EasyTurk: A User-Friendly Interface for High-Quality Linguistic Annotation with Amazon Mechanical Turk

Lorenzo Bocchi, Valentino Frasnelli
Dept. of Psychology and Cognitive Science
University of Trento
Rovereto (Trento), Italy
[name.surname]@studenti.unitn.it

Alessio Palmero Aprosio
Digital Humanities Unit
Fondazione Bruno Kessler
Trento, Italy
aprosio@fbk.eu

Abstract

Amazon Mechanical Turk (AMT) has recently become one of the most popular crowdsourcing platforms, allowing researchers from all over the world to create linguistic datasets quickly and at a relatively low cost. Amazon provides both a web interface and an API for AMT, but they are not very user-friendly and miss some features that can be useful for NLP researchers. In this paper, we present EasyTurk, a free tool that improves the potential of Amazon Mechanical Turk by adding to it some new features. The tool is free and released under an open source license.

A video showing EasyTurk and its features is available on YouTube.[1]

1 Introduction

In the last years, deep learning algorithms have achieved state-of-the-art results in most NLP tasks such as textual inference, machine translation, hate speech detection (Socher et al., 2012). Despite their accuracy, deep learning algorithms have a major downside, i.e. they require large amounts of data to be trained, making the data bottleneck issue even more problematic than with other machine learning algorithms like SVM (Gheisari et al., 2017). The need to leverage large amounts of manually annotated data has become a major challenge for the NLP community, since linguistic annotation performed by domain experts is both expensive and time-consuming. This explains why crowdsourcing platforms, offering access to a large pool of potential annotators, have been successfully used for the creation of annotated datasets.

Amazon Mechanical Turk (AMT) is probably the most widely used platform of this kind, enabling the distribution of low-skill but difficult-to-automate tasks to a network of humans who could work in parallel, when and where they prefer, for a certain amount of money. The availability of a lot of workers at the same time allows researchers all over the world to annotate large datasets in a fraction of the time and the money needed doing it through the recruitment of domain experts. Furthermore, crowd-workers are spread all over the world, offering the possibility to have annotation performed in different languages by native speakers. In the last years, AMT turned out to be successful in a wide range of NLP annotations, such as named entities from e-mails (Lawson et al., 2010) or medical texts (Yetisgen et al., 2010), subjectivity word sense disambiguation (Akkaya et al., 2010), image captioning (Rashtchian et al., 2010), and much more.

Unfortunately, annotations obtained by AMT workers are often of low quality, since: (i) they are non-expert and therefore they can make mistakes in annotations; (ii) some of them are *spammers* who try to maximise the earnings by submitting random answers as quickly as possible. Mitigating the effect of errors in datasets annotated by crowd-workers is one of the biggest challenge in using AMT. One mitigation strategy adopted by researchers is usually to collect multiple annotations of the same instance, and apply different methods to deal with this information redundancy. Most of the times, majority voting seems to be an appropriate strategy, i.e. the final label assigned to an instance is the one provided by the majority of the workers, even if they are not all in agreement. However, if spammers always choose the same answer to finish the task quicker, this strategy would finally assign a wrong label to the textual instance.

While past works have described how to successfully deal with non-expertness (Callison-Burch, 2009; Mohammad and Turney, 2010), it is more challenging to identify spammers. Some tools (Hovy et al., 2013) deal with the problem offline,

[1]https://youtu.be/OmKJOrNpGSs

when the task is completed, trying to identify spammers using redundant annotations and comparing the answers given by all crowd-workers. In this context, spammers are correctly identified, but they are nevertheless paid because their annotations are filtered out after the task is closed.

Another idea to find spammers is to use a gold standard, a set of very easy-to-understand instances, previously annotated by an expert, that a careful worker should not miss. In this paradigm, when a worker gives the wrong answer to a gold question, one may infer that the annotator is trying to cheat and should be blocked. The AMT API provide a way to do it automatically, but the feature is not included in the web interface, therefore the only way to get this result is by writing a program (in Python, php, or any supported language) that checks whether the gold instances have been answered correctly or not.

