MAISE: A Flexible, Configurable, Extensible Open Source Package for Mass AI System Evaluation

Omar F. Zaidan
Dept. of Computer Science
and
The Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218, USA
ozaidan@cs.jhu.edu

Abstract

The past few years have seen an increasing interest in using Amazon's Mechanical Turk for purposes of collecting data and performing annotation tasks. One such task is the mass *evaluation* of system output in a variety of tasks. In this paper, we present MAISE, a package that allows researchers to evaluate the output of their AI system(s) using human judgments collected via Amazon's Mechanical Turk, greatly streamlining the process. MAISE is open source, easy to run, and platform-independent. The core of MAISE's codebase was used for the manual evaluation of WMT10, and the completed package is being used again in the current evaluation for WMT11. In this paper, we describe the main features, functionality, and usage of MAISE, which is now available for download and use.

1 Introduction

The ability to evaluate system output is one of the most important aspects of system development. A properly designed evaluation paradigm could help researchers test and illustrate the effectiveness, or lack thereof, of any changes made to their system. The use of an automatic metric, whether it is a simple one such as classification accuracy, or a more task-specific metric such as BLEU and TER for machine translation, has become a standard part of any evaluation of empricial methods. There is also extensive interest in exploring *manual* evaluation of system outputs, and in making such a process feasible and efficient, time- and cost-wise. Such human feedback would also be valuable because it would help identify systematic errors and guide future system development.

Amazon's Mechanical Turk (MTurk) is a virtual marketplace that allows anyone to create and post tasks to be completed by human workers around the globe. Each instance of those tasks, called a Human Intelligence Task (HIT) in MTurk lingo, typically requires human understanding and perception that machines are yet to achieve, hence making MTurk an example of "artificial artificial intelligence," as the developers of MTurk aptly put it. Arguably, the most attractive feature of MTurk is the low cost associated with completing HITs and the speed at which they are completed.

Having discovered this venue, many researchers in the fields of artificial intelligence and machine learning see MTurk as a valuable and effective source of annotations, labels, and data, namely the kind requiring human knowledge.

One such kind of data is indeed human evaluation of system outputs. For instance, if you construct several speech recognition systems, and would like to know how well each of the systems performs, you could create HITs on MTurk that 'showcase' the transcriptions obtained by the different systems, and ask annotators to indicate which systems are superior and which ones are inferior. The same can be applied to a variety of tasks, such as machine translation, object recognition, emotion detection, etc.

The aim of the MAISE package is to streamline the process of creating those evaluation tasks and uploading the relevant content to MTurk to be judged, without having to familiarize and involve oneself with the mechanics, if you will, of Mechanical Turk. This would allow you to spend more time worrying about improving your system rather than dealing with file input and output and MTurk's sometimes finicky interface.

2 Overview

MAISE is a collection of tools for Mass AI System Evaluation. MAISE allows you to evaluate the output of different systems (and/or different variations of a system) using the workforce of Amazon's Mechanical Turk (MTurk). MAISE can be used to compare two simple variants of the same system, working with a couple of variations of your task, or it can be used to perform complete evaluation campaigns involving tens of systems and many variations.

The core of MAISE's codebase was written to run the manual component of WMT10's evaluation campaign. In the manual evaluation, various MT systems are directly compared to each other, by annotators who indicate which systems produce better outputs (i.e. better translations). Starting in 2010, the evaluation moved from using a locally hosted web server, and onto MTurk, taking advantage of MTurk's existing infrastructure, and making available the option to collect data from a large pool of annotators, if desired, rather than relying solely on recruited volunteers. That evaluation campaign involved around 170 submissions over eight different language pairs. In 2011, the number increased to 190 submissions over ten language pairs.

We note here that although MAISE was written with MT in mind, it **can** be used for other ML/AI tasks as well. Some of the supported features are meant to make MT evaluation easier (e.g. MAISE is aware of which language is being translated to and from), but those could simply be ignored for other tasks. As long as the task has some concept of 'input' and some concept of 'output' (e.g. a foreign sentence and a machine translation), then MAISE is appropriate.

Given this paper's venue of publication, the remainder of the paper assumes the task at hand is machine translation.

3 The Mechanics of MAISE

The components of MAISE have been designed to completely eliminate the need to write any data processing code, and to minimize the need for the user to perform any manual tasks on MTurk's interface, since MAISE facilitates communication with MTurk. Whenever MAISE needs to communicate with MTurk, it will rely on MTurk's Java SDK, which is already included in the MAISE release (allowed under the SDK's license, Apache License V2.0).

