Span-based Normalized Edit Distance Applied to Unsupervised Parsing

Simon Dennis (Simon.Dennis@adelaide.edu.au)
Department of Psychology; University of Adelaide
Adelaide, SA 5005 Australia

William Oliver (oliver@colorado.edu)
Institute of Cognitive Science; University of Colorado
Boulder, CO 80301 USA

Abstract

We present a span-based version of the normalized edit distance measure (Marzal & Vidal, 1993), which is more appropriate for linguistic tasks and give an $O(n^2m^3)$ algorithm for its calculation. Span similarities used in the algorithm are derived by taking the cosines between the left vectors of a reduced singular value decomposition of a span by context matrix. To test the model, an exemplar-based approach is used to provide unsupervised parses of sentences from the Penn Treebank using nearest neighbour extraction based on a version of Locality Sensitive Hashing (Indyk & Motwani, 1998; Gionis, Indyk, & Motwani, 1999). Initial results indicate that the method provides parsing recall and precision equivalent to other unsupervised methods.

Introduction

The nativist/empiricist debate on the origin on language has been one of the longest and most hotly contested in the history of cognitive science (Pinker, 1994; Elman, 1999). On the one hand, languages are clearly learned at some level with a great many variations that differ in quite subtle ways. Furthermore, the difficulty in creating an explanation of how the genes might influence language development suggests that it is unlikely that our biological endowment has a direct influence (Elman, 1999). However, the fact that humans have a much more complex system of language than other primates, that there are similarities across the world's languages and that language acquisition takes similar paths in different cultures suggest a strong innate component (Pinker, 1994).

One key, if unstated, plank in the nativist case is that to this point no statistical learning procedure capable of capturing the syntax of a complex natural language has been devised (see Dennis, under review; Klein & Manning, 2001). While connectionist models have demonstrated an ability to solve restricted problems with toy corpora (Elman, 1991), issues such as systematicity and constituent formation and movement remain unresolved (Hadley, 1994) seriously undermining the empiricist position.

In addition, from a practical perspective the inability to create syntactic analyses in an unsupervised fashion makes the application of natural language processing systems in new domains tedious. Either one must hand specify appropriate rules or one must create annotated corpora on which to train systems. Both of these tasks are difficult and time consuming.

In this paper, we outline attempts to improve an exemplar-based model of unsupervised parsing proposed by Dennis (under review) using span-based normalized edit distance (SNED). We start by defining normalized edit distance and the span-based modification. Then we discuss how one can calculate the span similarities necessary to apply the method to sentences. Next we describe a version of Locality Sensitive Hashing (Indyk & Motwani, 1998; Gionis et al., 1999) adapted to work with part of speech strings. Finally, we present recall and precision parsing data on sentences drawn from the Penn Treebank (Marcus et al., 1993).

Definitions of Edit Distances

Edit Distance

Following the notation of Marzal and Vidal (1993), let Σ be a finite alphabet and Σ^* be the set of all finite-length strings over Σ. Let $X = X_1X_2...X_n$ be a string of Σ^*, where X_i is the ith symbol of X. We denote by $X_{i...j}$ the substring of X that includes the symbols from X_i to $X_j, 1 \leq i, j \leq n$. The length of such a string is $|X_{i...j}| = j - i + 1$. If $i > j, X_{i...j}$ is the null string $\lambda, |\lambda| = 0$.

An elementary edit operation is a pair $(a, b) \neq (\lambda, \lambda)$, where a and b are strings of length 0 or 1. The edit operations are termed insertions (λ, b), substitutions (a, b) and deletions (a, λ). An edit transformation of X into Y is a sequence S of elementary operations that transforms X into Y. Typically, edit operations have associated costs $\gamma(a, b)$. The function γ can be extended to edit transformations $S = S_1S_2...S_l$ by letting $\gamma(S) = \sum_{i=1}^{l} \gamma(S_i)$.

Given $X, Y \in \Sigma^*$ and S^*_{XY} the set of all edit transformations of X into Y, then the edit distance is defined as:

$$\delta(X, Y) = min\{\gamma(S)|S \in S^*_{XY}\} \quad (1)$$

Note that the triangle inequality is a consequence of this definition, so provided $\gamma(a, a) = 0, \gamma(a, b) > 0$, if $a \neq b$, and $\gamma(a, b) = \gamma(b, a) \forall a, b \in \Sigma \cup \{\lambda\}$, δ is a metric.

