between n-grams in each T–H pair.

2.2 Using n-grams to align SVOs

To align subjects, verbs and objects within H and T, we build the set of all n-grams for T, and do the same

Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 113–118,
Prague, June 2007.

for H. Section 3.5 describes this process in more detail.

2.3 Deriving word senses with Latent Semantic Analysis

Our approach is based on the assumption that a word sense can be derived from the textual contexts in which that words occurs. This assumption was formalised in the Distributional Hypothesis (Harris, 1954).

We implemented a vector space model (Salton et al., 1975) to capture word semantics from linguistic (i.e. grammatical role) and contextual (i.e. frequency) information about each word. To avoid high matrix sparsity our vector space model uses second order co-occurrence (Widdows, 2004, p. 174).

We assumed that the corpus we generated the vector space model from has a probabilistic word distribution that is characterised by a number of semantic dimensions. The LSA literature seems to agree that optimal number of dimensions is somewhere between two hundred and one thousand depending on corpus and domain. As specified by LSA we applied Singular Value Decomposition (SVD) (Berry, 1992) to identify the characteristic semantic dimensions.

The resulting model is a lower dimensional projection of the original model that captures indirect associations between the vectors in the original model. SVD reduces the noise in word categorisations by producing the best approximation to the original vector space model.

3 Implementation

3.1 Development data set

The development data set consists of eight hundred T–H pairs, half of them positive. By positive pair we mean a T–H pair in which T entails H. All other pairs we call negative. Each T–H pair belongs to a particular sub-task. Those sub-tasks are Information Extraction (IE), Information Retrieval (IR), Question Answering (QA) and Summarisation (SUM). In the current prototype, we ignored annotations about sub-tasks.

3.2 Corpus analysis

3.2.1 Corpora used

The only knowledge source we used in our implementation was a parsed newswire corpus (Reuters News Corpus) (Lewis et al., 2004). To derive contextual information about the meaning of words constituting the SVOs, we analysed the Reuters corpus as explained below.

3.2.2 SVO triple extraction

For parsing the corpus, we used Minipar[1] because of its speed and because its simple dependency triple output format (`-t` option) contains word stems and the grammatical relations between them. A simple AWK script was used to convert the parse results into Prolog facts, one file for each sentence. A straightforward Prolog program then identified SVOs in each of these fact files, appending them to one big structured text file.

Our algorithm currently recognises intransitive, transitive, ditransitive, and predicative clauses. Intransitive clauses are encoded as SVOs with an empty object slot. Transitive clauses result in a fully instantiated SVOs. Ditransitive clauses are encoded as two different SVOs: the first containing subject, verb and direct object; the second triple containing the subject again, an empty verb slot, and the indirect object. Predicatives (e.g. "somebody is something") are encoded just like transitive clauses.

In this first prototype, we only used one word (which could be a multi-word expression) for subject, verb and object slot respectively. We realise that this approach ignores much information, but given a large corpus, it might not be detrimental to be selective.

3.2.3 SVO Stemming and labeling

To reduce the dimensionality of our vector space model we stem the SVOs using Snowball[2]. Then, we calculate how many times stems co-occur as subject, verb or object with another stem within the same SVO instance.

[1] Minipar can be downloaded from http://www.cs.ualberta.ca/~lindek/minipar.htm. It is based on Principar, which is described in Lin (1994).

[2] Snowball is freely available from http://snowball.tartarus.org/. The English version is based on the original Porter Stemmer (Porter, 1980).

To keep track of the grammatical role (i.e. subject, verb, object) of the words we stem them and label the stems with the corresponding role.

3.3 Building vector spaces to represent stem semantics

From the corpus, we built a model $(\mathcal{S}, \mathcal{V}, \mathcal{O})$ of the English (news) language consisting of three matrices: $\mathcal{S}$ for subjects, $\mathcal{V}$ for verbs, and $\mathcal{O}$ for objects.

We built the three stem-to-stem matrices from labeled stem co-occurrences within the extracted triples. The entries to the matrices are the frequencies of the co-occurrence of each labeled stem with itself or with another labeled stem. In our current prototype, due to technical restrictions explained in Section 3.4, each matrix has 1000 rows and 5000 columns.

Columns of matrix $\mathcal{S}$ contain entries for stems labeled as subject, columns of matrix $\mathcal{V}$ contain entries for stems labeled as verb, and columns of matrix $\mathcal{O}$ contain entries for stems labeled as object. The frequency entries of each matrix correspond to the set of identically labeled stems with the highest frequency.

Rows of the three matrices contain entries corresponding to the same set of labeled stems. Those labeled stems are the ones with the highest frequency in the set of all labeled stems. Of these, 347 stems are labeled as subject, 356 are labeled as verb, and 297 are labeled as object. Each row entry is the frequency of co-occurrence of two labeled stems within the same triple.

