Automatic Construction of Weighted String Similarity Measures

Jörg Tiedemann
Department of Linguistics
Uppsala University
joerg@stp.ling.uu.se

Abstract

String similarity metrics are used for several purposes in text-processing. One task is the extraction of cognates from bilingual text. In this paper three approaches to the automatic generation of language dependent string matching functions are presented.

1 Introduction

String similarity metrics are extensively used in the processing of textual data for several purposes such as the detection and correction of spelling errors (Kukich, 1992), for sentence and word alignments (Church, 1993; Simard et al., 1992; Melamed, 1995), and the extraction of information from monolingual as well as multi-lingual text (Resnik and Melamed, 1997; Borin, 1998; Tiedemann, 1998a). One important task is the identification of so-called cognates, token pairs with a significant similarity between them, in bilingual text.

A commonly used technique for measuring string similarity is to look for the *longest common subsequence* (LCS) of characters in two strings; the characters in this sequence do not necessarily need to be contiguous in the original strings (Wagner and Fischer, 1974; Stephen, 1992). The length of the LCS is usually divided by the length of the longer string of the two original tokens in order to obtain a normalized value. This score is called the *longest common subsequence ratio* - LCSR (Melamed, 1995).

However, when it comes to different languages, a simple comparison of characters is usually not satisfactory to indicate the total correspondence between words. Different languages tend to modify loan words derived from the same origin in different ways. Swedish and English are an example for two languages with a close etymological relation but a different way of spelling for a large set of cognates. The spelling usually follows certain language specific rules, e.g. the letter 'c' in English words corresponds to the letter 'k' in Swedish in most cases of cognates. Rules like this can be used for the recognition of cognates from specific language pairs. In this paper three approaches to the automatic generation of language pair specific string matching functions are introduced. They include comparisons at the level of characters and n-grams with dynamic length.

All the three approaches presume linguistic similarities between two languages. In this study they were applied to word pairs from a Swedish/English text corpus and experimental results are presented for each of them.

2 Resources

Two types of textual resources were used in this study:

- reference lexicons for the automatic generation of string matching functions
- bilingual word pairs to be investigated with regard to string similarity

A collection of bilingual word pairs is easy to produce. Similarity metrics should be applicable to every possible word pair from this set. However, some restrictions can be imposed on the choice of appropriate pairs. In this study, all word pairs were derived from sentence aligned corpora of technical texts which were collected in the PLUG corpus (Tiedemann, 1998b) as part of the PLUG project[1] (Ahrenberg et al., 1998).

Technical texts are suitable for investigations on string similarity. The text collection which is examined comprises about 180,000 words per language and includes a large amount of technical expressions. Therefore, a comprehensive list of cognates can be expected from this corpus.

Some further constraints were set in order to restrict the set of bilingual word pairs to be investigated:

minimal token length: Each token should contain at least a certain amount of characters. Very short strings do not represent reliable sources for string comparison. The minimal length of tokens used in this study was set to four characters.

[1] The PLUG project is a cooperative project funded by "The Swedish Council for Research in the Humanities and Social Sciences" HSFR and the "Swedish National Board for Industrial and Technical Development" NUTEK.

maximal distance: Token pairs were taken from sentence aligned bi-text. The position of each token in its sentence can be used to reduce the number of potential candidate pairs. One possibility is to set a maximum for the difference in position for each token pair. In this study the position difference may not exceed 10 token.

minimal length difference ratio: Cognates should be of comparable length. Therefore, it is appropriate to restrict the set of candidates to strings whose length difference does not exceed a certain value. The quotient of the length of the shorter string and the length of the longer string can be used to calculate a ratio for measuring this difference. In this study the set of candidates were restricted to token pairs whose length difference ratio does not exceed a value of 0.7.

Using these three restrictions a set of 308,362 candidate pairs were obtained from parts of the PLUG corpus.

The selection of reference lexicons should be done with care. These lists of word pairs are decisive for the quality of the string matching function which will be produced. For availability reasons it was decided to use bilingual word lists which were produced in an automatic word alignment process. This is not the perfect solution because they contain quite a few errors and therefore they degrade the quality of the results to be produced.

