Searching Similar (Sub)Sentences for Example-Based Machine Translation

Federica Mandreoli, Riccardo Martoglia, and Paolo Tiberio

Università di Modena e Reggio Emilia, Dip. di Ingegneria dell'Informazione, Modena, Italy
{tiberio.paolo, mandreoli.federica}@unimo.it; ricmart@sparc20.ing.unimo.it

Abstract. Translation is a repetitive activity. The attempt to automate such a difficult task has been a long-term scientific dream; in the past years research in this field has acquired a growing interest, making some forms of Machine Translation (MT) a reality. Among the several types of approaches in MT, one of the most promising paradigms is MAHT and, in particular, example-Based Machine Translation (EBMT). An EBMT system translates by analogy, using past translations to translate other, similar source-language sentences into the target language. The basic premise is that, if a previously translated sentence occurs again, the same translation is likely to be correct.
In this paper, we propose a solution based on a purely syntactic approach for searching similar sentences and parts of them in an EBMT system; the underlying similarity measure is based on the similarity between sequence of terms such that the sentences most close to a given one are those who maintain most of the original form and contents. The system efficiently retrieves and ranks the most similar sentences available and, when no useful suggestion exists, it proceeds with the retrieval of similar parts. We opted for a design that would require minimal changes to existing databases and whose similarity measure and search algorithms are completely independent from the involved languages. This work has been developed as a joint work with LOGOS S.p.A., a worldwide leader in multilingual document translation.

1 Introduction

Translation is a repetitive activity requiring a high degree of attention and the full range of human knowledge. The attempt to automate such a difficult task has been a long-term scientific dream of enormous social, political and scientific importance. Research in this field has acquired a growing interest in the past years, and, thanks also to recent technological advances, some degree of automatic translation (or Machine Translation - MT) is nowadays a reality. Among the above cited translation types, one of the most promising paradigms is MAHT and, in particular, example-Based Machine Translation (EBMT). Example-based translation is essentially translation by analogy. An EBMT system is given a set of sentences in the source language (from which one is translating) and their corresponding translations in the target language, and uses those examples to translate other, similar source-language sentences into the target language. The basic premise is that, if a previously translated sentence occurs again, the same translation is likely to be correct. Although in many cases EBMTs do not provide a translation by themselves, they suggest similar material thus helping to ensure the consistency of style and terminology. Such suggestions should be as close as possible to the source sentences so that editing the translation would take less time than generating a translation from scratch.

Usually, EBMT systems rely a DBMS. Corpora of bilingual text are maintained in a *translation memory*. More precisely, the translation memory is a database populated by previously translated documents together with the corresponding translation. The sentences of the source language document are first *aligned* with those of the target one, then the resulting pairs of sentences are stored in the translation memory together with some additional useful information. When the EBMT system is about to translate a document, it searches the translation

memory database for the document sentences, and presents the user with the translation(s) of the same sentences or similar ones. The effectiveness of translation memories increases with the amount of related translations it stores: As the user translates text, his/her translations become available in the database and they are particularly effective when translating a new document in the same field.

Due to the high complexity and extent of languages, in most cases it is rather difficult that a translation memory stores the exact translation of a given sentence. Thus, an EBMT system is proved to be a useful translator assistant only if most of the suggestions provided are based on similarity searching rather than exact matching. In this paper, we propose a solution based on a purely syntactic approach for searching similar sentences and parts of them in an EBMT system. The underlying similarity measure is based on the similarity between sequence of terms such that the sentences most close to a given one are those who maintain most of the original form and contents. The system efficiently retrieves and ranks the most similar sentences available in the translation memory and, when, no useful suggestion exists, it proceeds with the retrieval of similar parts. As far as the design of the system is concerned, we choose a solution that would require minimal changes to existing databases and whose similarity measure and search algorithms are completely independent from the involved languages. We show how sentence similarity search can be mapped into an SQL expression and optimized by conventional optimizer. The immediate practical benefit of our techniques is that approximate search in translation memory can be widely and efficiently deployed without changes to the underlying database.

The rest of the paper is organized as follows: in Section 2 we discuss related work. Section 3 presents the approach adopted for searching similar sentences and parts of them in the translation memory. In Section 4 we discuss the results of the conducted experiments. Finally, Section 5 concludes the paper.

2 Related work

The example(s) an EBMT system determines to be equivalent (or at least similar enough) to the text to be translated varies according to the approach taken by the system. The pioneer in EBMT system is Nagao [6] who suggested to emulate human translation practice in recognizing the similarity of a new source language sentence to a previously translated item by selecting identical phrases available in the translation memory except for a similar content word. This approach has been further developed by Brown [2] whose partial match is performed by allowing equivalence classes. Additional investigations of EBMT report interesting results restricted to subproblems such as function words [9], noun phrases [8], and propositional phrase attachment [9].

The main drawback of the above approaches is threefold. First, they presume the availability of a particular knowledge strictly related to the involved language, such as equivalence classes, thus requiring the intervention of the translator who has to annotate his/her translations before their insertion in the translation memory. Second, they never define a similarity measure thus forbidding the ranking of the results. Indeed, as far as we know, only few researchers tried to include the concept of approximate search in translation memories. Third, the entire sentence is always the smallest unit of search. EBMT systems do not consider the sentence contents for interesting parts.

Finally, commercial systems take an important role in the context of computer-aided translation. A popular set of commercial products includes Trados [11], Déjà Vu [3], and IBM's Translation Manager. They basically work in the same manner and show the drawbacks discussed above.

3 Searching Similar (Sub)Sentences in the Translation Memory

In this section, we propose a solution for searching similar sentences and parts of them in the translation memory. Given a sentence to be translated and a similarity threshold, the proposed techniques first search and rank the most similar sentences, that is those whose similarity with the query exceeds the threshold (*full matches*). If no sentence exists, the search is extended by comparing parts of the query with parts of the available sentences (*sub matches*). To efficiently identify candidate answers we use filters ad-hoc ensuring no false dismissals and few false positives. The proposed techniques are mapped into SQL expressions and can be widely and efficiently deployed without changes to the underlying database.