Once you create your evaluation tasks and upload the necessary content to MTurk, workers will begin to complete the corresponding HITs. On a regular (e.g. daily) basis, you will tell MAISE to retrieve the new judgments that workers provided since the last time MAISE checked. The process continues until either all your tasks are completed, or you decide you have enough judgments.

You can use MAISE with any evaluation setup you like, as long as you design the user interface for it. Currently, MAISE comes with existing support for a particular evaluation setup that asks annotators to rank the outputs of different systems relative to each other. When we say "existing support" we mean the user interface is included, and so is an analysis tool that can make sense of the judgments. This way, you don't need to do anything extra to obtain rankings of the systems. You can read more about this evaluation setup in the overview papers of the Workshop on Statistical Machine Translation (WMT) for the past two years.

3.1 Requirements and Setup

MAISE is quite easy to use. Beyond compiling a few Java programs, there is no need to install anything, modify environment variables, etc. Furthermore, since it is Java-based, it is completely platform-independent.

To use MAISE, you will need:

- Java 6
- Apache Ant
- A hosting location (where you place certain HTML files)
- An MTurk Requester account

You will also need an active Internet connection whenever new tasks need to be uploaded to MTurk, and whenever judgments need to be collected from MTurk. The setup details are beyond the scope of this paper, but are straightforward, and can be found in MAISE's documentation, including guidance with all the MTurk-related administrative issues (e.g. the last point in the above list).

3.2 Essential Files

MAISE will assume that the user has a certain set of "essential files" that contain all the needed information to perform an evaluation. These files are:

1) **The system outputs** should be in plain text format, one file per system. The filenames should follow the pattern `PROJECT.xx-yy.sysname`, where `PROJECT` is any identifying string chosen by the user, `xx` is a short name for the source language, and `yy` is a short name for the the target language.

2) **The source files** should be in plain text as well, one file per language pair. The source filenames should follow the pattern `PROJECT.xx-yy.src`, where `PROJECT` matches the identifying string used in the submission filenames. (The contents of such a file are in the `xx` language.)

3) **The reference files**, also one per language pair, with filenames `PROJECT.xx-yy.ref`. (The contents of such a file are in the `yy` language.)

4) **A specification file** that contains values for various parameters about the project (e.g. the location of the above files).

5) **A batch details file** that contains information about the desired number of MTurk tasks and their particular properties.

As one could see, the user need only provide the bare minimum to get their evaluation started. More details about items (4) and (5) are provided in the documentation. Essentially, they are easily readable and editable files, and all the user needs to do to create them is to fill out the provided templates.

3.3 The Components of MAISE

There are three main steps necessary to perform an evaluation on MTurk: **create** the evaluation tasks, **upload** them to MTurk, and **retrieve** answers for them. Each of those three steps corresponds to a single component in MAISE.

3.3.1 The `BatchCreator`

The first step is to create some input files for MTurk: the files that contain actual instantiations of our tasks, with actual sentences. This will be the first step that requires you to make some real executive decisions regarding your tasks. Among other things, you will decide how many judgments to collect and who to allow to give you those judgments.

Each **batch** corresponds to a single task on MTurk. Typically, each batch corresponds to a single language pair. So, if you are performing a full evaluation campaign, you would be creating as many batches as there are language pairs. If you are merely comparing several variants of the same system, say, for Arabic-English, you would probably have just one batch.

That said, you may have more than one batch for the same language pair, that nonetheless differ in other properties. In fact, each batch has a number of settings that need to be specified, including:

1) what language pair does this batch involve?

2) how many HITs does this batch include?

3) how many times should each HIT be completed?

4) what is the reward per assignment?

5) what are the qualifications necessary for an annotator to be allowed to perform the task (e.g. location, approval rating)?

Those settings are all specified in a single file, the abovementioned **batch details file**. The user them simply runs the `BatchCreator` component, which processes all this information and creates the necessary files for each batch.

3.3.2 The `Uploader`

After the `BatchCreator` creates the different files for the different batches, those files must be uploaded to MTurk in order to create the various batches. There will be a single file, called the *upload info file*, that contains the locations of the files to be uploaded. The upload info file is created automatically, and all the user needs to do is pass it as a parameter to the next MAISE component, the `Uploader`.