Dynamic programming algorithms of complexity $O(mn)$, where n is the length of X and m is the length of Y, exist to calculate edit distance and to retrieve minimal edit transformations (Wagner & Fischer, 1974).

Normalized Edit Distance

Let $L(S)$ be the length of a given edit transformation, then the normalized edit distance defined by Marzal and Vidal (1993) is:

$$d(X,Y) = min\{\gamma(S)/L(S)|S \in S^*_{XY}\} \qquad (2)$$

Note that normalized edit distance is not a metric. It can, however, be calculated in $O(nm^2)$ time using an algorithm provided by Marzal and Vidal (1993).

Marzal and Vidal (1993) also show that NED does not produce the same answer as postnormalizing, by finding the minimum path and dividing by its length. Furthermore, for a handwritten character recognition task, normalized edit distance produced better performance than either normal edit distance or post normalized edit distance.

Span-based Normalized Edit Distance (SNED)

While the normalized edit distance has proven successful in a number of tasks, when analyzing sentence structure we would prefer a version of the algorithm that aligns spans of symbols rather than individual symbols. Providing a definition of span-based edit distance involves relaxing the restriction in the normal algorithm, so that the strings a and b are drawn from Σ^*[1]. So, the edit operations become $(a,b) = (X_{i...j}, Y_{k...l})$ for $0 \leq i \leq j \leq n$, $0 \leq k \leq l \leq m$. Similarly, one can define span-based normalized edit distance in an analogous way. The appendix provides an algorithm capable of calculating the span-based normalized edit distance with time complexity $O(n^2m^3)$ and space complexity $O(nm^2)$.

Exemplar-based Parsing

The algorithm that we employ for parsing sentences is a version of the Syntagmatic Paradigmatic model (Dennis, in press, 2004, under review). In this model, sentence parsing involves aligning near neighbour exemplar sentences from memory with the target sentence. For instance, suppose we wish to parse the sentence "His dog was big." (see Figure 1).

We start by converting the sentence to a part of speech (POS) sequence - "PRP$ NN VBD JJ", where PRP$ = possessive pronoun, NN = noun, VBD = past tense verb and JJ = adjective. Next we identify near neighbour POS sequences from a large corpus and align each of these with the sentence (see Figure 2). In this case, we are using the 34,000 POS sequences that appeared at least twice in the first 350,000 sentences from the TASA corpus[2]. The number to the left of each is the corresponding span-based edit distance. Note that these alignments induce constituent structure. In this case, for example, we would propose that VBD-JJ should constitute one constituent and PRP$-NN another.

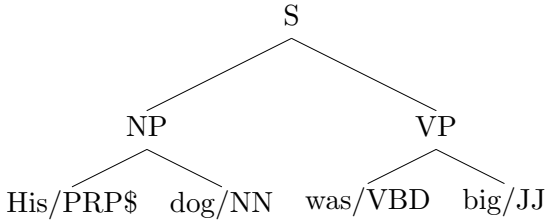

Figure 1. The correct parse of the sentence "His dog was big".

```
0.0000-|PRP$-NN-VBD-JJ|
       |PRP$-NN-VBD-JJ|
0.0011-|PRP$|NN|--VBD-JJ-|
       |PRP$|NN|MD-VB-VBN|
0.0017-|PRP$|NN|-VBD-JJ|
       |PRP$|NN|VBD-VBN|
0.0018-|PRP$-NN-|VBD|JJ|
       |DT-NN-NN|VBD|JJ|
```

Figure 2. Aligning multiple exemplars against a target sentence can approximate a traditional parse. PRP$ = personal pronoun, NN = Noun, VBD = Verb, past tense, JJ = Adjective, MD = Modal verb, VB = Verb, VBN = Verb, past participle, DT = Determiner.

While not constrained to be tree-like this structure may tend to correspond to a tree for many structurally unambiguous cases. To induce a tree the number of times each span of POS tags was identified by the model as a constituent was determined. The binary parse with the highest total constituent count is then chosen using the obvious dynamic programming algorithm (c.f. Klein & Manning, 2001). In the example, the nonsingleton spans have the following counts:

PRP$ NN VBD JJ	1
PRP$ NN VBD	0
NN VBD JJ	0
PRP$ NN	1
PRP$ VBD	0
VBD JJ	2

[1]For the current purposes, we assume that $a, b \neq \lambda$ although it would be useful to draw a and b from $\Sigma^* \cup \{\lambda\}$ as an alternative formulation.