In this paper, we describe EasyTurk, both a web interface and a powerful API that tackles all these issues and enhances the experience of using AMT. The tool can aggregate more than one instance of a task in a single page shown to the worker, concealing also gold standard instances. Furthermore, EasyTurk can be configured to take an action, e.g. block a worker when he or she misses too many gold answers, marking the already-given questions as not reliable. Finally, the software is open source and its user-friendly interface has been implemented using most recent guidelines for usability and responsiveness.

2 Amazon Mechanical Turk

Amazon Mechanical Turk[2] is an online marketplace for hiring workers and submit to them atomic tasks that are usually easy for humans but difficult for machines. The atomic unit of work is called *Human Intelligent Task* (HIT).

AMT has two kinds of users: requesters and workers. The formers create the HITs (using the API or the web interface) and upload them to the Amazon servers, along with the fee that will pay for each of them to be completed. The latters search the HIT database, choose the preferred tasks and complete them in exchange for monetary compensation.

Requesters can restrict the range of workers allowed to complete the task, based on demography, school level, spoken languages, and so on. Some requirements are free for the requester (for example the living country of the worker), but normally they raise the price of the HITs. Requesters can also assign custom qualifications to workers in order to filter out them during the submission of the HITs to the system.

The platform also provides an automatic mechanism that allows multiple unique workers to complete the same HIT. This is useful, for example in NLP tasks, for which requesters usually need more than one answer for each HIT, so that the majority label can be selected, resulting in a higher-quality final annotation thanks to the 'wisdom of the crowd'. Each annotation instance (a pair worker-HIT) is called *assignment*.

Requesters have the option of rejecting the answer of a particular worker, in which case they are not paid. The above-described custom qualifications can be used to filter out, for a particular task, workers who did not reach sufficient accuracy in previous HITs. In specific cases, for example as a consequence of particularly sloppy annotations, a worker can be blocked and is not able to perform HITs for the requester anymore.

One of the main issues with using AMT is that some features are available only using the API, while others can be used only in the web interface. For example, through the web interface a requester can upload a TSV file with the data to be annotated, or select which qualifications the workers should have to complete the HITs. These two features are not available in the API, but one can automatise acceptance/rejection of the worker job only through it. Given the above constraints, we developed EasyTurk so to allow non-skilled users to submit HITs without using a specific programming language, such as Python or Java, while using the features available through APIs.