The underlying similarity measure is described below. As far as we know, this is the first attempt to formally introduce the concept of sentence similarity measure.

3.1 A (dis)similarity measure between sentences

The problem of defining a similarity measure between sentences is addressed by taking into consideration the two following requirements:

- the similarity measure should be as much independent from the translation context, such as involved languages and subjects, as possible. Naturally, the Translation Memory that the system creates and uses has to be bound to a specific couple of languages, but we can nonetheless be able to ensure the independence of the similarity search algorithms underneath. In this way, we ensure applicability without additional semantic knowledge. Indeed, either such additional knowledge is fully specified by translators, which would be almost impractical, or it should be a generic knowledge such as that given by thesaurus or ontology, which would often lead to "out of context" suggestions.
- the similarity measure should take into account the position of words in the sentence. Consequently, two sentences are considered to be different if they share the same set of words but in different positions. This requirement is based on the assumption that editing the suggestion should take less time than generating a translation from scratch.

We define a sentence as a sequence of terms and we consider two sentences to be as much similar as they maintain the same order of the same terms. For instance, the sentence "the dog eats the cat" is more similar to "the dog eats the mouse" than "the cat eats the dog".

The (dis)similarity between sentences has to be based on a distance function. Our notion of similarity measure recalls the edit distance usually adopted to capture the concept of approximate equality between strings [7]. In the following definition we adapt the classical definition of edit distance to the sentence context.

Definition 1 (Edit Distance between sequence of terms). *Let S_1 and S_2 be two sentences represented by the following sequence of terms: $\sigma(S_1) = t_1^1 \dots t_n^1$ and $\sigma(S_2) = t_1^2 \dots t_m^2$. The edit distance ed between $\sigma(S_1)$ and $\sigma(S_2)$ ($ed(\sigma(S_1), \sigma(S_2))$) is the minimum number of edit operations (i.e., insertions, deletions, and substitutions) of single terms needed to transform the first sequence into the second.*

Notice that if σ is just the identity function then two sentences which are equal except for word inflexions have an edit distance greater than 0. In information retrieval, stemming is a compressive technique widely used for removing grammatical suffixes and stopwords and converting the set of the remaining words to a common root form [1]. We therefore consider an alternative σ definition based on stemming operations. In the following $\mathcal{I}(S)$ will denote S itself ($\mathcal{I}$ is the identity function) whereas $\phi(S)$ will denote the sequence of terms as outcome of the application of stemming operations to S.

Example 1. Let us consider the following sentences: S_1 = "the dog eats the cat" and S_2 ="the dogs eat the cats" where $\phi(S_1) = \phi(S_2) = dog\ eat\ cat$. Then $ed(S_1, S_2) = 3$ whereas a more significative outcome is obtained by applying stemming: $ed(\phi(S_1), \phi(S_2)) = 0$.

In the following we introduce the definition of similarity match based on the distance function of Def. 1. Similarity matches will be exploited for full matches and partial matches.

Definition 2 (Similarity Match). *Given two sentences S_1 and S_2 and a distance threshold d, the similarity match $editDistance(\sigma(S_1), \sigma(S_2), d)$ returns the edit distance value if its two sequence arguments are within edit distance of the integer argument d, a negative value otherwise.*

3.2 Searching for full matches

Translators usually submit one or more documents to the EBMT system. Such documents contain a considerable number of sentences. Given a set of sentences to be translated and a distance threshold d, we discuss an efficient technique for searching similar sentences in the translation memory based on the distance function of Def. 1. As far as the distance threshold d is implied notice that it should be specified as a natural number. In the context of EBMT systems, searching for sentences within d errors, where d is independent of the length of the query could be little meaningful. For example, searching for sentences within 3 errors given a query of length 6 is deeply different from searching sentences within the same number of errors given a query of length 20. For this reason we consider d as the percentage of admitted errors with respect to the sentences to be translated.

Definition 3 (Full matches). *Given the set of sentences Q to be translated, the set of sentences TM in the translation memory, and a distance error d, for each $S_q \in Q$ of length $|S_q|$ retrieve all sentences $S \in TM$ (denoted as suggestions) such that*

$$editDistance(\sigma(S_q), \sigma(S), round(d * |\sigma(S_q)|)) \geq 0$$

and rank them on the basis of the editDistance outcome.

In our mapping σ is the stemming function ϕ. Therefore, the SQL expression exploiting filtering techniques for full matches, similar to the one proposed in [4], has the following form:

```
INSERT INTO FULLMATCH
SELECT      R2.COD AS COD2, R1.COD AS COD1, R1.TARG_SENT AS SUGGESTION,
            edit_distance(R1.STEM_SENT,R2.STEM_SENT,ROUND(d*R2.LEN)) AS DIST
FROM        TM R1, Q R2
WHERE       <filters>
AND         edit_distance(R1.STEM_SENT,R2.STEM_SENT,ROUND(d*R2.LEN))>=0
ORDER BY    COD2, DIST, COD1
```

where `TM` is the translation memory and `Q` is an auxiliary table storing the sentences to be translated which is created on-the-fly when we perform full/partial matches and deleted upon completion; `edit_distance(,,)` is a user-defined function (UDF) implementing the similarity match $editDistance(,,)$. Schemata of `TM` and `Q` and the meaning of their attributes are shown in the following table.