The reference lexicons were generated by word alignment based on statistic measures and empirical investigations. The software which was used for the extraction is the Uppsala word alignment tool (Tiedemann, 1997; Tiedemann, 1999). The following two word lists were investigated:

GordSVEN: A list of 2,431 Swedish/English word alignments derived from the English/Swedish bi-text 'A Guest of Honour' by Nadine Gordimer with an estimated precision of about 95.8%.

ScaniaSVEN: A list of 2,223 Swedish/English word alignments derived from the Swedish/English bi-texts in the Scania95 corpus (Sca, 1998) by measuring LCSR scores[2] with an estimated precision of about 92.5%.

Both bi-texts are part of the PLUG corpus.

3 Basic Techniques

Dynamic Programming

A common technique for computing the length of the longest common subsequence for two given strings is to apply a dynamic programming algorithm (Stephen, 1992). If n is the length of string x and m is the length of string y an $(0..n, 0..m)$-matrix L describes the array of correspondences for these two strings. The initial column and the initial line of this matrix is set to 0. Now, a character matching function m has to be defined. The following definition for m is used to calculate the length of the LCS:

		b	a	l	a	n	c	e		a	r	m
	0	0	0	0	0	0	0	0	0	0	0	0
b	0	1	1	1	1	1	1	1	1	1	1	1
a	0	1	2	2	2	2	2	2	2	2	2	2
l	0	1	2	3	3	3	3	3	3	3	3	3
a	0	1	2	3	4	4	4	4	4	4	4	4
n	0	1	2	3	4	5	5	5	5	5	5	5
s	0	1	2	3	4	5	5	5	5	5	5	5
a	0	1	2	3	4	5	5	5	5	6	6	6
r	0	1	2	3	4	5	5	5	5	6	7	7
m	0	1	2	3	4	5	5	5	5	6	7	8
e	0	1	2	3	4	5	5	6	6	6	7	8
n	0	1	2	3	4	5	5	6	6	6	7	8
s	0	1	2	3	4	5	5	6	6	6	7	8

Figure 1: Calculating the length of the LCS of 'balance arm' and 'balansarmens' using Dynamic Programming.

$$\begin{array}{ll} m(x_i, y_j) = 1 & \forall x_i, y_j : x_i = y_j \\ m(x_i, y_j) = 0 & \forall x_i, y_j : x_i <> y_j \end{array}$$

Now, the matrix can be filled dynamically starting with the origin and using the following constraint:

$$\forall i \leq n \forall j \leq m : l_{i,j} = max(l_{i-1,j}, l_{i,j-1}, l_{i-1,j-1} + m(x_i, y_j))$$

Finally, the last field in this matrix contains the length of the LCS for the given strings. Note, that matching is defined for each element of the alphabet of characters including special symbols and white spaces. Consider the example in figure 1.

This algorithm can be modified by changing the character matching function. One possibility is to set priorities for specific matches by defining weights for the corresponding character. Now, the function m has to be modified to $m(x, y) = w(x)$ in all cases of $x = y$ where $w(x)$ is a weight for the character x[3]. Another possibility is to define a complete character matching function for all elements from the alphabet. That means, each $m(x, y)$ defines an independent matching value for the pair $[x, y]$[4].

[2] LCSR scores were calculated for tokens containing at least one alphabetic character and a threshold of 0.7 was used to filter the resulting list.

[3] A function that follows this definition will be referred to as *weighted matching function*.

[4] A function like this will be referred to as ***independent matching function*** in the further descriptions.

After this modifications the final value of the dynamic algorithm described above will be changed according to the new matching function. The result does not determine the length of the LCS anymore and therefore it will be considered as the *highest score of correspondence* (HSC).

Furthermore, the string segmentation can be modified. The algorithm above does not require a segmentation into characters. Dynamic programming can be applied to string pairs which were split into larger units than single characters. The only requirement for this is an adequate definition of the string matching function for all possible pairs of string units.