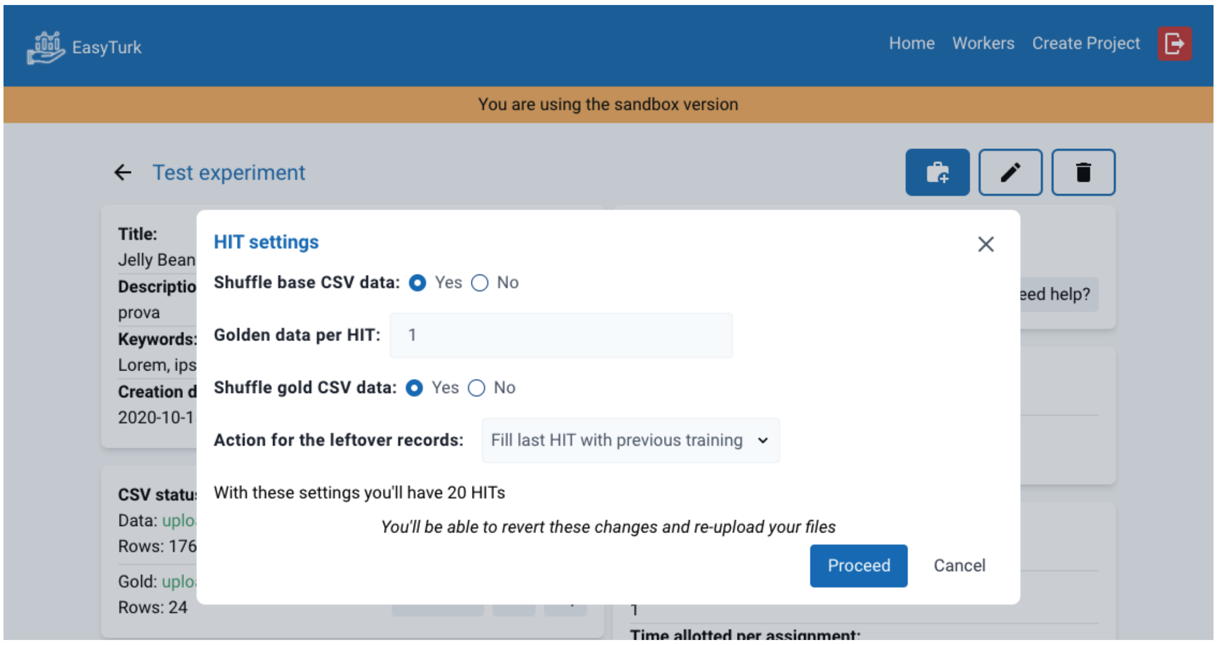
3 Description of EasyTurk

EasyTurk is composed of three modules: (i) the web interface; (ii) the API; (iii) the server. Most of the features included in EasyTurk are accessible directly from the web interface, but are managed by the server.

3.1 More annotations in one HIT

The original web interface of AMT has a powerful graphical editor for the templates, used by the requester to display the data they want the worker to annotate. After creating the template file, one

[2] http://www.mturk.com/



HIT settings

Shuffle base CSV data: [✓ Yes] No

Golden data per HIT: 1

Shuffle gold CSV data: [✓ Yes] No

Action for the leftover records: Fill last HIT with previous training

With these settings you'll have 20 HITs

You'll be able to revert these changes and re-upload your files

Figure 1: Selection box for mixing gold and unknown data.

can upload a text document with the data (usually a CSV or XML file), and then AMT submits the HITs (one per file record/line) to the workforce.

In NLP, it often happens that a task corresponds to a binary assignment, meaning that an instance is labeled with a value in the set true/false. Usually researchers have a list of instances in one single file (for example a JSON or CSV file). Submitting the record one by one, one per HIT, would be more expensive for the requester and time-consuming for workers, because they would need to click the confirm button after each instance annotation and wait for the new HIT to load, even if it is just a sentence or a short string.

In EasyTurk the requester can go beyond this limitation easily, by creating a template with multiple slots for the data. Then, using a sequential naming standard (for example, `text1`, `text2`, `text3`, etc.), the tool will automatically infer the number of records to fill in the template.

3.2 Upload of a gold standard

In AMT, the requester has two options to check the annotation accuracy. First, they can perform an offline check (after the whole task has ended) using the information obtained by majority voting (Hovy et al., 2013). As an alternative, AMT provides a mechanism to check the answer of a HIT against a gold standard. Depending on the worker answer, the system can accept or reject the HIT automatically. As outlined in Section 2, this is one of the features available only in the API, and missing in the web interface.