TM	Q	Attribute	Meaning
✓	✓	COD	key attribute
✓	✓	SOURCE_SENT	sentence in the source language
✓	✓	STEM_SENT	corresponding stemmed version
✓	✓	LEN	length of the stemmed sentence
✓		TARG_SENT	translation in the target language
✓		APOS	alignment information, used for partial matches (see Sub. 4.3)
✓		SPOS	alignment information, used for partial matches (see Sub. 4.3)

```
INSERT INTO FULLMATCH
SELECT      R2.COD AS COD2, R1.COD AS COD1, R1.TARG_SENT AS SUGGESTION
            edit_distance(R1.STEM_SENT,R2.STEM_SENT,ROUND(d*R2.LEN)) AS DIST
FROM        TM R1, TMq R1q, Q R2, Qq R2q
WHERE       R1.COD = R1q.COD
AND         R2.COD = R2q.COD
AND         R1q.Qgram = R2q.Qgram
            -- position filtering
AND         ABS (R1q.POS - R2q.POS) <= ROUND(d * R2.LEN)
            -- length filtering
AND         ABS (R1.LEN - R2.LEN) <= ROUND(d * R2.LEN)
GROUP BY    R2.COD, R1.COD, R1.STEM_SENT, R2.STEM_SENT, R1.LEN, R2.LEN
            -- count filtering
HAVING      COUNT(*)>= (R1.LEN - 1 - (ROUND(d * R2.LEN) - 1) * q)
AND         COUNT(*) >= (R2.LEN - 1 - (ROUND(d * R2.LEN) - 1) * q)
AND         edit_distance(R1.STEM_SENT,R2.STEM_SENT,ROUND(d*R2.LEN))>=0
ORDER BY    COD2, DIST, COD1
```

Fig. 1. `Q1`: Query for full matches with filters

Filters Filters are exploited for computing both full and partial matches. First, filters efficiently select a set of candidate answers ensuring no false dismissals and containing few false positives. Then, as a second step, we perform $editDistance(,,)$, an expensive and in-memory algorithm which eliminates all false positives. Filters rely on matching short parts of length q, denoted as *q-grams*, of the query sentences with those of the translation memory sentences. Below, we briefly introduce the notion of positional q-gram for sentences as a variant of that given in the literature for words [10]. Given a sentence S and its representation as sequence of terms $\sigma(S)$, its *positional q-grams* are obtained by "sliding" a window of length q over the terms of $\sigma(S)$. Since q-grams at the beginning and the end of the sentence can have fewer than q terms from $\sigma(S)$, we introduce new terms "#" and "$" not in the term grammar, and conceptually extend the sequence $\sigma(S)$ by prefixing it with $q-1$ occurrences of "#" and suffixing it with $q-1$ occurrences of "$".

Definition 4 (Positional q-gram). *A positional q-gram of a sentence S is a pair $(i; [i \dots i+q-1])$, where $[i \dots i+q-1]$ is the q-gram of $\sigma(S)$ that starts at position i, counting on the extended sentence. The set $G_{\sigma(S)}$ of all positional q-grams of a sequence $\sigma(S)$ is the set of all the $|\sigma(S)|+q-1$ pairs constructed from all q-grams of $\sigma(S)$.*

In our approach, q-grams are computed for the stemmed sentences of the query and of the translation memory and stored in two auxiliary tables `Qq` and `TMq` with the same schema `(COD,POS,Qgram)`. For each sentence S, its positional q-grams are represented as separate tuples in the above tables, where `POS` identifies the position of the q-gram `Qgram`. The positional q-grams of S share the same value for the attribute `COD`, which serves as the foreign key attribute to the table storing S.

Taking advantage of some key properties of q-grams, we exploit three filtering techniques which basically take into account the total number of q-gram matches and the position of individual q-gram match.

Proposition 1 (Count Filtering). *Consider sequences $\sigma(S_1)$ and $\sigma(S_2)$, of lengths $|\sigma(S_1)|$ and $|\sigma(S_2)|$, respectively. If $\sigma(S_1)$ and $\sigma(S_2)$ are within an edit distance of d, then the cardinality of $G_{\sigma(S_1)} \cap G_{\sigma(S_2)}$, ignoring positional information, must be at least $max(|\sigma(S_1)|, |\sigma(S_2)|) - 1 - (d-1) * q$.*

Proposition 2 (Position Filtering). *If sequences $\sigma(S_1)$ and $\sigma(S_2)$ are within an edit distance of d, then a positional q-gram in one cannot correspond to a positional q-gram in the other that differs from it by more than d positions.*

Proposition 3 (Length Filtering). *If sequences $\sigma(S_1)$ and $\sigma(S_2)$ are within an edit distance of d, their lengths cannot differ by more than d.*

Proves and explanations of the above filters can be found in [10].

The complete query `Q1` for full matches presented in Figure 1 shows that filters can be expressed as an SQL expression and efficiently implemented by a commercial relational query engine. Consequently, when documents to be translated are submitted to our EBMT system, it first searches full matches by mapping the problem to a conventional SQL expression. The involved SQL expression joins the auxiliary tables for q-gram sentences, `TMq` and `Qq`, with the query table `Q` and the translation memory `TM` to retrieve the sentence pairs with the corresponding suggestions in the target language to be inserted in the `FULLMATCH` table. Matches are first ordered on the basis of the query sentences then, for each query sentence, we rank similar sentences on the basis of the value returned by the distance function. The position filtering is implemented by pruning out all pair of sentences which share one q-gram, but the relative positions differ substantially. The length filtering is mapped by comparing the length of the involved sentences. Finally, the count filtering is implemented by comparing the number of q-gram matches with the length of the involved sentences. Some examples showing the effectiveness of full matches can be found in Subsec. 4.3.

3.3 Beyond full matches: searching for partial matches

Experience with several language pairs has shown that producing an EBMT system which provides reasonable translation coverage of unrestricted texts requires a great number of pre-translated text [2]. Consequently, translators may submit sentences for which no full match exists. Anyway, the sentences stored in the translation memory could be partially useful. We thus propose to search for matches between parts of query sentences and parts of the translation memory sentences. Given a distance error and a minimum suggestion length, for the query sentences for which no full match exists, we suggest those parts of the translation memory sentences exceeding the minimum length that we consider to be useful for the translation. In doing this, we avoid to present suggestions which are contained in other ones.