String Segmentation

There is a common segmentation problem with units larger than one element. The problem arises in case of overlapping units within the string. A simple approach is to parse the string from left to right and to find the longest possible segment starting at the current position. The segmentation process starts again at the position directly after the last position of the previous segment. This approach was used for string segmentation in this study.

Co-occurrence Statistics

Co-occurrence can be measured by different statistical metrics. They can be estimated by frequency counts for the elements to be considered. The value of $f(a)$ refers to the overall frequency counted for element a and the value of $f(x, y)$ refers to the cooccurrence frequency of the elements x and y in the collection of N aligned units.

The following formulas describe approximations of two commonly used metrics, Mutual Information (I) and the Dice coefficient ($Dice$) (Smadja et al., 1996; Church et al., 1991):

$$\begin{aligned} I(x,y) &\approx log_2 \frac{f(x,y) \cdot N}{f(x)f(y)} \\ Dice(x,y) &\approx \frac{2f_{x,y}}{f_x + f_y} \end{aligned}$$

Estimated Position

The proposed approaches to the generation of matching functions are based on the calculation of co-occurrence statistics. String units have to be matched at certain positions in order to measure co-occurrence frequencies. A so-called *estimated position* can be used to determine the position of the corresponding string unit. The following formula returns this value for the string pair $[x, y]$ and the ith element in y:

$$p(y_i) = round\left(i \cdot \frac{length(y)}{length(x)}\right)$$

Case Folding and Alphabet Restrictions

Case folding can be used to neutralize capitalization at the beginning of sentences. This can be useful for investigations of string similarity. However, valuable information can be lost especially when it comes to weighted matching functions. A higher priority for matching capitals would be desirable in cases of proper nouns. Furthermore, a reduced score might be useful when matching capitals with lower case characters. However, in this study case folding was applied.

Furthermore, the alphabet of the elements which shall be considered in the generation of the string matching function can be restricted. Results can be influenced strongly by wrong scores for special symbols and low frequent elements. This phenomenon appears especially in the case of independent matching functions, implying e.g. automatically generated m-functions may include matches for non-identical digits.

4 Generating the String Matching Function

4.1 Approach 1: Map Characters (VCchar)

The aim of this approach is to produce an independent matching function m based on a segmentation at the character level. The following heuristic is used:

> Pairs of vowels and consonants, respectively, which co-occur more frequently in the reference lexicon get a higher value in the m-function than lower frequency pairs. Pairs which do not co-occur at all get the function value 0.

In this approach the matching function is generated in three steps: First, all vowels at similar estimated positions in word pairs from the reference lexicon are mapped to each other. Consonants are processed in a similar manner. Second, the frequencies for all elements in the alphabet are counted on both sides and the frequency of each unique character mapping determines the co-occurrence frequency for each pair of characters. Finally, the Dice coefficient is used to calculate a value for each pair of characters in the list of character mappings. This value is used for the corresponding pair of characters in the final string matching function m. The Dice coefficient was chosen because it produces values between 0 and 1. In this way, the resulting similarity score remains a value in the range of 0 and 1 which is to prefer.

One problem arises with the definition of the set of vowels and consonants because the usage of letters can be context sensitive. For simplicity it was chosen to use a static disjunct definition of both sets (e.g. 'y' has been used as vowel only).

Dice score	freq	Swedish	English
0.6667	1	é	é
0.597	20	x	x
0.5189	261	m	m
0.5039	873	e	e
0.4925	736	a	a
0.4793	551	i	i
0.4702	402	o	o
0.2981	182	k	c
0.2292	413	a	e
0.2176	63	v	w
0.1812	273	a	i
0.1691	244	r	s
0.1656	238	e	i
0.1179	168	e	a
0.0681	40	å	o
0.0617	44	å	a
0.1019	111	ä	e
0.0914	34	ä	u
0.1032	112	ö	e
0.0982	64	ö	o

Figure 2: Approach 1: The top seven character mappings, the first seven non-identical character mappings, and the first two mappings for each Swedish diacritic in the Swedish/English VCchar matching function.