The `Uploader` communicates with MTurk via a web connection. Once it has completed execution, HITs for your tasks will start to appear on the MTurk website, available for MTurk's workers to view and complete them.

3.3.3 The `Retriever`

At this point, you would be waiting for Turkers to find your task and start accepting HITs and completing them. You can retrieve those answers by using another MAISE component that communicates with MTurk called the `Retriever`. It can be instructed to retrieve all answers for your HITs or only a subset of them. It retrieves all the answers for those HITs, and appends those answers to an answer log file.

Note that the `Retriever` does not necessarily approve any of the newly submitted assignments. It can be instructed to explicitly retrieve those answers without approving them, giving you the chance to first review them for quality. Alternatively, it can be instructed to approve the assignments as it retrieves them, and also to reject certain assignments or certain annotators that you have identified as being of sub-par quality. All this information is placed in plain text files, easy to create and maintain.

When you use MAISE to perform an actual evaluation on MTurk, you should run the `Retriever` fairly regularly, perhaps once every day or two. Each time, review the retrieved results, and rerun the `Retriever` in "decision mode" enabled, to aprove/reject the pending submissions.

4 Analyzing the Results: An Example

Once the tasks have been completed, all the answers will have been written into an *answers log file*. The log file is in plain format, and contains extensive information about each HIT completed, including a worker ID, time required to complete, and, of course, the answers themselves. Naturally, analyzing the results of the evaluation depends on what the task was, and what the interface you designed looks like. You can write your own code to read the log file and make sense out of them.

MAISE already comes equipped with an analysis tool for one particular task: the *ranking task*. In this setup, the annotator evaluates system outputs by **ranking** them from best to worst. The rank labels are interpreted as pairwise comparisons (e.g. 5 rank labels correspond to $\binom{5}{2} = 10$ pairwise comparisons), and each system is assigned a score reflecting how often it wins those pairwise comparisons. This is the setup used in the evaluation campaigns of WMT10 and WMT11.

The analysis tool takes as input the answers log file as is, and extracts from it all the rank labels. Each system's score is computed, and the tool produces a table for each language pair displaying the participating systems, in descending order of their scores. It also creates an additional *head-to-head* table, that summarizes for a specific pair of systems how often each system outranked the other. The output is created in HTML format, for easy viewing in a browser.

Furthermore, the tool produces a detailed **worker profile table**. Each row in this table corresponds to one worker, identified by their Amazon worker ID, and includes certain measures that can help guide you identify bad workers, who are either clicking randomly, or perhaps simply not doing the task properly. Those measures include:

- **Average time required per HIT**: a suspiciously fast annotator might not be performing the task diligently.
- **The *reference preference rate* (RPR)**: how often did the annotator correctly prefer an embedded reference translation; a low RPR almost certainly indicates random clicking, with typical good values at 0.97 and up.
- **Prevalence of tied rank labels**: an overly high percentage of tied comparisons indicates an overly 'conservative' worker, hesitant to distinguish between outputs.
- **The annotator's intra-annotator agreement**: i.e. the annotator's consistency with themselves, based on how often they repeated the same judgment when comparing the same system pair.

To appreciate the tool's output, the reader is encouraged to view the results of a real-life evaluation campaign at `http://bit.ly/jJYzkO`. These are results of analyzing 85,000+ rank labels in an evaluation campaign of 40+ MT systems over six language pairs.

5 Download and Licensing

MAISE can be obtained from the author's webpage: `http://cs.jhu.edu/~ozaidan/maise/`. The release includes MAISE's source code, instructions, documentation, and a tutorial. MAISE is an open-source tool, licensed under the terms of the GNU Lesser General Public License (LGPL). Therefore, it is free for personal and scientific use by individuals and/or research groups. It may not be modified or redistributed, publicly or privately, unless the licensing terms are observed. If in doubt, contact the author for clarification and/or an explicit permission. The distribution also includes the MTurk Java SDK v1.2.2, which is licensed under the terms of the Apache License V2.0.

Acknowledgments

I developed MAISE while I was funded by DARPA's GALE Program, and in part by the EuroMatrixPlus Project. I would like to thank Chris Callison-Burch, Ondřej Bojar, and everybody who gave feedback during the WMT10 evaluation campaign. More importantly, much thanks goes to Josh Schroeder, the author of the previous implementation used pre-2010, who helped me navigate his code when I first started reimplementing the data processing components.

References

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. 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, Uppsala, Sweden, July. Association for Computational Linguistics.