Definition 5 (Part suggestion). *Let Q be the set of sentences to be translated, $S_q \in Q$, TM be the set of sentences in the translation memory, $S \in TM$, $subD$ be a distance threshold, and $minL$ the minimum length. Then, $(\sigma(S)[t_i, \ldots, t_j], k, h)$ is an S_q part suggestion iff $(j-i+1) \geq minL$ and exists $\sigma(S_q)[s_k, \ldots, s_h]$ such that $s_k = t_i$ and $s_h = t_j$ and:*

$$editDistance(\sigma(S)[t_i, \ldots, t_j], \sigma(S_q)[s_k, \ldots, s_h], round(subD * |\sigma(S_q)[s_k, \ldots, s_h]|)) \geq 0$$

Notice that $\sigma(S)[t_i, \ldots, t_j]$ denotes the $\sigma(S)$ subsequence of length $j-i+1$ starting at position i. We denote with $PSug_{S_q}^{TM}$ the set of all S_q part suggestions available in the translation memory TM. Notice that $PSug_{S_q}^{TM}$ can be partitioned on the basis of the starting point: $PSug_{S_q}^{TM}[K] = \{(S[t_i, \ldots, t_j], k, h) \in PSug_{S_q}^{TM} \mid k = K\}$.

Definition 6 (Partial matches). *Given the set of sentences Q to be translated, the set of sentences TM in the translation memory, a distance threshold $subD$, and minimum length $minL$, for each $S_q \in Q$*

- *compute $PSug_{S_q}^{TM}$;*
- *delete from $PSug_{S_q}^{TM}$ all suggestions $(\sigma(S)[t_i, \ldots, t_j], k, h)$ such that $(\sigma(S)[t_n, \ldots, t_m], k', h')$ exists for which $[k, h] \subset [k', h']$;*
- *partition $PSug_{S_q}^{TM}$ into $\{PSug_{S_q}^{TM}[K] \mid K \in [1, |\sigma(S_q)| - minL]\}$;*
- *rank $PSug_{S_q}^{TM}[K]$ for $K \in [1, |\sigma(S_q)| - minL]\}$ on the basis of the distance outcome.*

Example 2. Suppose that $minL = 3$ and $subD = 0.3$ and that we search for the partial matches for the query sentence "So, welcome to the world of computer generated art" where "welcome world compute generate art" is the outcome of the stemming function. The sentences stored in the translation memory are shown in the following table:

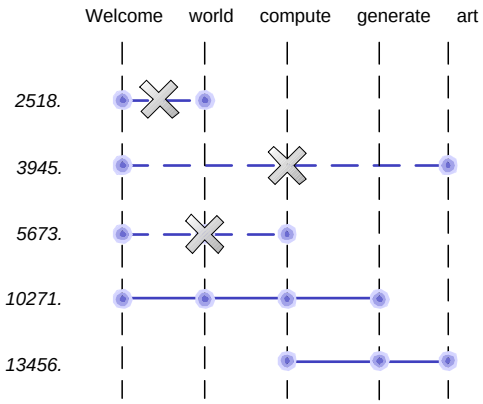


Fig. 2. An example of partial matches

COD	SOURCE_SENT	STEM_SENT
2518	So, welcome to the world of music.	welcome world music
3945	We welcome our guests to the Madrid Art Expo!	welcome guest Madrid art Expo
5673	Welcome to the world of computer aided translation!	welcome world compute aid translation
10271	We welcome you to the world of computer generated fractals.	welcome world compute generate fractal
13456	This is a computer generated art work.	be compute generate art work

All the above sentences present some common words with the query sentence. Figure 2 shows all the *candidate part suggestions* identified as those parts of the stemmed sentences starting and ending with the same term of some parts of the stemmed query shown on the top of the figure. For instance, 3945[*welcome, ... ,art*] is a candidate part suggestion since the following subsequence of the stemmed version of the sentence 3945 *welcome guest Madrid art* shares the first and last words with *welcome world compute generate art*. The part of 2518 is not a suggestion since it is too short, the part of 3945 is not a suggestion since

$$editDistance(3945[welcome, \dots, art], Q[welcome, \dots, art], round(0.3 * |5|)) < 0$$

The remaining candidate suggestions are all "true" suggestions. From the computation of the partial matches 5673[*welcome, ... ,compute*] is deleted since it is contained in 10271[*welcome, ... ,generate*].

The above example shows that the computation of partial matches requires two steps: the identification of all the candidate part suggestions followed by the proper computation of partial matches. More formally, a candidate suggestion is a part $[t_i, \dots, t_j]$ of a stemmed sentence $\phi(S)$ such that exists $[s_k, \dots, s_h]$ in the stemmed query sentence $\phi(S_q)$ where $t_i = s_k$ and $t_j = s_h$. As for full matches, also for partial matches we provide a mapping into SQL expressions.

Identification of candidate part suggestions The computation of candidate part suggestions is entrusted to the UDF `FindMatchPos (query_code,query,sent_code,sent)` which given a stemmed query sentence, and a stemmed sentence in the translation memory, it first lexicographically sorts the two sentences also maintaining the word original position information; then it searches for equal words and stores their positions in a table, named `MATCHPOS`. The schema of `MATCHPOS` is `(QCOD,TMCOD,QPOS,TMPOS)` where `QCOD` and `TMCOD` are the code of the query sentence and translation memory sentence respectively, `QPOS` and `TMPOS` are the positions of equal words in the query and in the sentence respectively. Notice that the computational cost of `FindMatchPos` is $O(n \log n)$ corresponding to the sort computational cost.

```
SELECT   R2.COD AS COD2, R1.COD AS COD1
FROM     TM R1, Q R2, TMq1 R1SUBq, Qq1 R2SUBq
WHERE    R1.COD = R1SUBq.COD
AND      R2.COD = R2SUBq.COD
AND      R1SUBq.Qgram = R2SUBq.Qgram
AND      R2.COD NOT IN (SELECT COD2 FROM FULLMATCH)
         -- count filtering
GROUP BY R2.COD, R1.COD, R1.STEM_SENT, R2.STEM_SENT
HAVING   COUNT(*) >= minL - 1 - (ROUND(subD*minL)-1)*q_sub - (q_sub-1)*2
AND      findMatchPos(R1.COD, R1.STEM_SENT, R2.COD, R2.STEM_SENT) >= 0
```

