

EAMT Sponsorship of Activities
Call for Proposals 2012
Small research & development project

Qualitative Search of MT Errors

Meritxell González *Laura Mascarell* *Lluís Màrquez*
{mgonzalez, lmascarell, lluism}@lsi.upc.edu

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Final Report

January 2014

Contact information:

<i>Meritxell González</i>	<i>Lluís Màrquez</i>
Postdoc Researcher	Associate Professor
Tel: +34-93-4137950	Tel: +34-93-4137866
mgonzalez@lsi.upc.edu	lluism@lsi.upc.edu

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Edifici Omega - Campus Nord UPC
Carrer Jordi Girona Salgado 1-3
08034, Barcelona, Catalonia, Spain

1 Executive summary

This document is the final report of the EAMT-funded project “*Qualitative Search of MT Errors*”. It reports the activities carried out during the second term of the project and reviews the overall achievements accomplished during the whole project.

This project focuses on the development of a new open tool designed to help MT developers in their evaluation tasks. The purpose of the tool is to facilitate the qualitative analysis of the translation output. To this end, we developed the *t*SEARCH tool, a web-based application that provides a query language and search capabilities for performing complex searches over collections of translation cases, which are evaluated with a large set of diverse MT quality measures.

The activity started on November 2012 and it was planned to span 8 months from the start. However, the final actions addressed in the system were finished in December 2013. The main achievements of the project during the whole period of the project, until December 2013, are listed below (items marked with * correspond to the second period of the project, that is, those not covered in the mid-term report):

1. Design and development of the *t*SEARCH tool, which is described in Section 2.
2. Web-based interface, which supports all planned functionalities. The application is publicly accessible on-line¹, and a brief demonstration of its most important features is given in the demonstrative video².
3. Research publication at ACL 2013 [GMM13]. The final version of the *t*SEARCH tool was presented in the “System demonstration” track in the ACL conference, held in Sofia on August 2013.
4. *Analysis of the performance of the *t*SEARCH tool in terms of hardware resources and response time, described in Section 3.
5. *Master Thesis dissertation of Laura Mascarell, which is primarily focused on her work in this project [Mas13]³.
6. *Study on the usability of the interface and user satisfaction with respect to the capabilities of the *t*SEARCH tool. In particular, we set up an scenario for testing and collecting user feedback on usefulness, facility of use, user friendliness, satisfaction, etc. The results of this study are discussed in Section 4.
7. *Preparation of a journal paper on the ASIYA PLATFORM, including *t*SEARCH and the performance and user satisfaction analyses. We are currently working on this article as a culmination of the work carried out in the project. We plan to submit it during the next weeks.

¹<http://asiya.lsi.upc.edu/demo/>

²<http://www.youtube.com/watch?v=4IQpdVsorKw>. The demonstrative video has been updated to cover also the latest developments and improvements of the *t*SEARCH tool.

³<http://upcommons.upc.edu/pfc/handle/2099.1/19777>

8. *Additionally, we are offering a tutorial on “MT and its evaluation” at LREC 2014. As an effort to disseminate the abilities and usefulness of *t*SEARCH, we are planning a hands-on activity to show how to take advantage of the ASIYA TOOLKIT and *t*SEARCH tool.

The budget for the completion of the project is divided into 8 person months (PM) for a full time student scholarship and 2 PMs that correspond to the supervision and dissemination tasks carried out by this project team (Lluís Màrquez and Meritxell González). We requested from EAMT the total amount of 8,000 EUR to partially support a student scholarship. These funds have been invested, as planned, for the recruitment of Laura Mascarell (a student enrolled in the Master’s Degree in Information Technology at UPC) from December 2012 until July 2013. She presented her Master’s Thesis in October 2013.

Regarding to the work plan initially envisaged in the proposal, the main milestone “M2: Prototype”, planned due M8, was fulfilled with all the tasks executed before the demonstration of the *t*SEARCH tool at the ACL-2013 conference. In addition, from M9 to M13 we executed additional evaluation tasks regarding the performance of *t*SEARCH and the usability of its interfaces. Although these tasks were not planned initially, we envisaged the need for these evaluations in order to assess to which extent the current version of the tool suits the actual needs of MT developers. The performance evaluation was finished due M11 and the usability test was completed on M13. After the completion of all these tasks, we can conclude that all the goals planned in the initial proposal have been achieved successfully. The additional evaluation tasks lead us believe we have developed a strong and useful tool. We paid close attention to the feedback gathered from the participants in the test and we will focus on addressing the main identified weaknesses in the near future work.

Next, we provide an extended version of deliverable D2 “Technical Description and User Manual” (formed by Section 2 and Appendix A). This deliverable also describes the performance analysis (Section 3) and the conclusions of the study on system usability and user satisfaction (Section 4).

2 The *t*SEARCH Tool: Technical Description

This section presents the *t*SEARCH tool, a web-based application that aims to alleviate the burden of manual analysis that translators and developers have to conduct to assess the translation quality aspects involved in the development of MT systems.

*t*SEARCH offers a graphical search engine with the mechanisms to do a quick qualitative evaluation of the translations quality. The system core retrieves all translation examples that satisfy certain properties related to the evaluation scores and/or the linguistic structures. The query language designed is simple and flexible, and it allows to combine many properties to build sophisticated searches.

As a toy example, consider for instance an evaluation setting with two hypothetical systems, s_1 and s_2 , and two evaluation metrics m_1 and m_2 . Assume also that m_1 scores s_1 to be better than s_2 in a particular test set, while s_2 predicts just the contrary. In order to analyze this contradictory evaluation one might be interested in inspecting from the test set the particular translation examples

that contribute to these results, i.e., text segments t for which the translation provided by s_1 is scored better by m_1 than the translation provided by s_2 and the opposite behavior regarding metric m_2 . *t*SEARCH allows to retrieve (visualize and export) these sentences with a simple query in a fast time response. Then, the search could be further constrained, by requiring certain margins on the differences, by including other systems or metrics, or by requiring some specific syntactic or semantic constructs to appear in the retrieved examples.

The functionalities for the application are i) to find concrete translation examples that have specific linguistic characteristics within translation test sets, ii) to obtain a number of linguistic analysis of the sentences and iii) clearly spot the divergences with respect to other candidates or reference translations that make them good or bad ones. Such a tool will help MT developers to speed up the development cycle and increase the quality of their systems analysis.

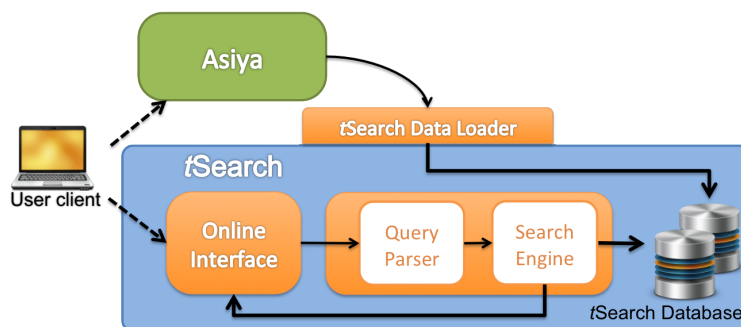


Figure 1: *t*SEARCH architecture

The *t*SEARCH architecture consists of the three components illustrated in Figure 1: 1) the storage system based on *NoSQL* technology that stores the resources generated by ASIYA, 2) the *t*SEARCH core, composed of a query language and a search engine able to look through the information gathered in the database, and 3) a graphical user interface that assists the user to write a query, returns the set of sentences that fulfill the conditions, including the relevant information related to the query and the segments, and allows to export these results in XML format.