The resulting list (sorted after descending Dice scores) contains mainly pairs of identical letters on the top. Figure 2 shows, besides the seven highest rankings of pairs in the list, the first seven non-identical pairs, and mappings of Swedish diacritics which were obtained from the application to the Swedish/English reference lexicon *GordSVEN*.

There are mappings of non-identical characters which are hard to retrace, e.g. the relation between 'a' and 'i'. However, most of the highest rankings of non-identical pairs reflect interesting connections between different characters in Swedish/English word pairs. Relations between 'k' and 'c' ('korrekt' - 'correctly', 'kopia' - 'copy'), 'a' and 'e' ('beskriva' - 'describe', 'deformerad' - 'deformed'), 'v' and 'w' ('vatten' - 'water', 'två' - 'two') can be recognized easily. Furthermore, the algorithm provides interesting weights for pairs of identical characters. The function shows that infrequent letters like 'é' and 'x' can be matched with high confidence. In contrast with this, higher frequent characters with larger inconsistency like 'k', 'c', and 'w' obtain a lower value in this function, e.g. the match of the character 'c' in Swedish with the identical character 'c' in English will be scored with only 0.1123 points.

The automatically generated m-function for matching character pairs was applied for string similarity calculation to the list of Swedish/English candidates from parts of the PLUG corpus. The program returned 1,449 alignment candidates with an estimated precision of 96.8% when using a threshold of 0.3[5]

4.2 Approach 2: Map Vowel and Consonant Sequences (VCseq)

The goal in this approach is to generate a function for matching pairs of vowel sequences and pairs of consonant sequences. The motivation for this study is to extend the segmentation of strings from the character level to an n-gram model. Similarly to approach 1 a reference lexicon is used to calculate co-occurrence statistics for pairs of elements from the alphabet of string units. However, the segmentation of strings has been changed. Each string from the lexicon is split into vowel sequences followed by consonant sequences and the other way around. Furthermore, these sequences may be interrupted by character sequences from the set of remaining elements in the alphabet (characters which are neither in the set of vowels nor in the set of consonants). Now, all vowel sequences and consonant sequences, respectively, at identical estimated positions are mapped to each other and the frequency of each unique mapping is counted. Similarly to approach 1, Dice scores are estimated by using overall frequencies for each character sequence and the frequencies of each pair in the list of mappings. Figure 3 shows some mappings from the application of this algorithm to the Swedish/English word list GordSVEN.

Again, the pairs with the highest ranking are mainly identical strings. In contrast to the VCchar function there are already two non-identical pairs in the top-seven of the list. However, the co-occurrence frequency for them is very low (4 respective 2) and therefore the statistics are not very reliable. The value for the pair 'np' and 'dj' is due to four dictionary entries with morphological variants of ('anpassa','adjust') and ('anpassning','adjustment') and the low overall frequencies of 'np' and 'dj'. Similarly, the link between 'ktt' and 'bs' is due to three word pairs with variants of 'iakttagare' and 'observer' in the reference lexicon. A higher threshold for the co-occurrence frequency can be used to remove these pairs. However, a lot of interesting links would be lost in this way as well.

The mappings for Swedish diacritics are not very reliable as reflected in their scores. These values will not influence the similarity measurements a lot.

The program returned 651 candidates when applied to word pairs from the PLUG corpus with a

[5] The threshold value has to be much lower compared to other string similarity metrics like LCSR because token pairs obtain a much lower score in average.

Dice score	freq	Swedish	English
1	10	eo	eo
1	4	sr	sr
1	4	np	dj
1	2	dgs	dgs
1	2	lsj	lzh
1	2	rsp	rsp
1	2	schw	schw
1	4	np	dj
1	2	lsj	lzh
0.8	4	ktr	ctr
0.8	2	gsn	dg
0.7368	7	skr	scr
0.6667	3	ktt	bs
0.6316	6	tj	tw
0.0648	7	å	ou
0.064	27	å	o
0.1256	25	ä	ea
0.0899	25	ä	u
0.1183	23	ö	ea
0.1024	50	ö	o

Figure 3: Approach 2: The top seven vowel/consonant-sequence mappings, the first seven non-identical pairs, and the first two mappings for each Swedish diacritic in the Swedish/English VCseq matching function.

threshold of 0.15. The result yielded an estimated precision of 92.9%.

