BLAST: A Tool for Error Analysis of Machine Translation Output

Sara Stymne
Department of Computer and Information Science
Linköping University, Linköping, Sweden
sara.stymne@liu.se

Abstract

We present BLAST, an open source tool for error analysis of machine translation (MT) output. We believe that error analysis, i.e., to identify and classify MT errors, should be an integral part of MT development, since it gives a qualitative view, which is not obtained by standard evaluation methods. BLAST can aid MT researchers and users in this process, by providing an easy-to-use graphical user interface. It is designed to be flexible, and can be used with any MT system, language pair, and error typology. The annotation task can be aided by highlighting similarities with a reference translation.

1 Introduction

Machine translation evaluation is a difficult task, since there is not only one correct translation of a sentence, but many equally good translation options. Often, machine translation (MT) systems are only evaluated quantitatively, e.g. by the use of automatic metrics, which is fast and cheap, but does not give any indication of the specific problems of a MT system. Thus, we advocate human error analysis of MT output, where humans identify and classify the problems in machine translated sentences.

In this paper we present BLAST,[1] a graphical tool for performing human error analysis, from any MT system and for any language pair. BLAST has a graphical user interface, and is designed to be easy and intuitive to work with. It can aid the user by highlighting similarities with a reference sentence. BLAST is flexible in that it can be used with output from any MT system, and with any hierarchical error typology. It has a modular design, allowing easy extension with new modules. To the best of our knowledge, there is no other publicly available tool for MT error annotation. Since we believe that error analysis is a vital complement to MT evaluation, we think that BLAST can be useful for many other MT researchers and developers.

[1]The BiLingual Annotation/Annotator/Analysis Support Tool, available for download at http://www.ida.liu.se/~sarst/blast/

2 MT Evaluation and Error Analysis

Hovy et al. (2002) discussed the complexity of MT evaluation, and stressed the importance of adjusting evaluation to the purpose and context of the translation. However, MT is very often only evaluated quantitatively using a single metric, especially in research papers. Quantitative evaluations can be automatic, using metrics such as Bleu (Papineni et al., 2002) or Meteor (Denkowski and Lavie, 2010), where the MT output is compared to one or more human reference translations. Metrics, however, only give a single quantitative score, and do not give any information about the strengths and weaknesses of the system. Comparing scores from different metrics can give a very rough indication of some major problems, especially in combination with a part-of-speech analysis (Popović et al., 2006).

Human evaluation is also often quantitative, for instance in the form of estimates of values such as adequacy and fluency, or by ranking sentences from different systems (e.g. Callison-Burch et al. (2007)). A combination of human and automatic metrics is

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 56–61,
Portland, Oregon, USA, 21 June 2011.

human-targeted metrics such as HTER, where a human corrects the output of a system to the closest correct translation, on which standard metrics such as TER is then computed (Snover et al., 2006). While these types of evaluation are certainly useful, they are expensive and time-consuming, and still do not tell us anything about the particular errors of a system.[2]

Thus, we think that qualitative evaluation is an important complement, and that error analysis, the identification and classification of MT errors, is an important task. There have been several suggestions for general MT error typologies (Flanagan, 1994; Vilar et al., 2006; Farrús et al., 2010), targeted at different user groups and purposes, focused on either evaluation of single systems, or comparison between systems. It is also possible to focus error analysis at a specific problem, such as verb form errors (Murata et al., 2005).

We have not been able to find any other freely available tool for error analysis of MT. Vilar et al. (2006) mentioned in a footnote that "a tool for highlighting the differences [between the MT system and a correct translation] also proved to be quite useful" for error analysis. They do not describe this tool any further, and do not discuss if it was also used to mark and store the error annotations themselves.

Some tools for post-editing of MT output, a related activity to error analysis, have been described in the literature. Font Llitjós and Carbonell (2004) presented an online tool for eliciting information from the user when post-editing sentences, in order to improve a rule-based translation system. The post-edit operations were labeled with error categories, making it a type of error analysis. This tool was highly connected to their translation system, and it required users to post-edit sentences by modifying word alignments, something that many users found difficult. Glenn et al. (2008) described a post-editing tool used for HTER calculation, which has been used in large evaluation campaigns. The tool is a pure post-editing tool and the edits are not classified. Graphical tools have also successfully been used to aid humans in other MT-related tasks, such as human MT evaluation of adequacy, fluency and system comparison (Callison-Burch et al., 2007), and word alignment (Ahrenberg et al., 2003).

[2]Though it does, at least in principle, seem possible to mine HTER annotations for more information

3 System Overview

BLAST is a tool for human annotations of bilingual material. Its main purpose is error analysis for machine translation. BLAST is designed for use in any MT evaluation project. It is not tied to the information provided by specific MT systems, or to specific languages, and it can be used with any hierarchical error typology. It has a preprocessing module for automatically aiding the annotator by highlighting similarities between the MT output and a reference. Its modular design allows easy integration of new modules for preprocessing. BLAST has three working modes for handling error annotations: for adding new annotations, for editing existing annotations, and for searching among annotations.