The databases (Section 2.1) are fed through the *tSearch Data Loader* API used by ASIYA. At run-time, during the calculation of the measures, ASIYA *inserts* all the information being calculated (i.e., metrics and parsers' output) and a number of precalculated variables, such as average, mean and percentiles. These operations are made in parallel, which makes the overhead of filling the database marginal.

The query parser (Section 2.2) receives the query from the on-line interface and converts it into a binary tree structure where each leaf is a single part of an operation and each node combines the partial results of the children. The search engine obtains the final results by processing the tree bottom-up until the root is reached.

2.1 Data Representation, Storage and Access

CF keys		CF values			
Testsuite_key + BLEU		0.2	...	0.6	1.0
	$s_1\{seg3, seg5\}$, $s_2\{seg3, seg5\}$...	$s_1\{seg8\}$	$s_2\{seg8\}$
Testsuite_key + ULC		0.0	...	0.8	0.85
	$s_1\{seg2\}$...	$s_1\{seg6\}$, $s_2\{seg7\}$	$s_2\{seg5\}$, $s_2\{seg8\}$

(a) Scores Column Family

CF keys		CF values						
Testsuite_key + BLEU		MIN	MAX	AVG	MEDIAN	PERC(1)	...	PERC(50)
		0.0	1.0	0.34	0.27	0.0-0.1	...	0.34-0.36
Testsuite_key + ULC		MIN	MAX	AVG	MEDIAN	PERC(1)	...	PERC(50)
		0.1	1.0	0.83	0.87	0.1-0.2	...	0.83-0.83

(b) Statistics Column Family

CF keys		CF values				
Testsuite_key + SP		DT	NN	VBZ	JJ	NNP
	$s_1\{seg1, seg2\}$, $s_2\{seg1, seg2\}$		$s_2\{seg1\}$	$s_1\{seg1, seg2\}$	$s_2\{seg1\}$...
Testsuite_key + CP		ADJP	CONJP	ADVP	PP	WHPP
	$s_1\{seg3\}$	$s_1\{seg1\}$, $s_2\{seg2, seg5\}$	$s_2\{seg1\}$	$s_1\{seg1, seg2, seg3\}$

(c) Linguistic Elements Column Family

CF keys		CF values					
Testsuite_key + DP		N_nsubj_V	D_nsubj_V	C_cc_V	I_prep_N	M_aux_V	N_pobj_I
	$s_1\{seg1, seg2, seg3\}$	$s_2\{seg4\}$	$s_1\{seg1, seg2\}$	$s_2\{seg1\}$	$s_2\{seg3\}$
Testsuite_key + SR		A0	A1	AM-TMP	AM-LOC	AM-ADV	R-AM-LOC
	$s_1\{seg1\}$, $s_2\{seg2, seg5\}$	$s_1\{seg2\}$	$s_1\{seg1, seg2\}$, $s_2\{seg3, seg5\}$	$s_2\{seg1\}$	$s_1\{seg1, seg2\}$, $s_2\{seg1, seg2\}$

Figure 2: *t*SEARCH data model. s_n stands for a translation system name and $segN$ represents a segment number. SP stands for Shallow Parsing and the column keys are PoS labels given by the parser. CP stands for Constituency Parsing and the column keys are syntactic labels given by the parser. DP stands for Dependency Parsing and the column keys are N-grams of PoS labels and dependency relations given by the parser. SR stands for Semantic Roles and the column keys are the the arguments of the predicate.

The amount of data generated by ASIYA can be very large for test sets with thousands of sentences. In order to handle the high volume of information, we decided to use the Apache Cassandra database⁴, a *NoSQL* (also known as *not only SQL*) solution that deals successfully with this problem.

It is important to remark that there is no similarity between *NoSQL* and the traditional relational database management system model (RDBMS). Actually, RDBMS uses SQL as its query language and requires a relational model, whereas *NoSQL* databases do not. Besides, the *t*SEARCH queries can be complex, with several conditions, which makes RDBMS perform poorly due the complexity of the tables. In contrast, *NoSQL* databases use *big-tables* having many querying information precalculated as key values, which yields for direct access to the results.

The Cassandra data model is based on *column families* (CF). A CF consists of a set of rows that are uniquely identified by its key and have a set of columns as values. So far, the *t*SEARCH data model has the three CFs shown in Figure 2. The *scores* CF in Figure 2(a) stores information related to metrics and score values. Each row slot contains the list of segments that matches the column key. The *statistics* CF in Figure 2(b) stores basic statistics, such as the minimum, maximum, average, median and percentiles values for every evaluation metric. The CF having the *linguis-*

⁴<http://cassandra.apache.org/>

tic elements in Figure 2(c) stores the results of the parsers, such as part-of-speech, grammatical categories and dependency relationships.

One of the goals of *NoSQL* databases is to obtain the information required in the minimum access time. Therefore, the data is stored in the way required by the *tSEARCH* application. For instance, the query $BLEU > 0.4$ looks for all segments in the test suite having a BLEU score greater than 0.4. Thus, in order to get the query result in constant time, we use the metric identifier as a part of the key for the *scores* CF, and the score 0.4 as the *column* key.

Query Type	Query Structure	Example	Description
Arithmetic Comparison	METRIC op <i>real</i>	$BLEU > 0.4$	Operators: $>$, $<$, $>=$, $<=$, $=$
Statistical Functions	METRIC op SF	(1) $BLEU > AVG$ (2) $BLEU > TH(40)$	Use precalculated statistical variables: average, median, min, max, percentiles [1..100], thresholds.
Range	METRIC IN [x,y] METRIC IN Q(n) METRIC IN PERC(m,m)	(1) $BLEU$ IN [0.2,0.3] (2) $BLEU$ IN Q(4) (3) $BLEU$ IN [PERC(2,10),PERC(3,10)]	In a range of values that can be predefined by the user or statistical values precalculated by the system.
Linguistic Elements	LE[type(item+)+]	(1) LE[SP(NN), DP(conj), CP(PP)] (2) LE[SP(Fz, VC, Vai)] (3) LE[DP(VBG,dobj,NNS)] (4) LE[SR(want,A0,A1,AM-TMP)]	'type' is the type of linguistic processor: shallow (SP), constiency (CP), dependency (DP), semantic (SR). 'item' is a linguistic element that belongs to the type of processor: pos, categories, relationships, roles, as in example (1). It's possible to ask for N-grams, as shown in the example (2), a chain of pos and dep. relations (3), or a list of role arguments (4).
Logical Composition	Query1 AND/OR Query2	$BLEU > 0.5$ AND $-PER < 0.7$	Logical operators to concatenate several conditions

Query type	Example	Description
System-level queries	$s_1[BLEU] > 0.4$	Segments from system s_1 translations that have a BLEU score above 0.4
Document level queries	(1) $news[BLEU] > 0.3$ (2) $s_1[news[BLEU]] > 0.3$	Segments from the <i>news</i> document (1) (and s_1 translations (2)) having a BLEU score above 0.3
Groups of Systems and/or Metrics	<ul style="list-style-type: none"> $s_{rb} = \{s_1, s_2\}$ $LEX = \{BLEU, NIST\}$ $SYN = \{CP-Op(*), SP-Op(*)\}$ (1) $s_{rb}[LEX] > AVG$ (2) $((s_{rb}[LEX] > AVG) \text{ OR } (s_3[LEX] < AVG)) \text{ AND } ((s_{rb}[SYN] < AVG) \text{ OR } (s_3[SYN] > AVG))$	(1) Segments from $s_{rb} = \{s_1 \text{ and } s_2\}$ translations that have BLEU and NIST scores above the average (2) Segments from s_{rb} having good scores for lexical measures and bad scores for syntactic measures, and same segments for s_3 having bad and good scores for lexical and syntactic measures, respectively.

