# iSEARCH: Flexible and Fast Search over Automatic Translations for Improved Quality/Error Analysis

Meritxell Gonzàlez and Laura Mascarell and Lluís Màrquez
TALP Research Center
Universitat Politècnica de Catalunya
{mgonzalez,lmascarell,lluism}@lsi.upc.edu

## Abstract

This work presents iSEARCH, a web-based application that provides mechanisms for doing complex searches over a collection of translation cases evaluated with a large set of diverse measures. iSEARCH uses the evaluation results obtained with the ASIYA toolkit for MT evaluation and it is connected to its on-line GUI, which makes possible a graphical visualization and interactive access to the evaluation results. The search engine offers a flexible query language allowing to find translation examples matching a combination of numerical and structural features associated to the calculation of the quality metrics. Its database design permits a fast response time for all queries supported on realistic-size test beds. In summary, iSEARCH, used with ASIYA, offers developers of MT systems and evaluation metrics a powerful tool for helping translation and error analysis.

## 1 Introduction

In Machine Translation (MT) system development, a qualitative analysis of the translations is a fundamental step in order to spot the limitations of a system, compare the linguistic abilities of different systems or tune the parameters during system refinement. This is especially true in statistical MT systems, where usually no special structured knowledge is used other than parallel data and language models, but also on systems that need to reason over linguistic structures. The need for analyzing and comparing automatic translations with respect to evaluation metrics is also paramount for developers of translation quality metrics, who need elements of analysis to better understand the behavior of their evaluation measures.

This paper presents iSEARCH, a web application that aims to alleviate the burden of manual analysis that developers have to conduct to assess the translation quality aspects involved in the above mentioned situations. As a toy example, consider for instance an evaluation setting with two 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 $m_2$ 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 $m_2$. iSEARCH allows to retrieve (visualize and export) these sentences with a simple query in a fast time response. The search can 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 examples.

iSEARCH is build on top of ASIYA (Giménez and Márquez, 2010), an open-source toolkit for MT evaluation; and it can be used along with the ASIYA ON-LINE INTERFACE (González et al., 2012), which provides an interactive environment to examine the sentences. ASIYA allows to analyze a wide range of linguistic aspects of candidate and reference translations using a large set of automatic and heterogeneous evaluation metrics. In particular, it offers a especially rich set of measures that use syntactic and semantic information. The intermediate structures generated by the parsers, and used to compute the scoring measures, could be priceless for MT developers, who can use them to compare the structures of several translations and see how they affect the performance of the metrics, providing more understanding in order to interpret the actual performance of the automatic translation systems.

iSEARCH consists of: 1) a database that stores...
the resources generated by ASIYA, 2) 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, and allows to export these results in XML format. The application is publicly accessible on-line\footnote{http://asiya.lsi.upc.edu/demo}, and a brief explanation of its most important features is given in the demonstrative video.

In the following, Section 2 gives an overview of the ASIYA toolkit and the information gathered from the evaluation output. Section 3 and Section 4 describe in depth the tSEARCH application and the on-line interface, respectively. Finally, Section 5 reviews similar applications in comparison to the functionalities addressed by tSEARCH.

## 2 MT Evaluation with the ASIYA Toolkit

Currently, ASIYA contains more than 800 variants of MT metrics to measure the similarity between two translations at several linguistic dimensions. Moreover, the scores can be calculated at three granularity levels: system (entire test-set), document and sentence (or segment).

As shown in Figure 1, ASIYA requires the user to provide a test suite. Then, the input files are processed in order to calculate the annotations, the parsing trees and the final metric scores. Several external components are used for both, metric computation and automatic linguistic analysis\footnote{A complete list of external components can be found in the Technical Manual at the ASIYA web-site}.

The use of these tools depends on the languages supported and the type of measures one needs to obtain. Hence, for instance, lexical-based measures are computed using the last version of most popular metrics, such as BLEU, NIST, METEOR or ROUGE. The syntax-wise measures need the output of taggers, lemmatizers, parsers and other analyzers. In those cases, ASIYA uses the SVMTool (Jiménez and Márquez, 2004), BIOS (Surdeanu et al., 2005), the Charniak-Johnson and Berkeley constituent parsers (Charniak and Johnson, 2005; Petrov and Klein, 2007), and the MALT dependency parser (Nivre et al., 2007), among others.