[2]We thank the late Stephen Ivens and Touchstone Applied Science Associates (TASA) of Brewster, New York for providing this valuvlable resource.

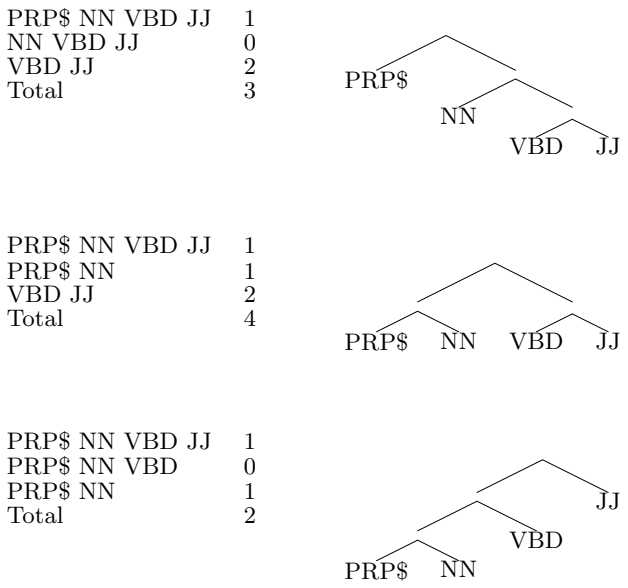

Figure 3. Possible binary parses of "His dog was big." and their associated counts. In this case, parse number two would be chosen.

Figure 3 shows the three possible binary parses of the example sentence and the counts of the associated spans. In this case, the second parse would be chosen as it has the highest total span count.

Calculating POS Span Costs

In order to apply the SNED algorithm one requires a γ function that indicates the cost of substituting one string of POS tags to another. To calculate substitution costs we combined ideas from the Context Constituent Model (CCM, Klein & Manning, 2001) and Latent Semantic Analysis (LSA, Landauer & Dumais, 1997). A span by context matrix was formed by taking all POS spans up to four tags in length and recording the number of times they appeared in each context consisting of the tags before and after the span. For instance, the example sentence, PRP$ NN VBD JJ, would generate the spans and contexts show in Figure 4.

Applying this procedure to the first 350,000 sentences of the TASA corpus generated 80,000 unique spans and 1300 unique contexts. Note that spans that tend to substitutable for each other will have similar sets of contexts. For instance, we might expect the pattern of contexts in which we find VBD JJ to be similar to the pattern in which we find MD VBD VBN as they are both verb phrases. Also, we column normalized as some contexts are much more frequent than others and the frequent ones (e.g. SS:EE) tend to be less informative.

The singular value decomposition (SVD) was then applied to factor the span by context matrix

Spans	Contexts
PRP$ NN VBD JJ	SS:EE
PRP$ NN VBD	SS:JJ
NN VBD JJ	PRP$:EE
PRP$ NN	SS:VBD
NN VBD	PRP$:JJ
VBD JJ	NN:EE
PRP$	SS:NN
NN	PRP$:VBD
VBD	NN:JJ
JJ	VBD:EE

Figure 4. Spans and associated contexts for the string PRP$ NN VBD JJ. Note SS and EE are tags indicating the start and end of the sentence, respectively.

M into three matrices:

$$M = UDV^T \tag{3}$$

where U is an orthonormal matrix where each row represents a span, D is a diagonal matrix of singular values and V is an orthonormal matrix where each row represents a context. To improve the signal to noise ratio all but the first 20 singular values were set to zero creating a *reduced* version of D which we will term $\hat{D}$. The distance between any two spans can then be calculated as follows:

$$\gamma(X_{i...j}, Y_{k...l}) = 1 - cos(U_x\hat{D}, U_y\hat{D}) \tag{4}$$

where U_x and U_y are the rows of U corresponding to span $X_{i...j}$ and $Y_{k...l}$, respectively. Figure 5 shows a multiple dimensional scaling solution for the vectors corresponding to the 60 most frequent spans. Note that there is clear similarity structure with spans representing sentences, verb phrases, $\overline{N}$ and $\overline{\overline{N}}$ structures well separated.