Figure 3: (top) Query operations and functions, (bottom) Queries for group of systems and metrics

2.2 The Query Language and Parser

The Query Parser module is one of the key ingredients in the *tSEARCH* application because it determines the query grammar and the allowed operations, it provides a parsing method to analyse any query and it produces the machine-readable version of the query semantics. It is also necessary in order to validate the query.

There are several types of queries, depending on the operations used: arithmetic comparisons, statistical functions (e.g., average, quartiles), range of values, linguistic elements and logical operators. Furthermore, the queries can be applied at segment-, document- and/or system-level, and it is even possible to create any group of systems or metrics. This is useful, for instance, in order

to limit the search to certain type of systems (e.g., rule-based vs. statistical) and specific metrics (e.g., lexical vs. syntactic). All possible query types are listed in Figure 3 and the details on the operators and its semantics are given in the User Manual (Section A) and the related publications ([GMM13] and [Mas13]).

2.3 On-line Interface and Export of the Results

*t*SEARCH is fully accessible on-line through the ASIYA ON-LINE INTERFACE. The web application runs ASIYA remotely, calculates the scores and fills the *t*SEARCH database. It also offers the chance to upload the results of a test suite previously processed. This way it feeds the database directly, without the need to run ASIYA.

Anyhow, once the *t*SEARCH interface is already accessible, one can see a tools icon on the right of the search box. It shows the toolbar with all available systems in the testbed, calculated metrics, and search functions and operations. The search box allows to query the database using the query language described in Section 2.2.

The screenshot displays the *t*SEARCH web interface. At the top, there are dropdown menus for Metrics, Groups, Systems, and Docs. Below these are buttons for MIN, MAX, AVG, MEDIAN, AND, OR, TH(), PERC(), IN(), and Q(). A search box contains the query `LE[DP(D, det, NN, nsubj, VB)] AND BLEU > AVG`. A **Search** button is next to it. Below the search box, a **Search Info** box shows `BLEU AVG = 0.1794`. To the right of the search box is a toolbar with icons for various systems and metrics. Below the search box, there are tabs for **All**, **By System**, and **By Segment**. An **Export all** button is also present. The main results area shows `LE[DP(D, det, NN, nsubj, VB)] AND BLEU > AVG` with **2 results**. A **Show:** dropdown is set to **1**. Below this, there are checkboxes for **D** (checked), **det** (checked), **NN** (checked), **nsubj** (checked), and **VB** (checked). A **Source:** dropdown is set to **src**. The results are displayed in a table with columns for **Reference** and **Segment**. The first result is `ref` with the text `The Spanish affiliate of the Disney Channel will debut fiction on March 4.`. Below this, there are two rows for **System** and **Translation**. The first row is `babelfish` with the translation `The Spanish branch of Disney Channel will release next the 4 of March the first totally Spanish fiction product.`. The second row is `systran` with the translation `The Spanish branch of Disney Channel will wear for the first time next the 4 of March the first totally Spanish fiction product.`. A tooltip is visible over the `Source:` dropdown, showing `Source: src`, `Reference: ref`, `System: babelfish`, `Document: UNKNOWN_DOC`, `Num. Segment: 3`, `Scores`, and `BLEU: 0.2915`.

Figure 4: The *t*SEARCH Interface

After typing a query, the user can navigate the results using three different views that organize them according to the user preferences: 1) *All segments* shows all segments and metrics mentioned in the query, the segments can be sorted by the score, in ascendant or descendent order, just tapping

on the metric name; 2) *Grouped by system* groups the segments by system and, for each system, by document; 3) *Grouped by segment* displays the segment organization, which allows an easy comparison between several translations. Each group contains all the information related to a segment number, such as the source and the reference sentences along with the candidate translations that matched the query.

Additionally, moving the mouse over the segments displays a floating box as illustrated in Figure 4. It shows some relevant information, such as the source and references segments, the system that generated the translation, the document which the segment belongs to, and the scores.

Finally, all output data obtained during the search can be exported as an XML file. It is possible to export all segments, or the results structured *by system*, *by segment*, or more specific information from the views.

3 *t*SEARCH Performance

This section details several experiments that we have run to analyze the performance of the *t*SEARCH tool under different testbed conditions.

Our purpose was, on the one hand, to analyze the convenience of calculating some data on demand when the user really requires it, either because the time required to calculate these data is too large, or because it is unlikely to be used in most evaluation schemata. On the other hand, we also want to compare the performance of the system for different testbed sizes, being our variables: 1) the number of segments, 2) the number of systems and 3) the query type.

The reason is that the size of the testbed, and the time required to process it, depends directly on the number of systems and the number of segments, i.e., a testbed with 1,000 segments and 3 systems, is analysing a total of 3,000 segments. Our experimental results correspond to testbeds extracted from the WMT10 Translation Task [CBKM⁺10] with corpora subsets of 100, 200, 500, 1,000 and 2,000 segments and 1, 2, 3, 5, 10 and 14 systems. Note that the amount of data generated (results of the linguistic processors and scores) that needs to be stored in the *t*SEARCH database is huge.

One of the main aspects we are worried about is the time that a user has to wait until the testbed is completely inserted into the *t*SEARCH database. Here we can distinguish two different approaches. In the first one, the pre-calculated (*P*) approach, all the information related to the evaluation scheme at the time to load a testbed is pre-calculated. This approach increases the elapsed *inserting* time, but then almost all information is already calculated and the query results are quickly computed. The second approach, not pre-calculated (*NP*), consists on not to pre-calculate the most expensive computations (namely, the linguistic elements). In this approach, the user decides to calculate these data exactly when it is needed. Moreover, other metrics that evaluate specific elements are not pre-calculated, since they are unlikely needed in most evaluation scheme, i.e., NE-Oe (MONEY)⁵.

⁵NE-Oe(MONEY) reflects lexical overlap between name entities of type 'MONEY'.

The following sections describe the experimental results regarding the insertion of a testbed (Section 3.1) in the two scenarios (the not pre-calculated *NP* and the pre-calculated *P* scenarios) and some queries worth to detail (Section 3.2).

3.1 Loading the Testbed

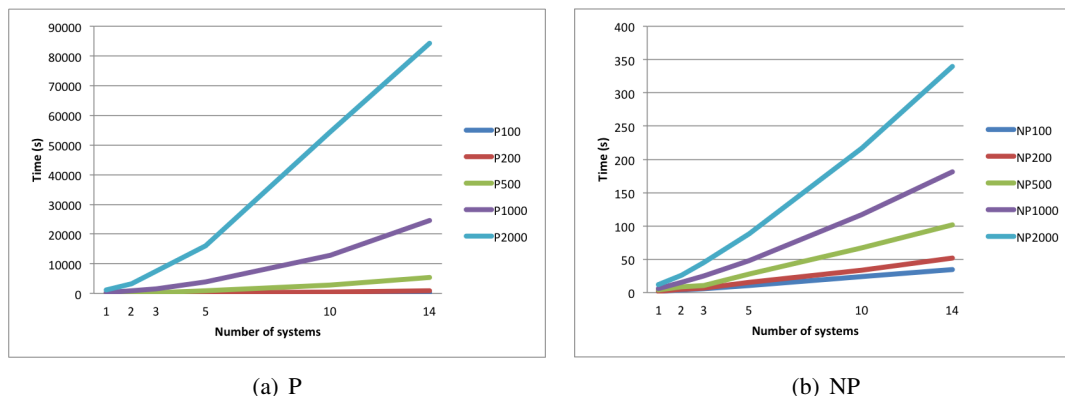


Figure 5: Time in seconds of running the *initialize* operation over several testbeds with corpora of 100, 200, 500, 1,000 and 2,000 segments and 1, 2, 3, 5, 10 and 14 systems; all them evaluated over 40 different metrics. The Figure on the left shows the results of initializing the different testbeds in the scenario *P*, i.e., pre-calculating everything. The Figure on the right displays the results of initializing the same testbeds in the scenario *NP*, i.e., without processing the files related to linguistic elements nor the less common metrics.