In the tSEARCH platform, the system manages the communication with an instance of the ASIYA toolkit running on the server. For every test suite, the system maintains a synchronized representation of the input data, the evaluation results and the linguistic information generated. Then, the system updates a database where the test suites are stored for further analysis using the tSEARCH tool, as described next.

## 3 The tSEARCH Tool

tSEARCH offers a graphical search engine to analyze a given test suite. The system core retrieves all translation examples that satisfy certain properties related to either the evaluation scores or the linguistic structures. The query language designed is simple and flexible, and it allows to combine many properties to build sophisticated searches.

The tSEARCH architecture consists of the three components illustrated in Figure 2: the web-based interface, the storage system based on NoSQL technology and the tSEARCH core, composed of a query parser and a search engine.

The databases (Section 3.1) are fed through the “Data Loader API” used by ASIYA. At run-time, during the calculation of the measures, ASIYA inserts all the information being calculated (metrics and parses) and a number of precalculated variables (e.g., average, mean and percentiles). These operations are made in parallel, which makes the overhead of filling the database marginal.

The query parser (Section 3.2) receives the query from the on-line interface and converts it into a binary tree structure where each leaf is a sin-
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 database, 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 iSEARCH query language 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 iSEARCH data model has the three CFs shown in Figure 3. The scores CF in Figure 3(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 3(b) stores basic statistics, such as the minimum, maximum, average, median and percentiles values for every evaluation metric. The CF having the linguistic elements in Figure 3(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 iSEARCH 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.

3.2 The Query Language and Parser

The Query Parser module is one of the key ingredients in the iSEARCH application because it determines the query grammar and the allowed operations, and it provides a parsing method to analyze any query and produce a machine-readable version of its 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 described in the following subsections (3.2.1 to 3.2.3) and listed in Figure 4.

3.2.1 Segment-level and Metric-based Queries

The most basic queries are those related to segment level scores, i.e., obtain all segments scored above/below a value for a concrete metric. The common comparison operators are supported, such as for instance, BLEU > 0.4 and BLEU gt 0.4, that are both correct and equivalent queries.

Basic statistics are also calculated at run-time, which allows to use statistic variables as values, e.g., obtain the segments scored in the fourth quartile of BLEU. The maximum, minimum, average, median and percentile values of each metric are precalculated and saved into the MAX, MIN, AVG, MEDIAN and PERC variables, respectively. The thresholds and quartiles (TH, Q) are calculated at run-time based on percentiles. MIN and MAX can also be used and allow to get all segments in the test set (i.e., BLEU ge MIN).

The threshold function implies a percentage. The query BLEU > TH(20) gets all segments that have a BLEU score greater than the score value of the bottom 20% of the sentences.

It is also possible to specify an interval of values using the operator IN[x, y]. The use of parenthesis is allowed in order to exclude the boundaries. The arguments for this operator can be either numerical values or the predefined functions for quartiles and percentiles. Therefore, the following example BLEU IN [TH(20), TH(30)] returns all segments with a BLEU score in the range between the threshold of the 20% (included) and the 30% (excluded).

The quartile function Q(X) takes a value between 1 and 4 and returns all segments that have their score in that quartile. In contrast, the percentile function generalizes the previous: PERC(n,M), where 1 < M <= 100; 1 <= n <= M, returns all the segments with a score in the n

3.2.2 System- and Document-level Queries

The queries described next implement the search procedures for more sophisticated queries involving system and document level properties, and also the linguistic information used in the calculation of the evaluation measures. The purpose of
this functionality is to answer questions related to groups of systems and/or metrics.

As explained in the introduction, one may want to find the segments with good scores for lexical metrics and, simultaneously, bad scores for syntactic-based ones, or vice versa. The following query illustrates how to do it: 

\((s_{rb}[\text{LEX}] > \text{AVG}) \lor (s_{rb}[\text{LEX}] < \text{AVG})\) AND \((s_{rb}[\text{SYN}] < \text{AVG}) \lor (s_{rb}[\text{SYN}] > \text{AVG})\)

where \(s_{rb}\) = \(\{s_1, s_2\}\) is the definition of a group of the rule-based systems \(s_1\) and \(s_2\), \(s_3\) is another translation system, and \(\text{LEX}=\{\text{BLEU, NIST}\}\) and \(\text{SYN}=\{\text{CP-Op(*), SP-Oc(*)}\}\) are two groups of lexical- and syntactic-based measures, respectively. The output of this kind of queries can help developers to inspect the differences between the systems that meet these criteria.

Concerning queries at document level, its structure is the same but applied at document scope. They may help to find divergences when translating documents from different domains.