BLAST can handle two types of annotations: error annotations and support annotations. Error annotations are based on a hierarchical error typology, and are used to annotate errors in MT output. Error annotations are added by the users of BLAST. Support annotations are used as a support to the user, currently to mark similarities in the system and reference sentences. The support annotations are normally created automatically by BLAST, but they can also be modified by the user. Both annotation types are stored with the indices of the words they apply to.

Figure 1 shows a screenshot of BLAST. The MT output is shown to the annotator one segment at a time, in the upper part of the screen. A segment normally consists of a sentence and the MT output can be accompanied by a source sentence, a reference sentence, or both. Error annotations are marked in the segments by bold, underlined, colored text, and support annotations are marked by light background colors. The bottom part of the tool, contains the error typology, and controls for updating annotations and navigation. The error typology is shown using a menu structure, where submenus are activated by the user clicking on higher levels.

3.1 Design goals

We created BLAST with the goal that it should be flexible, and allow maximum freedom for the user,

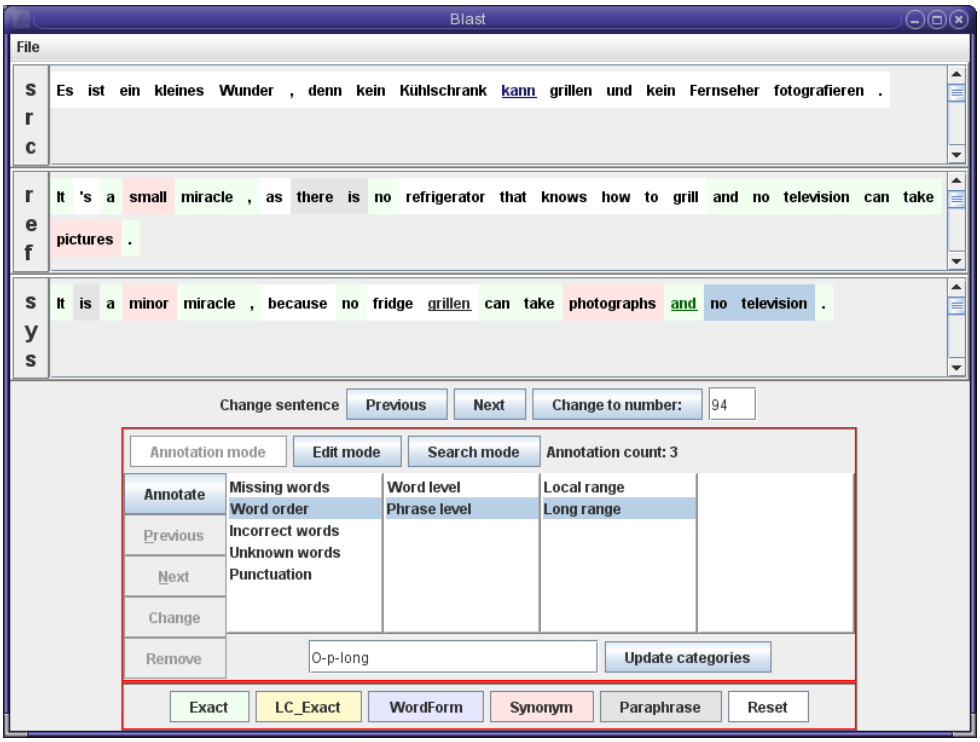

Figure 1: Screenshot of BLAST

based on the following goals:

- Independent of the MT system being analyzed, particularly not dependent on specific information given by a particular MT system, such as alignment information
- Compatible with any error typology
- Language pair independent
- Possible to mark where in a sentence an error occurs
- Possible to view either source or reference sentences, or both
- Possible to automatically highlight similarities between the system and the reference sentences
- Containing a search function for errors
- Simple to understand and use

The current implementation of BLAST fulfils all these goals, with the possible small limitation that the error typology has to be hierarchical. We believe this limitation is minor, however, since it is possible to have a relatively flat structure if desired, and to re-use the same submenu in many places, allowing cross-classification within a hierarchical typology.

The flexibility of the tool gives users a lot of freedom in how to use it in their evaluation projects. However, we believe that it is important within every error annotation project to use a set error typology and guidelines for annotation, but the annotation tool should not limit users in making these choices.