The *tSEARCH* Data Loader has two main operations *initialize* and *insert* that are called at the time to load a testbed.

On the one hand, *initialize* computes all the statistical information related to the metric scores, i.e., average, median, minimum, maximum and percentiles values. Furthermore, it computes all the linguistic elements information at the *P* scenario. The cost of reading the evaluation files and processing them is high and proportional to the number of metrics, systems and segments. For a fixed number of metrics (40), Figure 5 shows the computation time required to obtain the results having a testbed with different number of systems and segments. Note that the cost of computing the linguistic elements data (*P*) is by far higher than only computing the statistical information (*NP*).

On the other hand, the *insert* operation is in charge to send all the data previously calculated to the database connector in order to be inserted in the DB tables. The experimental results obtained from the different testbeds evaluated over 40 different metrics are shown in Figure 6. At the *P* scenario, the computational time tends to increase slightly along with the number of systems. Considering the number of corpora segments, again the time barely increases as we increase the

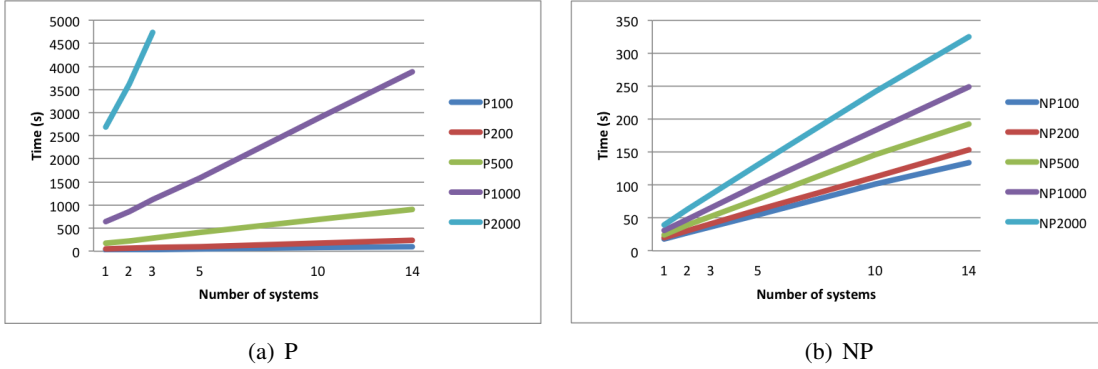


Figure 6: Time in seconds of running the *insert* operation over several testbeds with corpora of 100, 200, 500, 1,000 and 2,000 segments and 1, 2, 3, 5, 10 and 14 systems; all them evaluated over 40 different metrics. The Figure on the left shows the results of inserting the different testbeds in the scenario *P*. The Figure on the right displays the results of inserting the same testbeds in the scenario *P*.

number of segments. However, at *NP* scenario, the angle of slop increases excessively and mostly for those testbeds with a corpora of 1,000 segments or more. As it can be seen, the required time is almost constant despite of increasing the number of systems. The gap between *P* and *NP* approaches is the time that *tSEARCH* needs to insert the data regarding linguistic elements.

3.2 Query Analysis

Once the testbed is loaded, the system is ready to answer queries. We assess the performance analysing the response time required to produce the output for the following queries:

- BLEU \geq MIN: gets all testbed segments.
- The query BLEU \geq TH(20): needs to compute the threshold value at run-time from the percentiles that were pre-calculated and then, gets the top 80% of the segments.
- LE [DP (*, *, V)]: is one of the most time-consuming Linguistic-based queries. It has to compute all possible n -grams on DP labels.

The two first queries do not distinguish between scenarios *P* and *NP* because linguistic elements are not involved. The latter is related to linguistic-based queries, so that we show both, the computational time to get the output, and the time to calculate on demand all data related to the query. This test and also the results concerning the insertion of a testbed as described in previous Section 3.1, are **key ingredients on the decision of whether *tSEARCH* should pre-calculate linguistic elements or not**. An extended list of queries are analysed in [Mas13].

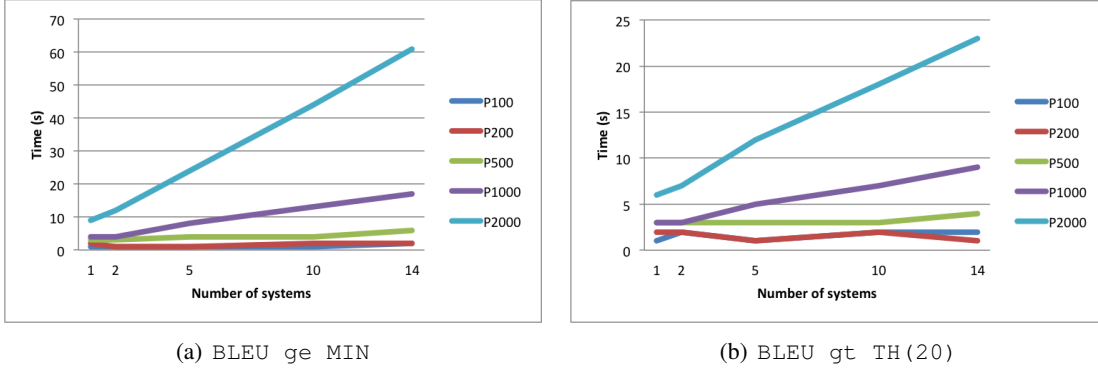


Figure 7: Time in seconds of executing the queries BLEU ge MIN (on the left) and BLEU gt TH(20) (on the right) over several testbeds inserted with corpora of 100, 200, 500, 1,000 and 2,000 segments and 1, 2, 3, 5, 10 and 14 systems; them all evaluated over 40 different metrics.

3.2.1 Getting all segments

In this experiment we want to know the time required by t SEARCH to prepare an output that contains all segments. For this purpose, we asked for the query BLEU ge MIN. Figure 7(a) presents the results obtained. Note that the maximum time value obtained is 61 seconds with a test suite of 14 systems and a corpora of 2,000 segments, i.e., 61 seconds to get an output of 28,000 segments. However, t SEARCH takes less than 5 seconds for smaller testbeds of up to 7,000 segments in total.

3.2.2 The TH() function

The main goal asking for the query BLEU ge TH(20) is to compute the time to obtain the 80% of the segments and observe if the computation of a function (here the value of the threshold), increases the computational time significantly. We have to take into account that thresholds, as quartiles, are calculated on run-time using the percentiles' values. As it can be seen in Figure 7(b) the time to get the 80% of the segments is slightly lower than getting all segments. This is because the accessing time to the DB is constant, but the number of segments returned is smaller.