Fig. 3. `Q2`: identification of candidate part suggestions with filters

The set of candidate sentence pairs to which we apply the `FindMatchPos` function is obtained by applying a cheap, approximate algorithm that prunes out all pair of sentences (S_q, S) such that the sentence S surely does not contain a part suggestion of the query S_q. We achieve this by performing the `FindMatchPos` function along with some additional filters (query `Q2` in Fig. 3). From the three filtering techniques defined for full matches, length filtering and position filtering are not applicable whereas count filtering is still applicable since the two sentences must obviously have a certain minimum number of q-grams, in order to contain the one a part suggestion for the other. Anyway, notice that the names of the auxiliary tables employed for storing positional q-grams, i.e. `R1SUBq` and `R2SUBq`, are distinct from those employed for full matches, i.e. `R1q` and `R2q`. Indeed, for partial matches q-grams are computed from stemmed sentences not extended with "#" and "$", since it is almost impractical to extend all the possible parts of all the sentences contained in the translation memory. Such a difference influences both the the size q_{sub} of q_{sub}-grams and count filtering. As to the former, the correctness of the filters based on q_{sub}-grams is ensured if q_{sub} is less or equal to the value specified in the following Proposition.

Proposition 4. *The propositional formula "If S contains an S_q part suggestion then exists at least one q_{sub}-gram shared by S and S_q" is true if and only if $q\_sub \in [1, \lfloor \frac{minL}{round(subD*minL)+1} \rfloor]$*

Intuitively, in order to establish the q_{sub} limits we consider the minimum length of the part suggestion, i.e. $minL$, and we distribute the $round(subD*minL)$ allowed errors. The worst case occurs when we have $round(subD*minL)+1$ "safe" parts with the same size: $\lfloor \frac{minL}{round(subD*minL)+1} \rfloor$.

As to count filtering, the formula $minL-1-(round(subD*minL)-1)*q_{sub}-(q_{sub}-1)*2$ is obtained from the condition shown in Proposition 1 by substituting $max(|\sigma(S_1)|, |\sigma(S_2)|)$ with the minimum length $minL$ and by subtracting to the total count the number of q_{sub}-grams with wild cards, i.e. $(q_{sub}-1)*2$. It corresponds to the minimum number of q_{sub}-grams shared by two sentences with a similar part whose length is at least $minL$.

Computation of partial matches Once the positions of equal words have been stored in `MATCHPOS`, we proceed with the proper computation of partial matches. Such partial matches should satisfy the following requirements: they should be at least $minL$ long, they should be similar to the query sentences, and, finally, they should not be contained in other suggestions. To this end, we present a technique for computing partial matches efficiently. Efficiency is again ensured by means of filters allowing the checking of the edit distance in a limited set of candidate part suggestions. In this case, filters are based on the same properties stated for full matches but take into account the relative positions of equal words. Let $E[\sigma(S_q), \sigma(S)] = \{(p^{S_q}, p^S)\}$ be the set of positions of equal words stored in table `MATCHPOS` for the sentence pair (S_q, S), then the exploited filtering techniques are described below.

Proposition 5 (Count Filtering for Sub Matching). *Consider sequences $\sigma(S_1)$ and $\sigma(S_2)$ and $(p^{S_1}, p^{S_2}) \in E[\sigma(S_1), \sigma(S_2)]$ and $(q^{S_1}, q^{S_2}) \in E[\sigma(S_1), \sigma(S_2)]$ such that $p^{S_1} \leq q^{S_1}$ and*

```
INSERT INTO SUBMATCH
SELECT      M1.QCOD AS QCOD, M1.QPOS AS QPOS1, M2.QPOS AS QMPOS2,
            M1.TMCOD AS TMCOD, M1.TMPOS AS TMPOS1, M2.TMPOS AS TMPOS2,
            edit_distance(Pos2Seq(R1.STEM_SENT, M1.TMPOS, M2.TMPOS),
            Pos2Seq(R2.STEM_SENT, M1.QPOS, M2.QPOS), ROUND(subD*R2.LEN)) AS DIST
FROM        MATCHPOS M1, MATCHPOS M2, TM R1, Q R2
WHERE       M1.TMCOD = M2.TMCOD AND M1.QCOD = M2.QCOD
AND         M1.TMCOD = R1.COD AND M1.QCOD = R2.COD
            -- extreme checking
AND         M1.TMPOS < M2.TMPOS AND M1.QPOS < M2.QPOS
            -- minimum length checking
AND         (M2.QPOS - M1.QPOS + 1) >= minL AND (M2.TMPOS - M1.TMPOS + 1) >= minL
            -- length filtering
AND          ...
            -- position filtering
AND          ...
            -- inclusion checking
AND         NOT EXISTS
            (SELECT M3.QPOS, M3.TMPOS, M4.QPOS, M4.TMPOS
            FROM MATCHPOS M3, MATCHPOS M4, TM R3, Q R4
            WHERE M3.TMCOD = M4.TMCOD AND M3.QCOD = M4.QCOD
            AND M3.TMCOD = R3.COD AND M3.QCOD = R4.COD
            AND M3.TMPOS < M4.TMPOS AND M3.QPOS < M4.QPOS
            AND (M4.TMPOS - M3.TMPOS + 1) >= minL
            -- length, count and position filtering
            AND ...
            AND edit_distance (Pos2Seq(R3.STEM_SENT, M3.TMPOS, M4.TMPOS),
            Pos2Seq(R4.STEM_SENT, M3.QPOS, M4.QPOS), <subD>) >= 0
            AND M3.QCOD = M1.QCOD
            AND ((M1.QPOS = M3.QPOS AND M4.QPOS > M2.QPOS)
            OR (M3.QPOS < M1.QPOS AND M2.QPOS = M4.QPOS)
            OR (M3.QPOS < M1.QPOS AND M4.QPOS > M2.QPOS)
            OR (M1.QPOS = M3.QPOS AND M2.QPOS = M4.QPOS AND M1.TMCOD = M3.TMCOD
            AND ((M1.TMPOS = M3.TMPOS AND M4.TMPOS > M2.TMPOS)
            OR (M3.TMPOS < M1.TMPOS AND M2.TMPOS = M4.TMPOS)
            OR (M3.TMPOS < M1.TMPOS AND M4.TMPOS > M2.TMPOS) ))))
ORDER BY    QCOD, QPOS1, DIST
```