### 3.2.3 Linguistic Element-based Queries

The last functionality in \textit{tSEARCH} allows searching the segments that contain specific linguistic elements (LE), estimated with any of the analyzers used to calculate the linguistic structures. Linguistic-wise queries will allow the user to find segments which match the criteria for any linguistic feature calculated by ASIYA: part-of-speech, lemmas, named entities, grammatical categories, dependency relations, semantic roles and discourse structures.

We have implemented queries that match n-grams of lemmas (\texttt{lemma}), parts-of-speech (\texttt{pos}) and items of shallow (\texttt{SP}) or constituent parsing (\texttt{CP}), dependency relations (\texttt{DP}) and semantic roles \texttt{SR}, such as \texttt{LE[lemma(be), pos(NN,adj), SP(NP, ADJP, VP), CP(VP, PP)]}. The \texttt{DP} function allows also specifying a compositional criterion (i.e., the categories of two words and their dependency relationship) and even a chain of relations, e.g., \texttt{LE[DP(N, nsubj j, V, dep, V)]}. In turn, the \texttt{SR} function obtains the segments that match a verb and its list of arguments, e.g., \texttt{LE[SR(ask, A0, A1)]}.

The asterisk symbol can be used to substitute any \texttt{LE}-item, e.g., \texttt{LE[SP(NP, *, PP), DP(*, *, V)]}. When combined with semantic roles, one asterisk substitutes any verb that has all the arguments specified, e.g., \texttt{LE[SR(*, A0, A1)]}, whereas two asterisks in a row allow arguments to belong to different verbs in the same sentence. For instance, \texttt{LE[SR(**, A1, AM–TMP)]} matches the sentence \textit{Those who prefer to save money, may try to wait a few more days}, where the verb \texttt{wait} has the argument \texttt{AM–TMP} and the verb \texttt{prefer} has the argument \texttt{A1}.

### 4 On-line Interface and Export of the Results

\textit{tSEARCH} is fully accessible on-line through the ASIYA ON-LINE INTERFACE. The web application runs ASIYA remotely, calculates the scores and fills the \textit{tSEARCH} 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 \textit{tSEARCH} interface is already accessible, one can see a tools icon on the right of the search box. It shows the toolbar with all available metrics, functions and operations. The search box allows to query the database using the query language described in Section 3.2.

After typing a query, the user can navigate the results using three different views that organize them according to the user preferences: 1) \textit{All segments} shows all segments and metrics mentioned in the query, the segments can be sorted by the score, in ascendent or descendent order, just tapping on the metric name; 2) \textit{Grouped by system} groups the segments by system and, for each system, by document; 3) \textit{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 5. 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.

5 Related Work and Conclusions

The ultimate goal of rSEARCH is to provide the community with a user-friendly tool that facilitates the qualitative analysis of automatic translations. Currently, there are no freely available automatic tools for aiding MT evaluation tasks. For this reason, we believe that rSEARCH can be a useful tool for MT system and evaluation metric developers.

So far, related works in the field address (semi-)automatic error analysis from different perspectives. A framework for error analysis and classification was proposed in (Vilar et al., 2006), which has inspired more recent works in the area, such as (Fishel et al., 2011). They propose a method for automatic identification of various error types. The methodology proposed is language independent and tackles lexical information. Nonetheless, it can also take into account language-dependent information if linguistic analyzers are available. The user interface presented in (Berka et al., 2012) provides also automatic error detection and classification. It is the result of merging the Hjerson tool (Popovic, 2011) and Addicter (Zeman et al., 2011). This web application shows alignments and different types of errors colored.

In contrast, the ASIYA interface and the rSEARCH tool together facilitate the qualitative analysis of the evaluation results yet providing a framework to obtain multiple evaluation metrics and linguistic analysis of the translations. They also provide the mechanisms to search and find relevant translation examples using a flexible query language, and to export the results.

Acknowledgments

This research has been partially funded by the Spanish Ministry of Education and Science (OpenMT-2, TIN2009-14675-C03), the European Community’s Seventh Framework Programme under grant agreement number 247762 (FAUST, FP7-ICT-2009-4-247762) and the EAMT Sponsorship of Activities: Small research and development project, 2012.

References

Jan Berka, Ondřej Bojar, Mark Fishel, Maja Popovic, and Daniel Zeman. 2012. Automatic MT error analysis: Hjerson helping Addicter. In Proc. 8th LREC.