3.2.3 Dependency Parsing Query

The Dependency Parsing Query is, by far, the most expensive computation on demand as it can be seen in Figure 8. Since this is a linguistic-based query, we show also the difference between the P and NP scenarios, i.e., the time required to compute the data requested *on demand* 8(b). When t SEARCH computes the data regarding dependency relationships, it processes the corresponding files and computes groups of three elements where the first one is the part-of-speech of a head, e.g., N (noun); the second one is the dependency relation, e.g., `nsubj` (nominal subject); and the last

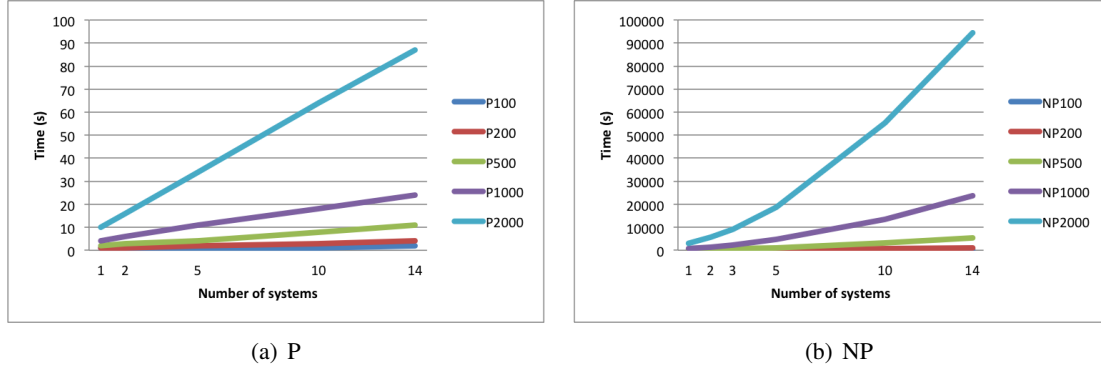


Figure 8: Time in seconds of computing the data needed to execute the dependency parsing (DP) function. The Figure on the left shows the time in seconds to retrieve the results concerning the query $LE [DP (*, *, V)]$ when all the data needed was already computed. The Figure on the right shows the time to process, insert the data from dependency parser output before retrieving the results.

one is the part-of-speech of the dependent, e.g., V (verb). Moreover, we also count the 3-grams for the fine-grained part-of-speech, e.g., NN (noun, singular or mass) or VBZ (verb, third person singular present). Thus, for each word there are four different possibilities, e.g., $\{N, nsubj, V\}$, $\{N, nsubj, VBZ\}$, $\{NN, nsubj, V\}$, $\{NN, nsubj, VBZ\}$. Furthermore, we also save the *any* option represented by the asterisk, e.g., $\{N, *, V\}$, $\{*, nsubj, V\}$, $\{N, nsubj, *\}$. As a result, for each word we compute 18 different variants which is the reason of the high cost of its computation. Fortunately, once the data is computed, the time to obtain the results remains constant and it is completely reasonable, as shown in Figure 8(a).

4 Usability Analysis

The purpose of the *t*SEARCH application is to assist MT developers to discover the translations in the testbeds that match any MT quality criteria required. However, the design and development of interactive interfaces is a difficult task. In general, even though a tool may provide convenient functionalities, the usability aspects of the user interfaces are crucial to engage users to adopt a new tool. In order to estimate to which extent the tool developed serve to this end, we engaged a number of target users to participate in the HCI test described in this section. The purposes of this experiment are to assess the usability aspects of the interface, detect further needs not yet addressed and tackle the weaknesses found according to the results of the user interactions and the feedback gathered from the participants.

The participants had to complete an experiment that namely consisted of three parts: First, there was a small video-training that shows the basic functionalities of the tools and how to use

them. The second part comprises several test scenarios, listed in Table 1, in which the user have to perform some actions, analyse some evaluation results and take some conclusions across the diverse interfaces offered by the ASIYA PLATFORM. Finally, the participants answered a questionnaire where they could express their satisfaction with regard several aspects of the interfaces. They could also give comments and suggestions, which were useful for clarifying the questionnaire answers and getting a qualitative analysis of the user perception. The whole process was completely anonymous.

Next, Section 4.1 gives the details of the experiment design and Section 4.2 analyses the results of the experiment.

S1	User training, two videos
S2	Download the testbed
S3	Navigation bar guidelines
S4	ASIYA - Set up the data format and upload the testbed
S5	ASIYA - Metric selection
S6	ASIYA - Run and wait
S7	ASIYA - Analyze the scores table
S8	ASIYA - Analyze the information related to Named Entities
S9	<i>t</i> SEARCH - Start the tool
S10	<i>t</i> SEARCH - Segment level query
S11	<i>t</i> SEARCH - System selection to segment level query
S12	<i>t</i> SEARCH - Metric selection to segment level query
S13	<i>t</i> SEARCH - Linguistic elements
S14	<i>t</i> SEARCH - System and metrics group creation
S15	<i>t</i> SEARCH - User choice
S16	Questionnaire

Table 1: The Test Scenarios

4.1 Experiment design

We established several mechanisms to track and register the user actions. On the one hand, we collected automatic cues directly from the interfaces. We logged the time that users spend in each scenario and the sequence of interactions with the graphical interface, such as uploaded documents, functionality launched (buttons) and commands written. We also logged the queries written, whether they were correct, how long the system takes to calculate the output and the number of results returned. Then, we wanted to obtain the perception of the users in relation to each proposed scenario. We required them to indicate, right after each scenario, if they thought they had *solved* it correctly or they had to *skip* it (e.g., because it was too difficult or it was taken too long to solve). Upon the completion of the scenarios, they answered the questionnaire and could wrote their feedback and further suggestions.

4.1.1 The Questionnaire

The design of the questionnaire is based on the PSSUQ, described in [Lew95], and adapted to our testing setting, where we are evaluating two tools altogether. Our adapted questionnaire (in Table 2) had 27 questions divided in three sections: 1) the ASIYA ON-LINE INTERFACE for evaluation, 2) the *t*SEARCH tool for error analysis, and 3) the overall perception of the ASIYA PLATFORM.

Users were required to think about all the tasks that they had done during the test and indicate how strongly they agree or disagree with the statements in the questionnaire. They were also allowed to leave an answer unselected whenever it did not apply to them (e.g., when it refers to an skipped scenario).

ASIYA ON-LINE INTERFACE	
1.	In the Asiya evaluation interface, I was able to complete the proposed tasks and scenarios efficiently
2.	The Asiya evaluation interface was simple to use
3.	I felt comfortable using the Asiya evaluation interface
4.	Learning to use the Asiya evaluation interface was easy
5.	Whenever I made a mistake, I could recover easily and quickly
6.	The information provided (such as on-line help, on-screen messages and other documentation) was clear
7.	The information I needed was easy to find
8.	The evaluation results provided were easy to understand
9.	The Asiya evaluation interface includes all the functions and capabilities I expect it to have
10.	I think the Asiya evaluation interface is useful to help MT developers to analyze their systems' performance
<i>t</i> SEARCH INTERFACE	
11.	In the <i>t</i> Search tool, I was able to complete the proposed tasks and scenarios efficiently
12.	The <i>t</i> Search tool was simple to use
13.	I felt comfortable using the <i>t</i> Search tool
14.	Learning to use the <i>t</i> Search tool was easy
15.	Whenever I made a mistake, I could recover easily and quickly
16.	In the <i>t</i> Search tool, the information provided (such as on-line help, on-screen messages and other documentation) was clear
17.	The information I needed was easy to find
18.	The search results provided were easy to understand
19.	The <i>t</i> Search tool includes all the functions and capabilities I expect it to have
20.	I think the <i>t</i> Search tool is useful to help MT developers to analyze their systems' performance
OVERALL	
21.	Overall, I am satisfied with how easy it is to use this system
22.	Overall, I believe I could become productive quickly using this system
23.	Overall, the system gave error messages that clearly told me how to fix problems
24.	Overall, the organization of the information on the system's screens was clear
25.	Overall, the interface of this system was friendly
26.	Overall, I am satisfied with this system
27.	Overall, I think I will use this evaluation framework in the future for my MT developments

Table 2: The adapted questionnaire.