Finding Nearest Neighbors

A final issue to be resolved is how the algorithm selects nearest neighbour sequences to align. Given that there may be large numbers of potential sentences the performance of the nearest neighbour search will have a significant impact on the performance of the algorithm as a whole. In our case, it is sufficient to have a set of approximate nearest neighbours, so we use a version of Locality Sensitive Hashing (LSH, Indyk & Motwani, 1998; Gionis et al., 1999) adapted to work in Σ^* rather than in R^d as is typical.

The basic idea of LSH is to create multiple hash functions each of which is designed so that similar sequences are likely to collide. Finding the nearest neighbours of a target string involves applying

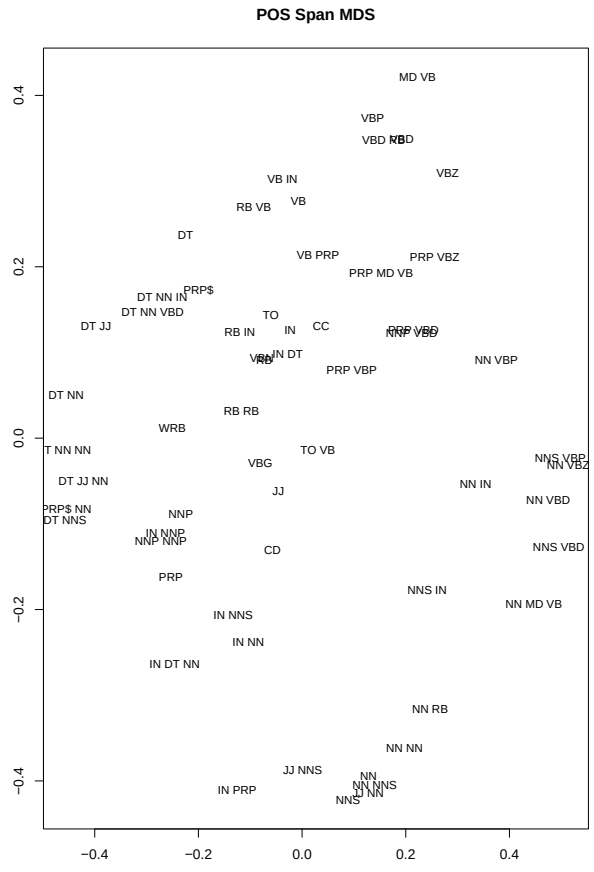

Figure 5. Multidimensional scaling solution for the vectors corresponding to the 60 most frequent span as derived from the SVD.

the hash functions to the new case and accumulating the strings that appear in the corresponding buckets. The SNED to each of these strings is then calculated to determine which are near neighbours.

To create the hash functions in Σ^* we create a set of rewrite rules that map one POS sequence to a simpler one. Different hash functions are created by permuting the rewrite rules. For example, suppose we have the sentence "Her little dolly felt sad.", which translates to PRP$ JJ NN VBD JJ, in our corpus and we wish to find the nearest neighbours of "His dog was big." (PRP$ NN VBD JJ). Further, suppose that we have the following rewrite rules JJ NN → NN, PRP$ NN → NN, DT NN → NN. Let:

h1 = [JJ NN → NN, PRP$ NN → NN, DT NN → NN]
h2 = [PRP$ NN → NN, DT NN → NN, JJ NN → NN]

Now for the two strings we get the following keys:

Target

h1(PRP$ NN VBD JJ) = NN VBD JJ
h2(PRP$ NN VBD JJ) = NN VBD JJ

Exemplar

h1(PRP$ JJ NN VBD JJ) = NN VBD JJ
h2(PRP$ JJ NN VBD JJ) = PRP$ NN VBD NN

Because the two strings have a hash key in common string two will be found when the system is queried with string one. In practice, locality sensitive hashing is fast and is not greatly affected by the size of the corpus. In our trials, we constructed a five hash system with hash functions containing 200 rewrite rules. On an AMD Opteron 800MHz system, 34,000 queries can be completed in 100 ms.