Fig. 4. Q3: Query for partial matches with filters

$p^{S_2} \leq q^{S_2}$. *If* $\sigma(S_1)[p^{S_1}, \ldots, q^{S_1}]$ *and* $\sigma(S_2)[p^{S_2}, \ldots, q^{S_2}]$ *are within an edit distance of* d*, then the number of common words, ignoring positional information, must be at least*

$$max(|\sigma(S_1)[p^{S_1}, \ldots, q^{S_1}]|, |\sigma(S_2)[p^{S_2}, \ldots, q^{S_2}]|) - d$$

where d *is the number of allowed errors.*

Proposition 6 (Position Filtering for Sub Matching). *If the sequences* $\sigma(S_1)[p^{S_1}, \ldots, q^{S_1}]$ *and* $\sigma(S_2)[p^{S_2}, \ldots, q^{S_2}]$ *are within an edit distance of* d*, then a word in one cannot correspond to a word in the other that differs from it by more than* d *positions.*

Proposition 7 (Length Filtering for Sub Matching). *If the sequences* $\sigma(S_1)[p^{S_1}, \ldots, q^{S_1}]$ *and* $\sigma(S_2)[p^{S_2}, \ldots, q^{S_2}]$ *are within an edit distance of* d*, their lengths cannot differ by more than* d*.*

The mapping of the filtering techniques into SQL is shown in Figure 4 where UDF `Pos2Seq(,,)` returns the part of the given sentence starting and ending at the given extremes. Due to the lack of space, we do not show the complete mapping of the supported filters which is similar to the one proposed for full matches (see Fig. 1). Notice that the inclusion checking verifies the non-existence of a subsequence which passes the filters and satisfies the condition on the distance function and which contains the subsequence considered.

4 Experimental Evaluation

In this section we present the results of an experimental evaluation of the techniques described in the previous sections.

4.1 Data Sets

To effectively test the system, we used two real data sets:

- *Collection1*, taken from two versions of a software technical manual. It consists of 1497 reference sentences and 419 query sentences.
- *Collection2*, a complete Translation Memory provided by LOGOS, containing translations from one of their customers' technical manuals. There are 34550 reference sentences, complete with translation, and 421 query sentences.

Collection1 only contains English sentences and it was mainly used to test the system effectiveness in finding useful suggestions in a relatively small translation memory. Collection2, on the other hand, contains both English sentences and their Italian translations; because of its greater size, we were able to use it not only to test the system behaviour with well established and more repetitive data, but also to test the system efficiency. Further, being the sentences available in two languages, the system has also been tested in aligning them and giving suggestions in the target language.

4.2 Implementation

The similarity search techniques described in previous sections have been implemented using Java2 and JDBC code; the underlying DBMS is Oracle 8i Enterprise Edition running on a Dell OptiPlex GX1 Microsoft NT workstation. We developed three options for the similarity search process:

1. *Search for full matches*: We issued query `Q1` to the DBMS.
2. *Search for partial matches - accurate*: We issued queries `Q2` and `Q3`. The size of q_{sub}-grams used in query `Q2` is 1 and the minimum length $minL$ is 3.
3. *Search for partial matches - fast*: We consider a variant of the above search option by using the same q-gram size for full and partial matches. The adopted value corresponds to the size q used for full matches and exceeds the limit stated in Proposition 4. In this way, the same auxiliary tables for q-grams, `TMq` and `Qq`, are exploited both for full and partial matches. Since filtering technique in query `Q2` is no longer correct, we expected the exclusion of useful suggestions for partial matches. In this case, we were interested in quantifying the tradeoff between the number of excluded suggestions and the time gained in the execution.

Queries have been implemented using standard SQL code and Java Stored Procedures. In particular, the edit distance function has been implemented as the optimized "diagonal transition algorithms" described by Ukkonen [12]: the computation of only certain elements of the dynamic programming matrix allows the improvement of the worst case computational complexity which is $O(d^2)$ for checking whether the distance is $\leq d$ or not. The computation and storage of q-grams of the translation memory sentences in the corresponding auxiliary tables were performed after their insertion in the translation memory. We therefore preferred the above solution to the option of creating auxiliary tables on-the-fly, as suggested in [4]. Indeed, in our case, the creation on the fly would have required four/five minutes, which could not be considered a negligible time during query processing.

As for query efficiency, by issuing conventional SQL expressions we have been able to rely on the DBMS standard optimization algorithms; we just added some appropriate indexes on the q-gram tables to further speed up the filtering and search processes.