4.1.2 The participants

In general, the experiments that involve real users are hard to implement, especially when the purpose of the experiment is to understand how difficult is to use a tool and how long it takes to learn to use it, because in this type of experiments participants can get involved only once. For this test, we established a conservative setting that consisted of 3 rounds of at least 5 user each (the motivation behind is analysed in [NL93]). The first round allow us to detect misleadings in the overall testing setting and correct them before continue with the test. The participants in this round were selected from developers that work in our department that are somehow related to the MT field (*indoor*). For the other two rounds, we collected a list of scientist, all around the world, related to the MT research community (*outdoor*). During the second round we wanted to assess that the problems found in the first round were amended and we did not introduced new ones. For this reason, we also selected a reduced number of participants from the bunch of real users. The third and final round was launched to the rest of researchers to their personal email and they were encouraged to distribute the test among their students and collaborators. Among the total number of users that subscribed to the test, only 22 of them completed all the scenarios and answered the questionnaire. As discussed next, in Section 4.2, there were two main differences between *indoor* and *outdoor* participants. First, we believe that internal users are less critical against the tool, and second, they had attended at least one talk (seminar or thesis presentation) about the tool. These differences are statistically significant when scoring the *t*SEARCH tool, so in the next discussions we show the results for only the *outdoor* participants and them all (*indoor* and *outdoor*).

4.2 Analysis of the user experience

As mentioned above, 22 participants completed the test and questionnaire, 9 of them were *indoor* and 13 *outdoor*. Since the ASIYA ON-LINE INTERFACE was released some time ago, we are aware that some of the users already knew the ASIYA tool and some even use it actively (both, the command-line and the on-line version). In contrast, none of them had used before the *t*SEARCH tool.

4.2.1 The questionnaire results

Table 3 summarizes the results obtained from the questionnaires. First, in order to validate the questionnaire we calculated the overall satisfaction as the mean of the ASIYA and *t*SEARCH questions (Q1-Q20) and we compare them with the overall scores given by the participants (Q21-Q27). The means of the two series are slightly different, but the significance test showed that the difference was not statistically significant (t-Test with $\alpha = 0.1$).

In general, users expressed a good satisfaction overall the ASIYA PLATFORM (4.34 and 4.08 over 6.0 points ⁶) and the *t*SEARCH tool (3.65 and 3.21). The ASIYA evaluation tool received

⁶The questionnaire was a 1 to 7-points likert scale for convenience to the users. To calculate the statistics we adapted it to 0 to 6 scale.

	<i>All Participants</i>			<i>Outdoor Participants</i>		
	Overall	ASIYA	<i>t</i>SEARCH	Overall	ASIYA	<i>t</i>SEARCH
Q1-Q20	4.34 ± 1.46	5.02 ± 0.97	3.65 ± 1.56	4.08 ± 1.63	4.93 ± 1.04	3.21 ± 1.67
Q21+Q27	4.44 ± 1.22			4.17 ± 1.19		
SIMPLE	4.36 ± 1.34	5.23 ± 0.81	3.47 ± 1.42	4.14 ± 1.52	5.19 ± 0.75	3.00 ± 1.55
LEARN	4.23 ± 1.54	5.14 ± 0.91	3.36 ± 1.53	3.96 ± 1.68	5.15 ± 0.90	2.77 ± 1.42
CLEAR	4.17 ± 1.47	4.71 ± 0.85	3.00 ± 1.75	3.82 ± 1.50	4.46 ± 0.88	2.69 ± 1.84
USEFUL	4.69 ± 1.29	4.93 ± 1.21	4.39 ± 1.39	4.42 ± 1.47	4.85 ± 1.38	4.00 ± 1.62
EASY	4.36 ± 1.42	4.86 ± 0.99	3.84 ± 1.76	4.02 ± 1.65	4.77 ± 1.14	3.24 ± 1.98
ROBUST	3.89 ± 1.52	4.81 ± 0.98	3.37 ± 1.50	3.78 ± 1.46	4.67 ± 1.07	3.25 ± 1.54
TSKCMPL	4.49 ± 1.31	5.45 ± 0.60	3.41 ± 1.18	4.24 ± 1.38	5.38 ± 0.51	3.00 ± 1.22

Table 3: Summary of the questionnaire answers divided by tool and score category

	<i>Outdoor Participants</i>											
	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Solved	13	13	13	12	12	11	10	9	2	5	11	8
Skipped	0	0	0	1	1	2	3	4	11	8	2	5

Table 4: Number of Solved vs. Skipped test scenarios S4 to S15.

higher scores wrt. to the *t*SEARCH tool, which can be explained by the fact that ASIYA is a matured software. This is also reflected in the feedback given by the users, as discussed below. There is a higher satisfaction of the *indoor* users, as we already expected.

We also analyse each type of score that refers to specific aspects of the interfaces. The most valued aspects of the *t*SEARCH tool were its **usefulness** to analyse MT systems’ outputs and that the results provided were **easy to find and understand**. In general, lower scores are assigned to the **clarity** of the on-line help and on-screen messages and how easy is to **learn** the query language. The latter is further discussed next.

4.2.2 The difficult tasks in the test scenarios

The test was divided into 16 tasks. Tasks S1 to S3 correspond to the instructions, whereas task S16 corresponds to the questionnaire. In the rest of the tasks the user had to figure out how to solve the proposed scenario in the ASIYA ON-LINE INTERFACE (S4 to S8) and the *t*SEARCH INTERFACE (S9 to S10). The participants in the experiment could mark each scenario as *solved* or *skipped*. We logged these flags about task completion only from the *outdoor* participants.

Table 4 and Table 5 show, respectively, the number of solved and skipped scenarios and the time invested in each one. In the ASIYA ON-LINE INTERFACE, only tasks S7, S8 and S9 were skipped by one of the participants (a different one each time). When using the *t*SEARCH tools, they found the proposed scenarios relatively easy except tasks S12 and S13. The analysis of the queries

Scenario	All Participants	Outdoor Participants
S1-S2	10.68 \pm 8.93	9.69 \pm 4.19
S3	1.77 \pm 3.80	2.15 \pm 4.83
S4	1.91 \pm 3.02	3.23 \pm 3.37
S5	1.27 \pm 1.80	2.15 \pm 1.91
S6	4.14 \pm 3.45	2.08 \pm 1.12
S7	4.77 \pm 5.28	2.15 \pm 2.64
S8	3.05 \pm 3.40	5.15 \pm 2.91
S9	0.55 \pm 1.01	0.92 \pm 1.19
S10	4.09 \pm 3.35	4.62 \pm 3.71
S11	2.68 \pm 2.17	2.23 \pm 1.59
S12	5.09 \pm 4.67	6.31 \pm 5.39
S13	3.73 \pm 3.59	4.00 \pm 4.14
S14	2.32 \pm 2.30	1.62 \pm 1.66
S15	2.00 \pm 2.41	1.08 \pm 1.66
S16	5.77 \pm 6.62	9.62 \pm 6.12
TOTAL	54.23 \pm 30.13	57.38 \pm 27.83

Table 5: Time in minutes invested in each test scenario

shows that in S12, even they could write syntactically correct queries, they could not get the results required by the scenario description. In contrast, 8 out of 13 participants needed to skip the task S13 because they got syntactic errors and could not get any result.