Finally, each column entry is divided by the number of times the labeled stem associated with that column occurs within all triples.

3.4 Calculating the singular value decomposition

We calculated the Singular Value Decompositions (SVDs) for $\mathcal{S}$, $\mathcal{V}$ and $\mathcal{O}$. Each SVD of a matrix $\mathcal{A}$ is defined as a product of three matrices:

$$\mathcal{A} = U \times S \times V' \tag{1}$$

SVD is a standard matrix operation which is supported by many programming libraries and computer algebra applications. The problem is that only very few can handle the large matrices required for real-world LSA. It is easy to see that the memory required for representing a full matrix of 64 bit floating point values can easily exceed what current hardware offers. Fortunately, our matrices are sparse, so a library with sparse matrix support should be able to cope. Unfortunately, these are hard to find outside the Fortran world. We failed to find any Java library that can perform SVD on sparse matrices.[3] We finally decided to use SVDLIBC, a modernised version of SVDPACKC using only the LAS2 algorithm. In pre-tests with a matrix derived from a different text corpus (18371 rows $\times$ 3469 columns, density 0.73%), it completed the SVD task within ten minutes on typical current hardware. However, when we try to use it for this task on a matrix $\mathcal{S}$ of dimension 5000 $\times$ 5000 (density 1.4%), SVDLIBC did not terminate[4]. In theory, there is a Singular Value Decomposition for every given matrix, so we assume this is an implementation flaw in either SVDLIBC or GCC. With no time left to try Fortran alternatives, we resorted to reducing the size of our three matrices to 1000 $\times$ 5000, thus losing much information in our language model.

3.5 Looking for evidence of H in T using variable size n-grams

3.5.1 Building variable size n-grams

Our Minipar triple extraction algorithm is not able to handle SVOs that are embedded within other SVOs (as e.g. in "Our children play a new game that involves three teams competing for a ball."). Therefore, in order to determine if SVOs displayed in H are semantically similar to any of those displayed in T, we generate all n-grams of all lengths for each T and H: one set for subjects, one for verbs and another one for objects.

Example: "The boy played tennis."

Derived n-grams: the; the boy; the boy played; the boy played tennis; boy; boy played; boy played

[3] The popular JAMA library and the related Jampack library have no sparse matrix support at all. MTJ and Colt do support sparse matrices but cannot perform SVD on them without first converting them to full matrices.

[4] We tried various hardware / operating system / compiler combinations. On Linux systems, SVDLIBC would abort after about 15 minutes with an error message "imtqlb failed to converge". On Solaris and Mac OS X systems, the process would not terminate within several days.

tennis; played; played tennis; tennis.

We use n-grams to model subjects, verbs and objects of SVOs within T and H.

3.5.2 How to compare n-grams

We generate three vector representations for each n-gram. To do this, we add up columns from the Reuters Corpus derived matrices. To build the first vector representation, we use the $\mathcal{S}$ matrix, to build the second vector we use the $\mathcal{V}$ matrix, and to build the third vector we use the $\mathcal{O}$ matrix. Each of the three representations is the result of adding the columns corresponding to each stem within the n-gram.

To calculate the semantic similarity between n-grams, we fold the three vector representations of each n-gram into one of the dimensionally reduced matrices $\mathcal{S}_{200}$, $\mathcal{V}_{200}$ or $\mathcal{O}_{200}$. Vector representation originating from the $\mathcal{S}$ matrix are folded into $\mathcal{S}_{200}$. We proceed analogously for vector representations originating from $\mathcal{V}_{200}$ and $\mathcal{O}_{200}$. We apply equation 2 to fold vectors from G^r where $r \in \{\mathcal{S}, \mathcal{V}, \mathcal{O}\}$. G is a matrix which consists of all the vector representations of all the n-grams modeling T or H. S^r_{200} and U^r_{200} are SVD results reduced to 200 dimensions.

$$G^{r\prime} \times U^r_{200} \times (S^r_{200})^{-1} = G^r_{200} \tag{2}$$

For each T–H pair we calculate the dot product between the G matrices for H and T as expressed in equation 3

$$^{text}G^r_{200} \times ^{hypothesis} G^{r\prime}_{200} = O^r \tag{3}$$

The resulting matrix O^r contains the dot product similarity between all pairs of n-grams within the same set. Finally, for each T–H pair we obtain three similarity values s, v, o by selecting the entry of O^r with the highest value.

3.5.3 Scoring

Now we have calculated almost everything we need to venture a guess about textual entailment.