Evaluating the model

The procedure outlined above was applied to all of the sentences from the Wall Street Journal section of the Penn treebank (Marcus et al., 1993) that were of length 10 or less. To assess performance the parses produced by the model were compared against the gold standard parses provided by the treebank. Three measures were calculated:

- Unlabelled Recall: The mean proportion of constituents in the gold standard that the model proposed.
- Unlabelled Precision: The mean proportion of constituents in the models answer that appear in the gold standard.
- F_1: The harmonic mean of unlabelled recall and unlabelled precision.

Because the treebank provides parses that are not binary (in Chomsky normal form) but the procedure used makes this assumption it is not possible to achieve perfect performance. Klein and Manning (2001) calculated that the best possible F_1 measure that can be achieved is 87%.

Figure 6 shows the performance of the model against chance selection of trees and against three versions of the Constituent Context Model (CCM) proposed by Klein and Manning (2001). Clearly, all of these models are performing well above chance although all are still well below the theoretically achievable maximum of 87%.

A key issue in the performance of the model is the number of nearest neighbours that are returned by the locality sensitive hashing algorithm. Figure 7 shows the impact of restricting the analysis to the items that return nearest neighbour sets of different sizes. The performance of the model when applied to POS strings for which at leat 30 nearest neighbours were retieved is approximately the same level as the CCM model with an F_1 measure around 64%, but lags the CCM with tags model.

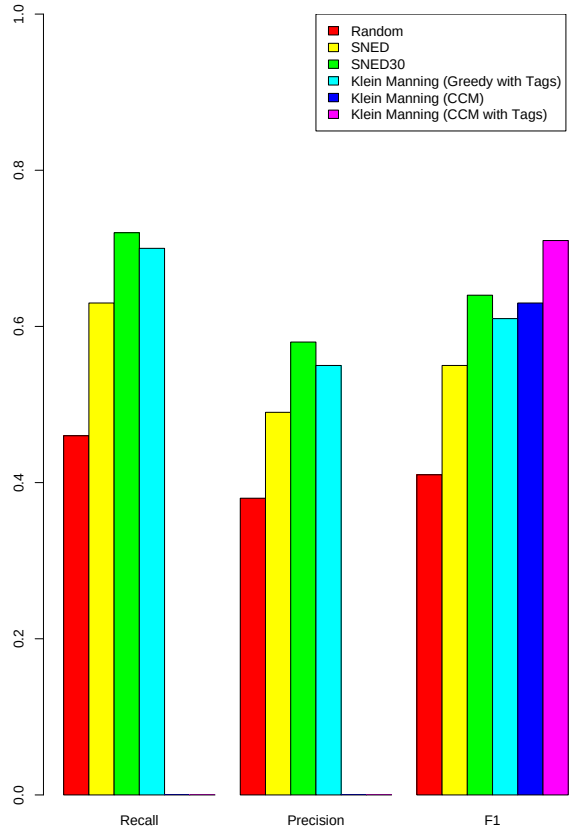

Figure 6. Results of Unsupervised Parsing Experiment.

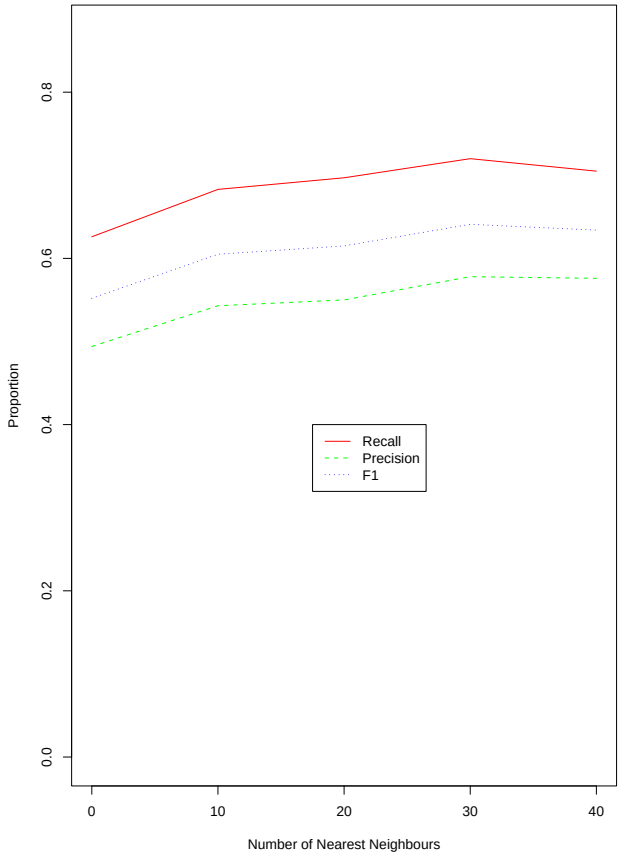

Figure 7. Results of Unsupervised Parsing Experiment.