In terms of time spent in each test scenario, we find notable differences when comparing *all participants* and *outdoor* ones in the ASIYA tasks (S4 to S8). In contrast, the scenarios S9 to S15 show similar times to both type of users, which was the expected behaviour since the *t*SEARCH was a novel tool for all the participants. *outdoor* users required more time to think about their questionnaire answers.

It is interesting to note that this table shows also that it is a matter of few minutes to analyse the evaluation results of a full 500-sentence and 5-system testbed, and a matter of around 1 hour to evaluate and analyse it in depth.

4.2.3 Learning to write *t*SEARCH queries

Task scenarios S10 to S15 were devoted to write queries using the *t*SEARCH tool. In these scenarios users were required to write queries to analyse the translations in the testbed. If we analyse the number of queries written in each scenario, we can see that most of the users could learn the query language after few trials. Table 6 shows the mean number of *Correct* and *Wrong* queries written by all the users. As it can be noted, the standard deviations are very high, which may be an indication that some users tried harder to get their queries work (*proactive* users). These type of users are especially interesting to analyse the learning curve of the query language. We established a threshold of 6 queries (the limit of the first quartile) to consider a participant as a *proactive* user.

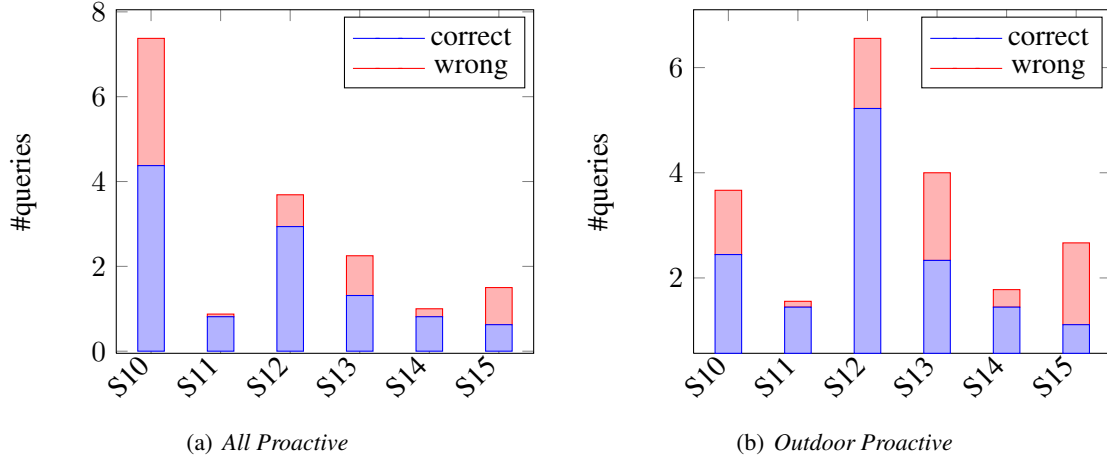


Figure 9: Mean number of *correct* vs. *wrong* queries written for each scenario

	All Participants			Outdoor Participants		
	Max	Mean	Q1	Max	Mean	Q1
QUERIES	43.00	13.14 ± 12.97	3.00	42.00	15.46 ± 11.90	6.00
CORRECT	33.00	8.68 ± 9.25	2.75	33.00	10.77 ± 9.35	3.50
ERRORS	16.00	4.45 ± 4.62	0.00	10.00	4.69 ± 3.47	1.50
%	71%	26 ± 23%	0%	71%	30 ± 20%	14%
RATIO	10.00	1.50 ± 2.30	0.00	10.0	2.15 ± 2.64	0.50

Table 6: Mean number of *Correct* vs. *Wrong* queries in Scenarios S10 to S15

Then, Table 6 shows the mean number of *Correct* and *Wrong* queries written for each scenario. As it can be noted, during the first scenario S10 the users needed to refine their queries several times until they got the right syntax. Then, S11 required just to step up the query, which resulted fairly easy to accomplish. The scenario S12 was more difficult and, as mentioned above, even some users could write correct queries, they could not get satisfactory results. For this reason, they tried several ways to solve the problem. It is interesting to note that most of the queries they wrote were syntactically correct. This may indicate that they understood the way the query language is built, although they may need more time to understand the semantics of the queries. The following scenario S13 was completely different than previous ones. It required the use of LEs (linguistic elements). In general, half of the users could not build correct queries. In contrast, it was fairly easy to create the system- and metric-groups proposed in S14. Finally, in S15 users were required to write free queries using the groups just created and using other optional features respectively. These scenarios are interesting because users tried to obtain the information that might be of interest for them. Some of the queries they wrote were not supported by the system but are fairly interesting

and they will be included in the next revisions of the system.

4.2.4 Feedback and suggestions from the users

At the end of each block of questions in the questionnaire, the participants could write their impressions and clarifications about the tools. Some of them gave also interesting suggestions to improve their capabilities that will be taken into account in the future work. This section summarizes their valuable comments regarding the *t*SEARCH tool.

Namely, all the participants mentioned that they found the tool very powerful and useful. Their concerns were addressed to the way the interface helps writing queries. The mechanisms developed did not help them enough. Few of the users mentioned that they would also appreciate some more help on the syntax and semantics of the queries (for instance the scope of the operators wrt. the metrics selected), and a better approach for the error messages (sometimes were poorly helpful and need to be polished).

In addition, some additional functionalities proposed by the participants will be addressed in the future work. For instance, include the source sentence in the search (the current version analyses the candidates and references), and additional query types to compare two systems and a side-by-side visualization. In fact, a better layout and organisation of the search information provided will be also addressed.

Finally, it is also interesting to note that ASIYA can also deal with human evaluations to perform the metric evaluations. This is an interesting capability for metric-designers and developers. However, the on-line interfaces cannot yet deal with these data. This is also a challenging capability that we will analyse in the near future.

References

- [CBKM⁺10] Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, 2010. Revised August 2010.
- [GMM13] Meritxell González, Laura Mascarell, and Lluís Màrquez. tSearch: Flexible and Fast Search over Automatic translation for Improved Quality/Error Analysis. In *Proc. 51st Meeting of the ACL. System Demonstration*, Sofia, Bulgaria, August 2013.
- [Lew95] James R. Lewis. Ibm computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *Int. J. Hum.-Comput. Interact.*, 7(1):57–78, January 1995.
- [Mas13] Laura Mascarell. tSearch : Leveraging the Automatic Analysis of Machine Translation Evaluation for Fast and Flexible Error Analysis. Master’s thesis, Universitat Politècnica de Catalunya, 2013.
- [NL93] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems*, CHI ’93, pages 206–213, New York, NY, USA, 1993. ACM.

A *t*SEARCH User Manual

A.1 Getting started

Let us introduce the user manual of *t*SEARCH⁷, a web-based application that aids the error analysis stage of machine translation development by facilitating the qualitative analysis of translation quality. The *t*SEARCH interface is accessible at <http://asiya.lsi.upc.edu/demo/> where you can find two ways to access it. The first one consists of evaluating a testbed with ASIYA and once the evaluation is completed, *t*SEARCH appears as one of the tools that the user can run. However, if you have the data already evaluated by ASIYA, the second option allows you to upload the compressed folder that contains the ASIYA evaluation output and start using *t*SEARCH.