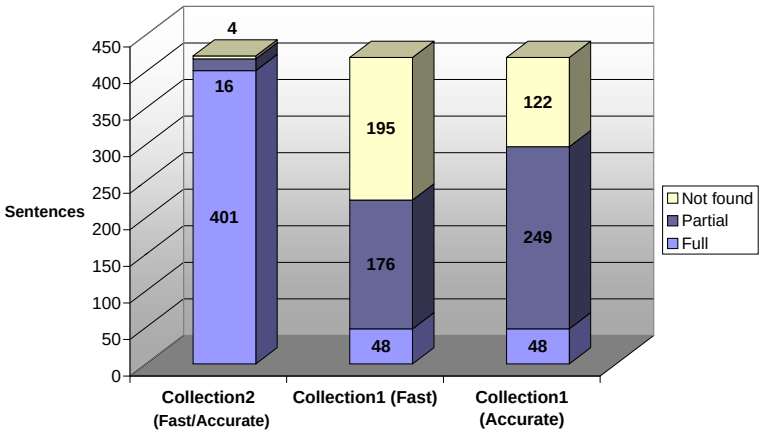


Fig. 5. Coverage of the collections as outcome of the similarity search process

4.3 Performance

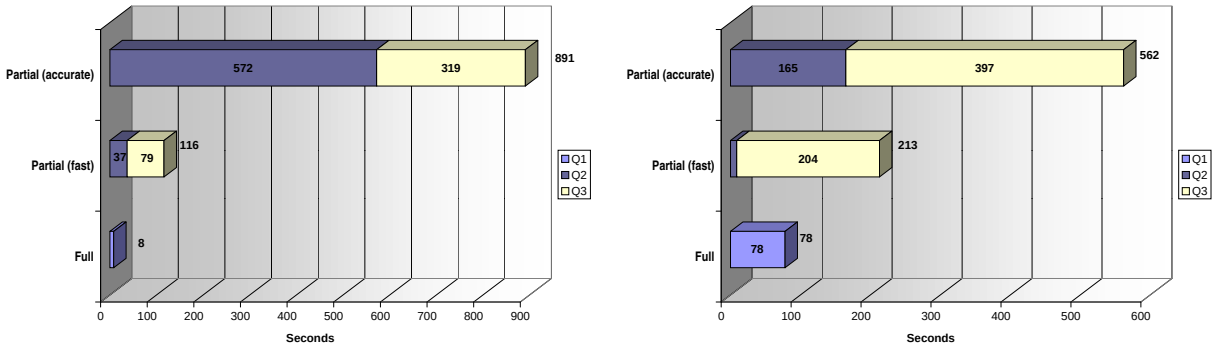
The system performance was tested both with respect to the effectiveness and the efficiency of the proposed search techniques. As far as efficiency is concerned, we observed performance trends under the parameters that are associated with our problem (i.e. q-gram size, number of allowed errors expressed in term of the percentage of admitted errors with respect to the query sentence). Finally, we used two performance metrics corresponding to the running time of the algorithms and the size of the candidate set. Unless explicitly specified, all experiments have been conducted by setting the value of q to 3 and d to 0.2.

Effectiveness To test standard Information Retrieval systems effectiveness it is necessary to evaluate how precise is the answer set to a user query: this is often achieved by calculating recall and precision figures over a well known test collection. Since there are no such collections to test a machine translation and sentence similarity system, we were able to evaluate more practically other test figures, focused on our particular problem. One of the fundamental aspects related to the effectiveness of the proposed techniques is the *coverage* of the query sentences, that is the percentage of query sentences for which at least one suggestion, obtained both from a full or a partial match, has been found in the translation memory. Figure 5 shows that our search techniques ensure a good coverage for the considered collections. In particular, the good consolidation of Collection 2 implies a very high level of coverage (over 99%), where most of the suggestions concerns full matches, whereas for the remaining query sentences the coverage is independent from the implementation of fast or accurate search. As to Collection 1 which is relatively small, notice that partial matches cover a percentage of query sentences (about 60%) which is five times that covered by full matches (about 12%). Moreover, from the comparison between accurate and fast search it follows that fast search prunes out about 18% more sentences than accurate search.

Figure 6 shows some examples of suggestions retrieved in Collection 2 by applying full and partial matches. The emphasized parts denote interesting parts for suggestions. Notice that, while full sentence suggestions in the target language can be directly selected from the translation memory, part suggestions must be extracted from the sentence in the target language containing some similar parts. To this end, once the submatches in the source language have been found out, the corresponding parts in the target language, which are the proper suggestions, are determined by means of some word alignment techniques. Due to the lack of space, we do not discuss those techniques. More details can be found in [5].

Query sentence: Position the 4 clips (D) as shown and at the specified dimensions.
Similar sentence in the source language: Position the 4 clips (A) as shown and at the specified distance.
Corresponding sentence in the target language: Posizionare le 4 mollette (A) come indicato e alla distanza prevista.
Query sentence: On completion of *electrical connections, fit the cooktop in place from the top* and *secure it by means of the clips* as shown.
Sentence containing a similar part: After the *electrical connection, fit the hob from the top* and hook it to the support springs, according to the illustration.
Corresponding sentence in the target language: Dopo aver eseguito il *collegamento elettrico, montare il piano cottura dall' alto* e agganciarlo alle molle di supporto come da figura.
Suggestion in the target language: collegamento elettrico, montare il piano cottura dall'alto
Sentence containing a similar part: *Secure it by means of the clips.*
Suggestion in the target language: Fissare definitivamente per mezzo dei ganci.

Fig. 6. Examples of full and partial matches



(a) Collection1 (b) Collection2

Fig. 7. Running time tests

Efficiency We examined first the running time of the three similarity search processes. As to Collection2, figure 7 shows that the running time required to cover 401 query sentences by means of full matches over 34550 reference sentences is 78 seconds. The execution time difference between the accurate and the fast version of partial search, i.e. 349 over 562 seconds, makes the fast solution much more feasible also considering the equally high coverage (see Fig. 5). Our experiments on Collection 1 show that a relatively small and little consolidated translation memory hits the performance of the system. Anyway, notice that, thanks to filtering techniques, the system takes 8 seconds to compare the 419 query sentences against the 1497 reference sentences for full matches. As to partial matches, also in this case the fast solution is 7.5 times faster than the accurate solution but it shows a smaller coverage.

The scalability test performed on Collection2 are synthetized in Fig. 8. Query `Q1` shows a sub-linear behaviour, queries `Q2` and `Q3` have a linear behaviour. The average time for sentence processing is from 2 sec. over 25% of the total amount of query sentences up to 1.5 sec. over the total amount.