Conclusions

A ket point in favour of the nativist account of language acquisition is the fact that no statistical mechanism capable of accounting for the human syntactic capability has yet been devised. While the model presented in this paper does not perform at a sufficient level to indicate that the problem has been solved, the results demonstrate that significant structure can be extracted from a natural corpus in an unsupervised fashion suggesting that the task may be able to be accomplished. Furthermore, the current results are preliminary. There are still a number of areas in which the model could be improved. These include the mechanism for calculating POS span similaities and the locality sensitive hashing method for identifying nearest neighbours.

Acknowledgments

We would like to acknowledge the many discussions that have influenced the current work. In particular, we would like to thank Walter Kintsch, Tom Landauer and Jose Quesada for helpful comments and suggestions. This research was supported by Australian Research Foundation Grant A00106012, U.S. National Science Foundation Grant EIA-0121201 and U.S. Department of Education Grant R305G020027.

References

Dennis, S. (2004). An unsupervised method for the extraction of propositional information from text. *Proceedings of the National Academy of Sciences*, *101*, 5206-5213.

Dennis, S. (in press). A memory-based theory of verbal cognition. *Cognitive Science.*

Dennis, S. (under review). Introducing word order in an LSA framework. In *Latent Semantic Analysis: A road to meaning.*

Elman, J. L. (1991). Distributed representations, simple recurrent networks and grammatical structure. *Machine Learning*, *7*, 195-225.

Elman, J. L. (1999). The origins of language: A conspiracy theory. In B. Mc Whinney (Ed.), *The emergence of language.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. In *The VLDB journal* (p. 518-529).

Hadley, R. F. (1994). Systematicity in connectionist language learning. *Mind and language*, *9*(3), 247-272.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In (pp. 604–613).

Klein, D., & Manning, C. D. (2001). Distributional phrase structure induction. In W. Daelemans & R. Zajac (Eds.), *Connl-2001* (p. 113-120). Toulouse, France.

Landauer, T. K., & Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, *104*, 211-240.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., et al. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, *19*(2), 313-330.

Marzal, A., & Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE trans. on Pattern Analysis and Machine Intelligence*, *15*(9), 926–932.

Pinker, S. (1994). *The language instinct: How the mind creates language*. New York: William Morro.

Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, *21*(1), 168-173.

Appendix: Python code that implements Span-based Normalized Edit Distance.

```
def SnedMatrix(s1, s2):
  slen = min(len(s1),len(s2))
  M = ones((len(s1)+1, len(s2)+1, slen+1),
           Float) * float(sys.maxint)
  M[0,0,0] = 0.0
  # do top left corner
  for k in xrange(1, len(s1)):
    for l in xrange(1, len(s2)):
      M[k,l,1] = d(s1[0:k], s2[0:l])
  M[len(s1), len(s2), 1] = d(s1, s2)

  # do rest
  for steps in xrange(1,slen):
    for fS1 in xrange(steps, len(s1)):
      for fS2 in xrange(steps,len(s2)):
        for tS1 in xrange(fS1+1, len(s1)):
          for tS2 in xrange(fS2+1, len(s2)):
            val = d(s1[fS1:tS1], s2[fS2:tS2]) + \
                  M[fS1, fS2, steps]
            if val < M[tS1, tS2, steps+1]:
              M[tS1,tS2,steps+1] = val
        val = d(s1[fS1:], s2[fS2:]) + \
          M[fS1, fS2,steps]
        if val < M[len(s1), len(s2), steps+1]:
          M[len(s1),len(s2), steps+1] = val
  return M
```

```
def Sned(s1,s2,M=None):
  if M == None:
    M = NedMatrix(s1,s2)
  minValue = 100
  for step in xrange(1,min(len(s1),len(s2))+1):
    value = M[len(s1),len(s2),step]/step
    if minValue > value:
      minValue = value
  return(minValue)
```