For each T–H pair, we have three scores s, v, o for for subject, verb and object slot respectively. The remaining task is to combine them in a meaningful way in order to make a decision. This requires some amount of training which in the current prototype is as simple as computing six average values: $\bar{s}_p$, $\bar{v}_p$, $\bar{o}_p$ are the average scores of subject, verb and object slots over those T–H pairs for which textual entailment is known to hold. Conversely, $\bar{s}_n$, $\bar{v}_n$, $\bar{o}_n$ are the averages for those pairs that do not stand in a textual entailment relation. The (rounded) values were determined are shown in table 1.

	$\bar{s}$	$\bar{v}$	$\bar{o}$
positive	0.244	$5.05 \cdot 10^{-7}$	0.323
negative	0.196	$4.76 \cdot 10^{-7}$	0.277

Table 1: Values computed for $\bar{s}_p, \bar{v}_p, \bar{o}_p, \bar{s}_n, \bar{v}_n, \bar{o}_n$

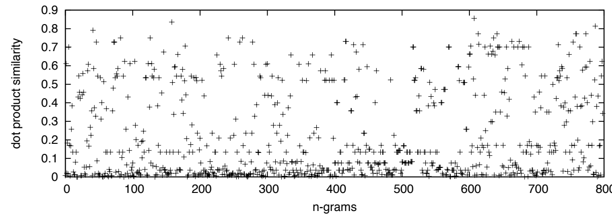

Figure 1: Subject similarities $s = \max O^{\mathcal{S}}$ for all H–T pairs

Note that the average values for non-entailment are always lower than the average values for entailment, which indicates that our system indeed tends to discriminate correctly between these cases.

The very low values for the verb similarities (figure 3) compared to subject similarities (figure 1) and object similarities (figure 2) remind us that before we can combine slot scores, they should be scaled to a comparable level. This is achieved by dividing each slot score by its corresponding average. Ignoring the difference between positive and negative pairs for a moment, the basic idea of our scoring algorithm is to use the following threshold:

$$\frac{s}{\bar{s}} + \frac{v}{\bar{v}} + \frac{o}{\bar{o}} = 3 \tag{4}$$

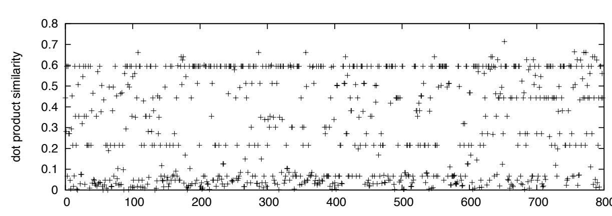

Figure 2: Object similarities $o = \max O^{\mathcal{O}}$ for all H–T pairs

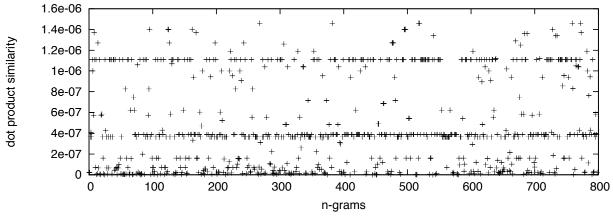

Figure 3: Verb similarities $v = \max O^{\mathcal{V}}$ for all H–T pairs

At this point we observed that scaling the verb similarities so much seemed to make results worse. It seems to be necessary to introduce weights:

$$\sigma \frac{s}{\bar{s}} + \phi \frac{v}{\bar{v}} + \omega \frac{o}{\bar{o}} = \sigma + \phi + \omega \tag{5}$$

Without loss of generality, we may assume $\phi = 1$:

$$\sigma \frac{s}{\bar{s}} + \frac{v}{\bar{v}} + \omega \frac{o}{\bar{o}} = \sigma + 1 + \omega \tag{6}$$

The complete scoring formula with both positive and negative scores is shown below. We assumed that the weights σ and ω are the same in the positive and in the negative case, so $\sigma = \sigma_p = \sigma_n$ and $\omega = \omega_p = \omega_n$.

$$\sigma \frac{s}{\bar{s}_p} + \frac{v}{\bar{v}_p} + \omega \frac{o}{\bar{o}_p} + \sigma \frac{s}{\bar{s}_n} + \frac{v}{\bar{v}_n} + \omega \frac{o}{\bar{o}_n} = 2(\sigma + 1 + \omega) \tag{7}$$

At this point, some machine learning over the development data set should be performed in order to determine optimal values for σ and ω. For lack of time, we simply performed a dozen or so of test runs and finally set $\sigma = \omega = 3$.

Our entailment threshold is thus simplified:

$$3 \frac{s}{\bar{s}_p} + \frac{v}{\bar{v}_p} + 3 \frac{o}{\bar{o}_p} + 3 \frac{s}{\bar{s}_n} + \frac{v}{\bar{v}_n} + 3 \frac{o}{\bar{o}_n} = q \tag{8}$$

If $q > 14$, our prototype predicts textual entailment. Otherwise, it predicts non-entailment.