A.1.1 Getting to know *t*SEARCH

The Figure 10 describes some of the features available in *t*SEARCH interface:

1. **Toolbar:** use the toolbar to find all metrics, systems and documents, operate with groups, view examples and select the functions and operations available.
2. **Output area:** this area displays the results of your query.
3. **Query input:** use this input box to write your query.
4. **View tabs:** navigate through the different organization views: all segments, by system or by segment.
5. **Info panel:** gives you additional information related to the query such as groups of metrics, systems and documents, and the actual values used for the statistical functions such as MIN, MAX, AVG, MEDIAN, TH () , PERC () or Q () .

⁷There is also a video tutorial available at <http://www.youtube.com/watch?v=4IQpdVsorKw&hd=1>.

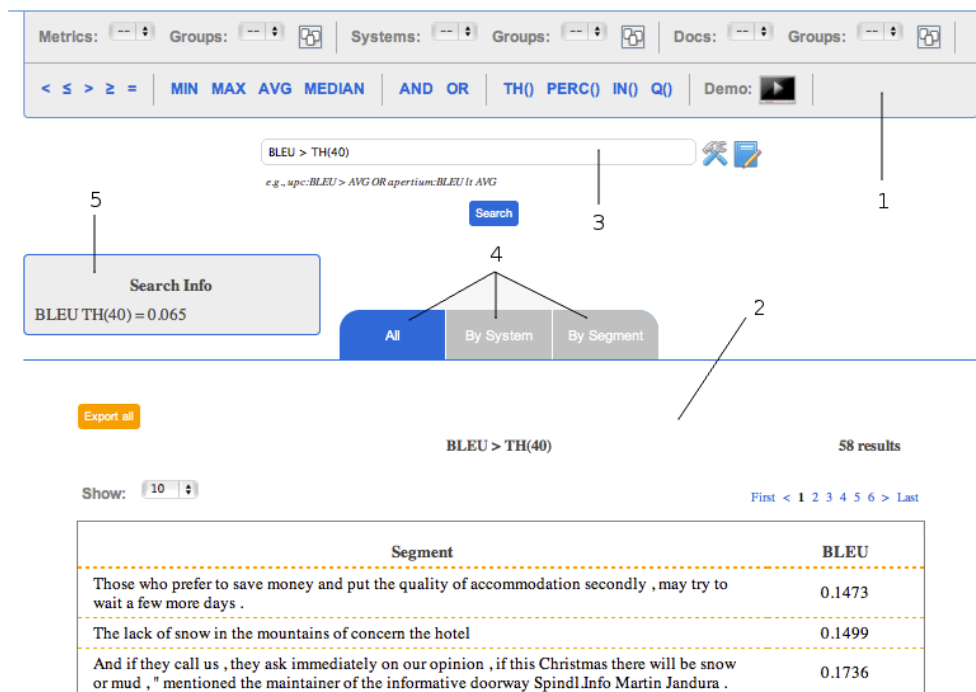


Figure 10: Getting to know *t*SEARCH

Click the following icons to...

Toolbar



Show and hide the toolbar.



Create or edit a group of metrics, systems or documents.



See the video manual.

Other



Export as an XML file the partial results depending on the current view.



Show and hide the examples window.

A.1.2 Views

SEARCH lets you navigate the results of the search across all the automatic translations selected and their evaluations. Three different views organize the segments according to the user preferences:

All: this view shows all segments and the scores for the metrics involved in the query.


By system: it groups the segments by system name and, for each system, by document name.

By segment: this view offers the segment organization, which facilitates the comparison between several translations, the reference and the source for each segment.


A.2 Create and Edit Groups

The interface allows to create groups of systems, documents and/or metrics. The purpose of this feature is to facilitate the comparison between types of systems (e.g., statistical vs. rule-based) or metrics (e.g., lexical vs. syntactic) or even, groups of documents that belong to different domains.

The following steps describe how to create a new group:

1. From the toolbar, click the Groups button . The button is in three different blocks in order to distinguish between metrics, systems and documents.
2. Write the name of your new group at the *Group name* field.
3. Chose from the left panel what do you want to include in your group moving them to the right panel.
4. Click the create button.

Later on, if you want to edit an existing group...

1. From the toolbar, click the Groups button . The button is in three different blocks in order to distinguish between metrics, systems and documents.
2. Select from the *Groups* list the one you want to edit and then, its name and elements are displayed.
3. Edit the values you want to change, i.e., the name of the group or the elements, moving to the left panel the ones you want to eliminate from the group or moving to the right panel the ones you want to include.
4. Click the update button.

A.3 Let's query

There are several types of queries, depending on the operations used: arithmetic comparisons, statistical functions (e.g., average, quartiles), range of values, linguistic elements and logical operators. Table 7 lists some of the most representative queries of each group.

Regarding metric-based queries, the arithmetic comparison queries let you obtain all segments scored above/below a value for a concrete metric. Such value can be a real number or also a statistical variable such as minimum MIN, maximum MAX, median MEDIAN, average AVG or the threshold function TH(). We have also implemented statistical functions such as the quartile function Q() or the percentile PERC(n , M), which returns all the segments with a score in the n^{th} part, when the range of scores is divided in M parts of equal size. The last query in this group refers to the system comparison. Thus, given an evaluation measure, it allows comparing its score among several systems.

Concerning linguistic-based queries, we have implemented queries that match N-grams of lemmas lemma, parts-of-speech pos and items of shallow SP or constituent parsing CP, dependency relations DP, semantic roles SR and named entities NE. The DP function allows specifying a structure composition criterion (i.e., the categories of two words and their dependency relationship) and even a chain of relations. The SR function obtains the segments that match a verb and its list of arguments. The use of the asterisk symbol substitutes any value, e.g., LE[CP(NP, *, PP), DP(*, *, V)]. However, when combined with semantic roles, one asterisk substitutes any verb that has all the arguments specified, e.g., LE[SR(*, AO, A1)], whereas two asterisks in a row allow arguments to belong to different verbs in the same sentence.

The above queries are applied at segment level. However, applying them at system and document-level is as easy as specifying the system and/or document names, e.g., (upc:BLEU > AVG) AND (upc:LE[DP(*, nsubj, *)]). In addition, there is also the possibility to use a group of metrics, systems and/or documents instead, e.g.,

(LEX:RBMT > AVG) AND (RBMT:LE[DP(*, nsubj, *)]), where RBMT is a group of rule-based systems and LEX is a group of lexical metrics defined and created by the user.

Metric-based Queries	Arithmetic Comparison	BLEU > 0.4 BLEU > TH(40) BLEU le MEDIAN
	Range of Values	BLEU IN [0.2, 0.3) BLEU IN Q(4) BLEU IN PERC(2,10) BLEU IN (TH(20),TH(40))
	System comparison	upc:BLEU > dfki:BLEU
LE-based Queries	N-grams	LE[SP(NN,*,VBZ)] LE[CP(NP,PP)] LE[lemma(be),CP(VP,PP)] LE[pos(DT,JJ,*)] LE[NE(ORG)]
	Semantic Roles	LE[SR(ask,A1,AM-TMP)] LE[SR(*,A1,AM-TMP)] LE[SR(**,A1,AM-TMP)]
	Dependency Relationships	LE[DP(N,nsubj,V)] LE[DP(N,nsubj,V,dep,V)] LE[DP(*,nsubj,*)]
Group Creation and Complex Queries	Logical Composition	BLEU > AVG AND LE[DP(N,nsubj,V)] LEX = {BLEU,NIST} SYN = {DP-Or(*),SP-Op(*)} SMT = {bing,google} (SMT:LEX > AVG OR aptium:LEX < AVG) AND (SMT:SYN < AVG OR aptium:SYN > AVG)

Table 7: *t*SEARCH query examples