A Advanced Features

Here we discuss advanced features that allow for more sophisticated types of analysis using other sources of information than the references and system outputs themselves.

Label-wise Abstraction One feature that greatly improves the flexibility of analysis is `compare-mt`'s ability to do analysis over arbitrary word labels. For example, we can perform word accuracy analysis where we bucket the words by POS tags, as shown in 4. In the case of the PBMT vs. NMT analysis above, this uncovers the interesting fact that PBMT was better at generating base-form verbs, whereas NMT was better at generating conjugated verbs. This can also be applied to the n-gram analysis, finding which POS n-grams are generated well by one system or another, a type of analysis that was performed by Chiang et al. (2005) to understand differences in reordering between different systems.

Labels are provided by external files, where there is one label per word in the reference and system outputs, which means that generating these labels can be an arbitrary pre-processing step performed by the user without any direct modifications to the `compare-mt` code itself. These labels do not have to be POS tags, of course, and can also be used for other kinds of analysis. For example, one may perform analysis to find accuracy of generation of words with particular morphological tags (Popović et al., 2006), or words that appear in a sentiment lexicon (Mohammad et al., 2016).

Source-side Analysis While most analysis up until this point focused on whether a particular word on the *target* side is accurate or not, it is also of interest what source-side words are or are not accurately translated. `compare-mt` also supports word accuracy analysis for source-language words given the source language input file, and alignments between the input, and both the reference and the system outputs. Using alignments, `compare-mt` finds what words on the source side were generated correctly or incorrectly on the target side, and can do aggregate word accuracy analysis, either using word frequency or labels such as POS tags.

Word Likelihood Analysis Finally, as many recent methods can directly calculate a log likelihood for each word, we also provide another tool

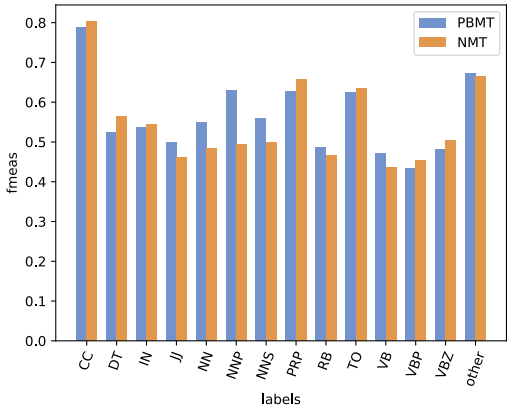

Figure 4: Word F-measure bucketed by POS tag.

`compare-ll` that makes it possible to perform holistic analysis of these log likelihoods. First, the user creates a file where there is one log likelihood for each word in the reference file, and then, like the word accuracy analysis above, `compare-ll` can calculate aggregate statistics for this log likelihood based on word buckets.

Extending `compare-mt` One other useful feature is `compare-mt`'s ability to be easily extended to new types of analysis. For example,

- If a user is interested in using a different evaluation metric, they could implement a new instance of the `Scorer` class and use it for both aggregate score analysis (with significance tests), sentence bucket analysis, or sentence example analysis.
- If a user wanted to bucket words according to a different type of statistic or feature, they could implement their own instance of a `Bucketer` class, and use this in the word accuracy analysis.

B Related Research and Tools

There have been a wide variety of tools and methods developed to perform analysis of machine translation results. These can be broadly split into those that attempt to perform *holistic analysis* and those that attempt to perform *example-by-example* anaylsis.