3.2 Error Typologies

As described above, BLAST is easily configurable with new typologies for annotation, with the only restriction that the typology is hierarchical. BLAST currently comes with the following implemented typologies, some of which are general, and some of which are targeted at specific language (pairs):

- Vilar et al. (2006)
  - General
  - Chinese
  - Spanish
- Farrús et al. (2010)
  - Catalan–Spanish
- Flanagan (1994) (slightly modified into a hierarchical structure)
  - French

  - German
- Our own tentative fine-grained typology
  - General
  - Swedish

The error typologies can be very big, and it is hard to fit an arbitrarily large typology into a graphical tool. BLAST thus uses a menu structure which always shows the categories in the first level of the typology. Lower subtypologies are only shown when they are activated by the user clicking on a higher level. In Figure 1, the subtypologies to *Word order* were activated by the user first clicking on *Word order*, then on *Phrase level*.

It is important that typologies are easy to extend and modify, especially in order to cover new target languages, since the translation problems to some extent will be dependent on the target language, for instance with regard to the different agreement phenomena in languages. The typologies that come with BLAST can serve as a starting point for adjusting typologies, especially to new target languages.

3.3 Implementation

BLAST is implemented as a Java application using Swing for the graphical user interface. Using Java makes it platform independent, and it is currently tested on Unix, Linux, Mac, and Windows. BLAST has an object-oriented design, with a particular focus on modular design, to allow it to be easily extendible with new modules for preprocessing, reading and writing to different file formats, and presenting statistics. Unicode is used in order to allow a high number of languages, and sentences can be displayed both right to left, and left to right. BLAST is open source and is released under the LGPL license.[3]

3.4 File formats

The main file types used in BLAST is the annotation file, containing the translation segments and annotations, and the typology file. These files are stored in a simple text file format. There is also a configuration file, which can be used for program settings, besides using command line options, for instance to configure color schemes, and to change preprocessing settings. The statistics of an annotation project are printed in a text file in a human-readable format (see Section 4.5).

The annotation file contains the translation segments for the MT system, and possibly for the source and reference sentences, and all error and support annotations. The annotations are stored with the indices of the word(s) in the segments that were marked, and a label identifying the error type. The annotation file is initially created automatically by BLAST based on sentence aligned files. It is then updated by BLAST with the annotations added by the user.

The typology file has a header with main information, and then an item for each menu containing:

- The name of the menu
- A list of menu items, containing:
  - Display name
  - Internal name (used in annotation file, and internally in BLAST)
  - The name of its submenu (if any)

The typology files have to be specified by the user, but BLAST comes with several typology files, as described in Section 3.2.

4 Working with BLAST

BLAST has three different working modes: annotation, edit and search. The main mode is annotation, which allows the user to add new error annotations. The edit mode allows the user to edit and remove error annotations. The search mode allows the user to search for errors of different types. BLAST can also create support annotations, that can later be updated by the user, and calculate and print statistics of an annotation project.

4.1 Annotation

The annotation mode is the main working mode in BLAST, and it is active in Figure 1. In annotation mode a segment is shown with all its current error annotations. The annotations are marked with bold and colored text, where the color depends on the main type of the error. For each new annotation the user selects the word or words that are wrong, and selects an error type. In figure 1, the words *no television*, and the error type *Word order→Phrase level→Long* are selected in order to add a new error

[3] http://www.gnu.org/copyleft/lesser.html

annotation. BLAST ignores identical annotations, and warns the user if they try to add an annotation for the exact same words as another annotation.

4.2 Edit

In edit mode the user can change existing error annotations. In this mode only one annotation at a time is shown, and the user can switch between them. For each annotation affected words are highlighted, and the error typology area shows the type of the error. The currently shown error can be changed to a different error type, or it can be removed. The edit mode is useful for revising annotations, and for correcting annotation errors.

4.3 Search

In search mode, it is possible to search for errors of a certain type. To search, users choose the error type they want to search for in the error typology, and then search backwards or forwards for error annotations of that type. It is possible both to search for specific errors deep in the typology, and to search for all errors of a type higher in the typology, for instance, to search for all word order errors, regardless of subclassification. Search is active between all segments, not only for the currently shown segment. Search is useful for controlling the consistency of annotations, and for finding instances of specific errors.

4.4 Support annotations

Error annotation is a hard task for humans, and thus we try to aid it by including automatic preprocessing, where similarities between the system and reference sentences are marked at different levels of similarity. Even if the goal of the error analysis often is not to compare the MT output to a single reference, but to the closest correct equivalent, it can still be useful to be able to see the similarities to one reference sentence, to be able to identify problematic parts easier.

For this module we have adapted the code for alignment used in the Meteor-NEXT metric (Denkowski and Lavie, 2010) to BLAST. In Meteor-NEXT the system and reference sentences are aligned at the levels of exact matching, stemmed matching, synonyms, and paraphrases. All these modules work on lower-cased data, so we added a module for exact matching with the original casing kept. The exact and lower-cased matching works for most languages, and stemming for 15 languages. The synonym module uses WordNet, and is only available for English. The paraphrase module is based on an automatic paraphrase induction method (Bannard and Callison-Burch, 2005), it is currently trained for five languages, but the Meteor-NEXT code for training it for additional languages is included.