4.3 Approach 3: Map Non-Matching Parts (NMmap)

The last approach which will be discussed here differs from the other two by its general principle. In contrast to the other approaches the goal of the third approach is to extend a common matching function with some additional values for specific pairs. The basic matching function is represented by the m-function for LCS calculation (see section 3). Similarly to the other approaches a reference lexicon is taken to generate matching values for some specific pairs. Dynamic programming and a best trace computation can be used to identify non-matching parts of two strings. Now, these parts can be analyzed in order to find language pair specific correspondences. A simple idea is to match corresponding parts from the lists of non-matching strings to each other if they do not exceed a certain length. In this study a length of three character was chosen as a threshold. Consider figure 4 for an example of the mapping of non-matching parts for the Swedish/English word pair (kritiska,critical).

source string	k	r	i	t	i	sk	a	
target string	c	r	i	t	i	c	a	l

non-matching pairs: 'k' → 'c'
'sk' → 'c'
'' → 'l'

Figure 4: Approach 3: An example for mapping non-matching parts.

nm-weight	freq	Swedish	English
1	6	ska	c
0.942	162	k	c
0.875	21	sk	c
0.714	5	ras	d
0.545	6	v	w
0.532	25	e	a
0.5	9	ä	a

Figure 5: Approach 3: The top seven non-matching pair mappings in the Swedish/English matching function with a frequency higher than four.

Now, a weight for each pair of non-matching strings $[x, y]$ can be calculated by dividing its frequency by the total number of non-matching mappings for the source string x. Figure 5 shows the seven non-matching pairs with the highest ranking and a frequency of more than four which were computed from the Swedish/English list of cognates *ScaniaSVEN.*

The mappings reflect some typical differences in the writing of similar words in these two languages. The relation between 'ska' and 'c' can be seen in word pairs like (asymmetri**ska**,asymmetri**c**) and (automati**ska**,automati**c**). Correspondences between 'k' and 'c' are common in a lot of Swedish/English pairs, e.g. in (**k**orre**k**t,**c**orre**c**t) and (fun**k**tion,fun**c**tion). The mapping of 'sk' and 'c' appears similarly to the mapping of 'ska' to 'c' but for indefinite singular forms of Swedish adjectives . The connection between 'ras' and 'd' can be found in passive voice constructions, e.g. in (rekommende**ras**,are recommende**d**). The mapping of 'v' and 'w' is due to the fact that the letter 'w' does not exist in Swedish in practice. Finally, the change of vowels is quite common. The relation between 'e' and 'a' can be seen for example in pairs like (sid**a**,sid**e**) and (lin**a**,lin**e**). Furthermore, Swedish diacritics are represented by other characters in English. The mutation of 'ä' to 'a' can be seen for example in the pair (m**ä**rk,m**a**rk) but as reflected in the corresponding matching value this is not as reliable as the match of e.g. 'k' and 'c'.

The program returned 2,006 pairs with a score higher than 0.7 when applied to the Swedish/English word list which were obtained from parts of the PLUG corpus. This represents a gain of about 21%

additional links compared to the number of pairs which were obtained by calculating the basic LCSR scores and using the same threshold of 0.7. Even the estimation for the precision shows an improvement from 92.5% for LCSR extraction to 95.5% for the approach including non-matching scores.

5 Conclusion

In this paper three approaches were introduced with the common goal of generating language dependent string matching functions automatically in order to improve the recognition of string similarity. However, the first two approaches differ from the third one with regard to their general principle.