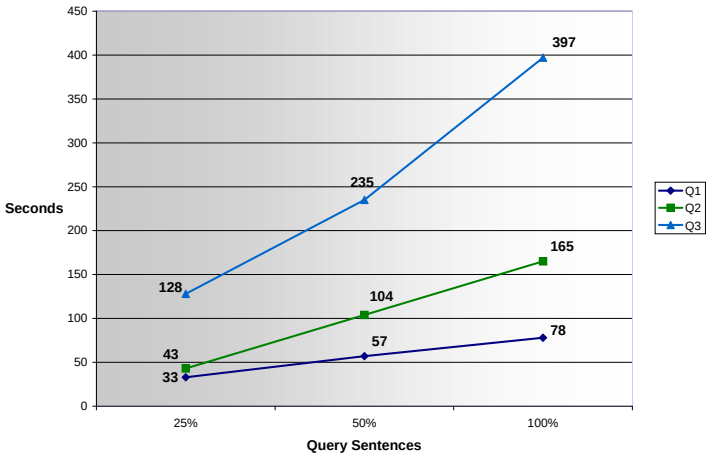


Fig. 8. Scalability tests on Collection2

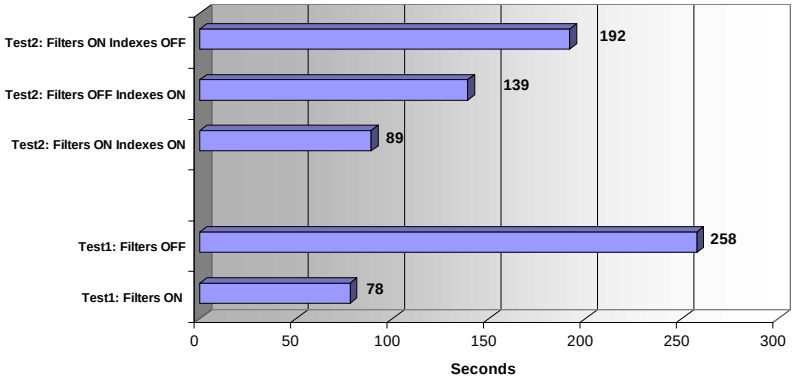
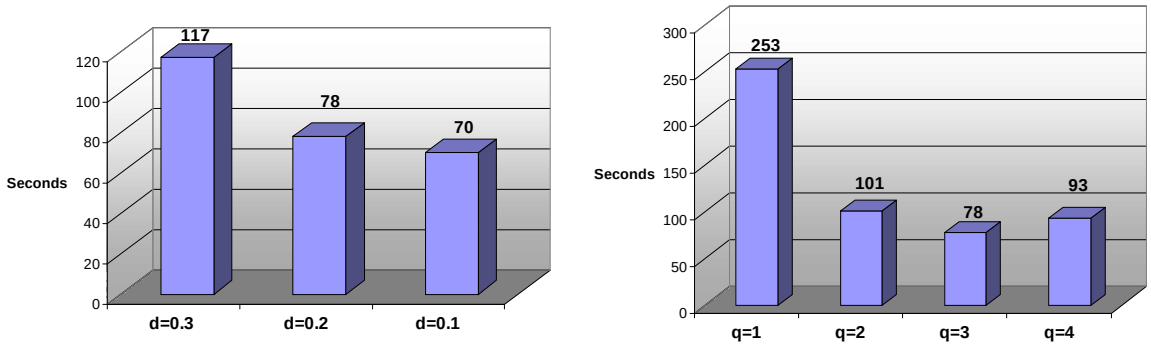


Fig. 9. Filtering test

Effect of filters and indexes We also examine how effective each filter, each index and each combination of filters and indexes is by running different queries, enabling different filters and indexes each time, and measuring the running time and the size of the candidate set. Due to the lack of space, we only show the experiments running time for full matches. We conduct two experiments: test1 on Collection2 and test2 on an extended version of Collection2 with 61420 reference sentences (two times that of Collection2) and 50 query sentences. Figure 9 shows that enabling all filters reduces the response time from 258 to 78 seconds for Collection2. The presence of indexes further improves the system performance as shown by test2 where we compare the response time with indexes enabled and disabled.

Finally, subfigure 10(b) proves that $q = 3$, adopted in our experiments, is the best assignment for q. As far as d is implied, we only performed tests with d less or equal to 0.3 (see subfig. 10(a)) since greater values would have led to almost useless suggestions. Indeed, translators usually set the value of d from 0.2 to 0.1.



(a) d from 0.1 to 0.3

(b) q from 1 to 4

Fig. 10. Response time for various distance threshold d and q-gram length

5 Conclusions

In this paper we presented an approach for searching similar sentences and parts of them in a translation memory. The experiments we conducted show the effectiveness of the underlying similarity measure and the efficiency of the SQL mapping together with filtering techniques. Full and partial search have been implemented in an environment, named EXTRA (EXample based TRanslation Assistant) [5], where innovative techniques were also exploited for phrase and word alignment. As a final remark, we are interested in improving the proposed techniques, in particular partial search, and we will go further on this direction in the future.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
2. R.D. Brown. Example-Based Machine Translation in the Pangloss Systems. In *Proc. of 16th Int'l Conf. on Computational Linguistics*, pages 169–174, 1996.
3. Atril Deja Vu - Translation Memory and Productivity System. Home page http://www.atril.com.
4. L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate String Joins in a Database (Almost) for Free. In *Proc. of 27th Int'l Conf. on Very Large DataBases (VLDB)*, 2001.
5. R. Martoglia. EXTRA: Progetto e Sviluppo di un Ambiente per Traduzioni Multilingua Assistite (in italian). Tesi di laurea, Università degli studi di Modena e Reggio Emilia, 2000/2001.
6. M. Nagao. *A Framework of a Mechanical Translation between Japanese and English by Analogy Principle*. Nato Publications, 1984.
7. G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.
8. S. Sato. Example Based Translation of Technical Terms. In *Proc. of 5th Int'l Conf. on Theoretical and Methodological Issues in Machine Translation*, 1993.
9. H. Sumita, O. Furuse, and H. Iida. An Example Based Disambiguation of Prepositional Phrase Attachment. In *Proc. of 5th Int'l Conf. on Theoretical and Methodological Issues in Machine Translation*, 1993.
10. E. Sutinen and J. Tarhio. On Using q-gram Locations in Approximate String Matching. In *Proc. of 3rd Annual European Symposium*, 1995.
11. Trados Team Edition - Translation Memory Technologies. Home page http://www.trados.com.
12. E. Ukkonen. Algorithms for Approximate String Matching. In *Information and Control*, volume 64, pages 100–118, 1985.