4 Results

Using the scoring function described in section 3.5.3, our system achieved an overall accuracy of 0.5638 on the development dataset. Table 2 shows results for the system run on the test dataset. On this unseen dataset, the overall accuracy decreased only slightly to 0.5500. We take this as a strong indication that the thresholds we derived from the development dataset work well on other comparable input. Results show that our system has performed significantly above the 0.5 baseline that would result from a random decision.

	all	IE	IR	QA	SUM
accuracy	0.5500	0.4950	0.5750	0.5550	0.5750
av. prec.	0.5514	0.4929	0.5108	0.5842	0.6104

Table 2: Results on the test set

As shown in section 3.5.3, the values in the three similarity plots (see figures 1, 2 and 3) obtained with the development set seem to be scattered around the means. Therefore it seems that the threshold values used to the decide whether or not T entails H do not fully reflect the semantics underlying textual entailment.

The nature of the SVD calculations do not allow us directly to observe the performance of the variable size n-grams in independently aligning subject, verb and objects from T and from H. Nevertheless we can infer from figures 1, 2 and 3 that many of the values shown seem to be repeated. These value configurations can be observed in the three horizontal lines. These lines better visible in figures 2 and 3 are the effect of (a) many empty vectors resulting from the rather low number of stems represented by columns in our Reuters-derived matrices $\mathcal{S}$, $\mathcal{V}$ and $\mathcal{O}$, and (b) the effect of folding the n-gram vector representations into reduced matrices with two hundred dimensions.

5 Conclusion

Even though our system was developed from scratch in a very short period of time, it has already outperformed other LSA-based approaches to recognising textual entailment (Clarke, 2006), showing that it is both feasible and desirable to move away from a bag-of-words semantic representation to a semi-structured (here, SVO) semantic representation even when using LSA techniques.

Our system displays several shortcomings and limitations owing to its immature implementation state. These will be addressed in future work, and we are confident that without changing its theoretical basis, this will improve performance dramatically. Envisaged changes include:

- using larger matrices as input to SVD
- using the complete Reuters corpus, and adding Wikinews texts
- performing corpus look-up for unknown words
- extracting larger chunks from S and O slots
- using advanced data analysis and machine learning techniques to improve our scoring function

In addition, our approach currently does not take into consideration the directionality of the entailment relationship between the two text fragments. In cases where T1 entails T2 but T2 does not entail T1, our approach will treat (T1, T2) and (T2, T1) as the same pair. We expect to correct this misrepresentation by evaluating the degree of specificity of words composing the SVOs in asymmetric entailment relationships where the first text fragment is more general than the second one. For that purpose, one can use term frequencies as an indicator of specificity (Spärck Jones, 1972).

Obviously, system performance could be further improved by taking a hybrid approach as e.g. in de Marneffe et al. (2006), but we find it more instructive to take our pure LSA approach to its limits first.

6 Acknowledgements

We are grateful to Prof. Donia Scott, head of the Natural Language Generation group within the Centre for Research in Computing of the Open University, who made us aware of the RTE-3 Challenge and encouraged us to participate.

References

[Berry1992] M. W. Berry. 1992. Large-scale sparse singular value computations. *The International Journal of Supercomputer Applications*, 6(1):13–49, Spring.

[Clarke2006] Daoud Clarke. 2006. Meaning as context and subsequence analysis for entailment. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

[de Marneffe et al.2006] Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy.

[Harris1954] Zelig S. Harris. 1954. Distributional structure. *WORD*, 10:146–162. Reprinted in J. Fodor and J. Katz, The structure of language: Readings in the philosophy of language, pp. 33–49, Prentice-Hall, 1964.

[Landauer and Dumais1997] T. K. Landauer and S. T. Dumais. 1997. A solution to Plato's Problem. The Latent Semantic Analysis theory of the acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

[Lewis et al.2004] D. D. Lewis, Y. Yang, T. Rose, and F. Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

[Lin1994] Dekang Lin. 1994. PRINCIPAR – an efficient, broad-coverage, principle-based parser. In *Proc. COLING-94*, pages 42–488, Kyoto, Japan.

[Porter1980] M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Salton et al.1975] G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[Spärck Jones1972] Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21. Reprinted 2004 in 60(5):493–502 and in Spärck Jones (1988).

[Spärck Jones1988] Karen Spärck Jones. 1988. A statistical interpretation of term specificity and its application in retrieval. *Document retrieval systems*, pages 132–142.

[Widdows2004] Dominic Widdows. 2004. *Geometry and Meaning*. Number 172 in CSLI Lecture Notes. University of Chicago Press.