Both the first and the second approach produce an independent string matching function which does not rely on the comparison of characters itself. Therefore, these approaches are independent from the character sets which are used in each language. The difference between the first and the second approach concerns segmentation. While approach 1 uses a simple segmentation into sequences of characters the second approach groups vowels and consonants into n-grams. Because of the large variety of possible n-grams it is much less probable to get a hit when matching word pairs. Therefore, a much lower threshold has to be used in approach 2 in order to obtain cognate candidates. The problem with this is a much higher risk of finding wrong candidates especially for short strings. However, both approaches produce results with high precision between 92% and 97%. The recall is lower than the value which can be reached by means of LCSR scores. Compared at a similar level of precision the first approach returns roughly 87% and the second approach 39% as many candidates as LCSR extraction.

The third approach is based on LCS calculations. The goal is to add matching values for common non-identical characters and n-grams. It is not so flexible when applied to languages with different character sets, but it does produce the best result in the experiments that were carried out with Swedish/English word pairs. The basic set of cognates obtained by LCSR extraction was extended by about 21%. Even the precision for the resulting list could be estimated with a slight improvement from 92.5% for LCSR extraction to 95.5%[6]. Therefore, the third approach is by far the best of the three methods if languages with a fairly common character set are considered.

[6] The same threshold of 0.7 was used for both approaches.

References

Lars Ahrenberg, Magnus Merkel, Katarina Mühlenbock, Daniel Ridings, Anna Sågvall Hein, and Jörg Tiedemann. 1998. Parallel corpora in Linköping, Uppsala and Göteborg. Project application, available at http://stp.ling.uu.se/~corpora/plug/.

Lars Borin. 1998. Linguistics isn't always the answer: Word comparison in computational linguistics. In *Proceedings of the 11th Nordic Conference on Computational Linguistics NODALI98*, Center of Sprogteknologi and Department of General and Applied Linguistics, University of Copenhagen.

Kenneth W. Church, William Gale, Patrick Hanks, and Donald Hindle. 1991. Using Statistics in Lexical Analysis. In *Uri Žernik, editor, Lexical Acquistition: Using on-line resources to build a lexicon.* Lawrence Erlbaum.

Kenneth W. Church. 1993. Char_align: A Program for Aligning Parallel Texts at the Character Level. In *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives, ACL.* Association for Computational Linguistics.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computer Surveys.*

I. Dan Melamed. 1995. Automatic Evaluation and Uniform Filter Cascades for Inducing N-best Translation Lexicons. In *Proceedings of the 3rd Workshop on Very Large Corpora*, Boston/Massachusetts.

Philip Resnik and Dan I. Melamed. 1997. Semiautomatic aquisition of domain-specific translation lexicons. In *Proceedings of the Conference on Applied Natural Language Processing*, Washington, D.C.

1998. The Scania project. http://stp.ling.uu.se/~corpora/scania/.

Michael Simard, George F. Foster, and Pierre Isabelle. 1992. Using Cognates to Align Sentences in Bilingual Corpora. In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation*, Montreal/Canada.

Frank Smadja, Kathleen R. McKeown, and Vasileios Hatzivassiloglou. 1996. Translation Collocations for Bilingual Lexicons: A Statistical Approach. *Computational Linguistics, 22(1).*

Graham A. Stephen. 1992. String search. Technical report, School of Electronic Engineering Science, University College of North Wales, Gwynedd.

Jörg Tiedemann. 1997. *Automatical Lexicon Extraction from Aligned Bilingual Corpora.* Diploma thesis, Otto-von-Guericke-University, Magdeburg, Department of Computer Science.

Jörg Tiedemann. 1998a. Extraction of translation equivalents from parallel corpora. In *Proceedings of the 11th Nordic Conference on Computational Linguistics NODALI98*, Center for Sprogteknologi and Department of General and Applied Linguistics, University of Copenhagen.

Jörg Tiedemann. 1998b. Parallel corpora in Linköping, Uppsala and Göteborg (PLUG). work package 1. Technical report, Department of Linguistics, University of Uppsala.

Jörg Tiedemann. 1999. Uplug - a modular corpus tool for parallel corpora. In *Proceedings of the Symposium on Parallel Corpora*, Department of Linguistics, Uppsala University, Sweden.

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