Support annotations are normally only created automatically, but BLAST allows the user to edit them. The mechanism for adding, removing or changing support annotations is separate from error annotations, and can be used regardless of mode.

4.5 Create Statistics

The statistics module prints statistics about the currently loaded annotation project. The statistics are printed to a file, in a human-readable format. It contains information about the number of sentences and errors in the project, average number of errors per sentence, and how many sentences there are with certain numbers of errors. The main part of the statistics is the number and percentage of errors for each node in the error typology. It is also possible to get the number of errors for cross-classifications, by specifying regular expressions for the categories to cross-classify in the configuration file.

5 Future Extensions

BLAST is under active development, and we plan to add new features. Most importantly we want to add the possibility to annotate two MT systems in parallel, which can be useful if the purpose of the annotation is to compare MT systems. We are also working on refining and developing the existing proposals for error typologies, which is an important complement to the tool itself. We intend to define a new fine-grained general error typology, with extensions to a number of target languages.

The modularity of BLAST also makes it possible to add new modules, for instance for preprocessing and to support other file formats. One example would be to support error annotation of only specific phenomena, such as verb errors, by adding a preprocessing module for highlighting verbs with support

annotations, and a suitable verb-focused error typology. We are also working on a preprocessing module based on grammar checker techniques (Stymne and Ahrenberg, 2010), that highlights parts of the MT output that it suspects are non-grammatical.

Even though the main purpose of BLAST is for error annotation of machine translation output, the freedom in the use of error typologies and support annotations also makes it suitable for other tasks where bilingual material is used, such as for annotations of named entities in bilingual texts, or for analyzing human translations, e.g. giving feedback to second language learners, with only the addition of a suitable typology, and possibly a preprocessing module.

6 Conclusion

We presented BLAST; a flexible tool for annotation of bilingual segments, specifically intended for error analysis of MT. BLAST facilitates the error analysis task, which we believe is vital for MT researchers, and could also be useful for other users of MT. Its flexibility makes it possible to annotate translations from any MT system and between any language pairs, using any hierarchical error typology.

References

Lars Ahrenberg, Magnus Merkel, and Michael Petterstedt. 2003. Interactive word alignment for language engineering. In *Proceedings of EACL*, pages 49–52, Budapest, Hungary.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*, pages 597–604, Ann Arbor, Michigan, USA.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Proceedings of WMT*, pages 136–158, Prague, Czech Republic, June.

Michael Denkowski and Alon Lavie. 2010. METEOR-NEXT and the METEOR paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of WMT and MetricsMATR*, pages 339–342, Uppsala, Sweden.

Mireia Farrús, Marta R. Costa-jussà, José B. Mariño, and José A. R. Fonollosa. 2010. Linguistic-based evaluation criteria to identify statistical machine translation errors. In *Proceedings of EAMT*, pages 52–57, Saint Raphaël, France.

Mary Flanagan. 1994. Error classification for MT evaluation. In *Proceedings of AMTA*, pages 65–72, Columbia, Maryland, USA.

Ariadna Font Llitjós and Jaime Carbonell. 2004. The translation correction tool: English-Spanish user studies. In *Proceedings of LREC*, pages 347–350, Lisbon, Portugal.

Meghan Lammie Glenn, Stephanie Strassel, Lauren Friedman, and Haejoong Lee. 2008. Management of large annotation projects involving multiple human judges: a case study of GALE machine translation post-editing. In *Proceedings of LREC*, pages 2957–2960, Marrakech, Morocco.

Eduard Hovy, Margaret King, and Andrei Popescu-Belis. 2002. Principles of context-based machine translation evaluation. *Machine Translation*, 17(1):43–75.

Masaki Murata, Kiyotaka Uchimoto, Qing Ma, Toshiyuki Kanamaru, and Hitoshi Isahara. 2005. Analysis of machine translation systems' errors in tense, aspect, and modality. In *Proceedings of PACLIC 19*, pages 155–166, Taipei, Taiwan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Maja Popović, Adrià de Gisper, Deepa Gupta, Patrik Lambert, Hermann Ney, José Mariño, and Rafael Banchs. 2006. Morpho-syntactic information for automatic error analysis of statistical machine translation output. In *Proceedings of WMT*, pages 1–6, New York City, New York, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human notation. In *Proceedings of AMTA*, pages 223–231, Cambridge, Massachusetts, USA.

Sara Stymne and Lars Ahrenberg. 2010. Using a grammar checker for evaluation and postprocessing of statistical machine translation. In *Proceedings of LREC*, pages 2175–2181, Valetta, Malta.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. 2006. Error analysis of machine translation output. In *Proceedings of LREC*, pages 697–702, Genoa, Italy.