`compare-mt` is a tool for holistic analysis over the entire corpus, and many of the individual pieces of functionality provided by `compare-mt` are inspired by previous works on this topic. Our word error rate analysis is inspired by previous work on *automatic error analysis*, which takes a

typology of errors (Flanagan, 1994; Murata et al., 2005; Vilar et al., 2006), and attempts to automatically predict which sentences contain these errors (Popović and Ney, 2011; Zeman et al., 2011; Fishel et al., 2012). Many of the ideas contained in these works can be used easily in `compare-mt`. Measuring word matches, insertions, and deletions decomposed over POS/morphological tags (Popović et al., 2006; Popović and Ney, 2007; Zeman et al., 2011; El Kholy and Habash, 2011) or other "linguistic checkpoints" (Zhou et al., 2008; Naskar et al., 2011) can be largely implemented using the labeled bucketing functionality described in §A. Analysis of word reordering accuracy (Birch et al., 2010; Popović and Ney, 2011; Bentivogli et al., 2016) can be done through the use of reordering-sensitive measures such as RIBES as described in §2. In addition, the extraction of salient n-grams is inspired by similar approaches for POS n-gram (Chiang et al., 2005; Lopez and Resnik, 2005) and word n-gram (Akabe et al., 2014) based analysis respectively. To the best of our knowledge, and somewhat surprisingly, no previous analysis tool has included the flexible sentence-bucketed analysis that is provided by `compare-mt`.

One other practical advantage of `compare-mt` compared to other tools is that it is publicly available under the BSD license on GitHub,[3] and written in modern Python, which is quickly becoming the standard program language of the research community. Many other tools are either no longer available (Stymne, 2011), or written in other languages such as Perl (Zeman et al., 2011) or Java (Naskar et al., 2011), which provides some degree of practical barrier to their use and extension.

In contrast to the holistic methods listed above, example-by-example analysis methods attempt to intelligently visualize single translation outputs in a way that can highlight salient differences between the outputs of multiple systems, or understand the inner workings of a system. There are a plethora of tools that attempt to make the manual analysis of individual outputs of multiple systems, through visualization or highlighting of salient parts of the output (Lopez and Resnik, 2005; Stymne, 2011; Zeman et al., 2011; Madnani, 2011; Aziz et al., 2012; Gonzàlez et al., 2012; Federmann, 2012; Chatzitheodorou and Chatzistamatis, 2013; Klejch et al., 2015). There has also been work that attempts to analyze the internal representations or alignments of phrase-based (DeNeefe et al., 2005; Weese and Callison-Burch, 2010) and neural (Ding et al., 2017; Lee et al., 2017) machine translation systems to attempt to understand why they arrived at the decisions they did. While these tools are informative, they play a complementary role to the *holistic analysis* that `compare-mt` proposes, and adding the ability to more visually examine individual examples to `compare-mt` in a more extensive manner is planned as future work.

Recently, there has been a move towards creating special-purpose diagnostic test sets designed specifically to test an MT system's ability to handle a particular phenomenon. For example, these cover things like grammatical agreement (Sennrich, 2017), translation of pronouns or other discourse-sensitive phenomena (Müller et al., 2018; Bawden et al., 2018), or diagnostic tests for a variety of different phenomena (Isabelle et al., 2017). These sets are particularly good for evaluating long-tail phenomena that may not appear in naturalistic data, but are often limited to specific language pairs and domains. `compare-mt` takes a different approach of analyzing the results on existing test sets and attempting to extract salient phenomena, an approach that affords more flexibilty but less focus than these special-purpose methods.

C Example Command

Fig. 5 shows an example of the command that was used to generate the report containing the figures and tables used in this paper.

[3] https://github.com/neulab/compare-mt

```
compare-mt
  example/ted.ref.eng example/ted.sys1.eng example/ted.sys2.eng
  --compare_scores
    score_type=bleu,bootstrap=1000
    score_type=ribes,bootstrap=1000
    score_type=length,bootstrap=1000
  --compare_word_accuracies
    bucket_type=freq,freq_corpus_file=example/ted.train.eng
    bucket_type=label,ref_labels=example/ted.ref.eng.tag,out_labels=\
     "example/ted.sys1.eng.tag;example/ted.sys2.eng.tag",\
     label_set=CC+DT+IN+JJ+NN+NNP+NNS+PRP+RB+TO+VB+VBP+VBZ
  --output_directory outputs
  --sys_names PBMT NMT
```

Figure 5: The command used to generate the figures and tables in this paper.