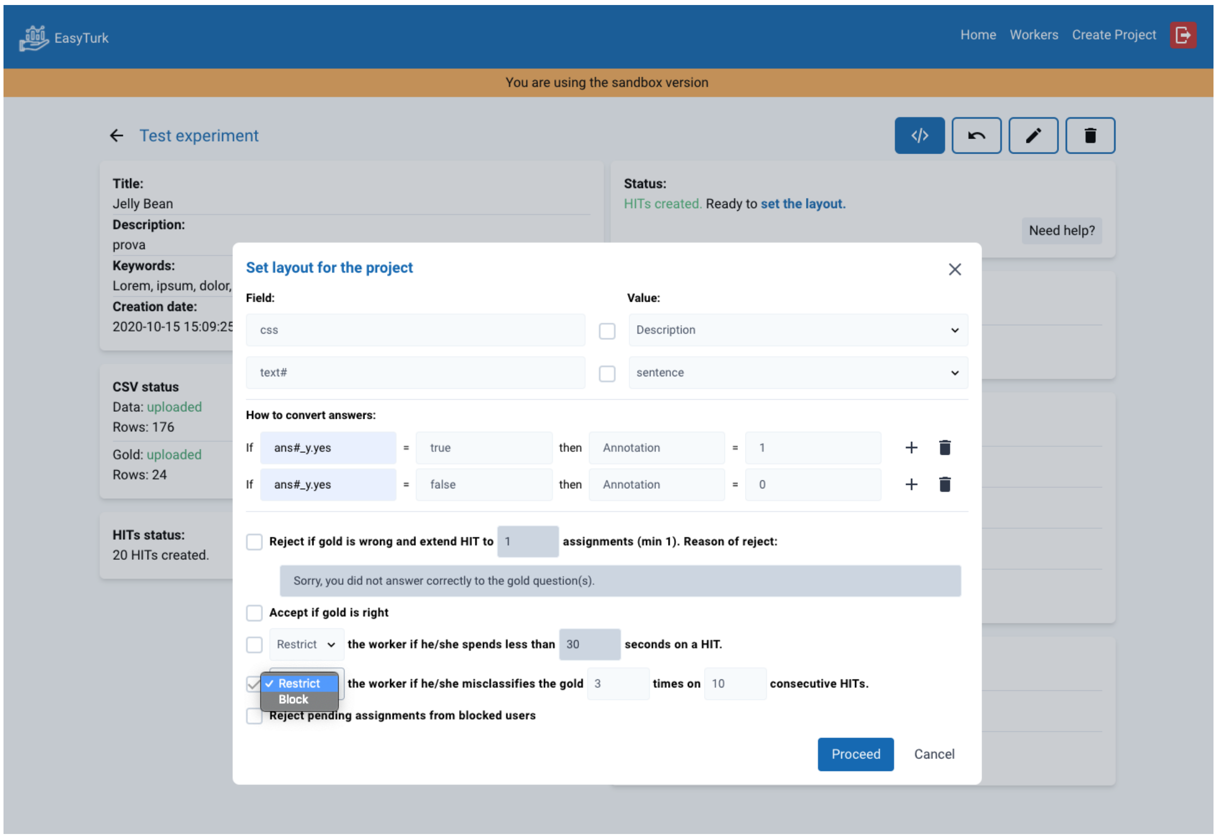
In EasyTurk, the requester can optionally add a document with some additional data containing the correct annotation. When populating the template, they can select how many gold instances need to be added for each HIT (see Figure 1), and decide - among a set of available options - the behavior of the system when the worker misses the gold instance(s).

In order to avoid that a worker is blocked or restricted for having missed a single answer, the system can check the accuracy of the workers on a span of HITs, and then take action after the worker completed at least that span (see Figure 2).

3.3 Automatic block/restrict the workers

When a worker misses a considerable amount of gold instances, the requester can decide what will be the behavior of the tool. Figure 2 shows the range of possible options. First of all, one has to decide whether to accept or reject the assignment. In the second case, the worker can be *restricted* or *blocked*. With *restriction*, it is intended that this worker cannot participate any more in the tasks of the current project, but they are allowed to complete HITs when a new project from the same requester is submitted to AMT. EasyTurk uses AMT *qualifications* to this purpose.[3] When a worker is

[3]A *qualification* is a custom property that a requester can assign to one or more workers. In EasyTurk, each project is as-



You are using the sandbox version

Test experiment

Title:
Jelly Bean
Description:
prova
Keywords:
Lorem, ipsum, dolor,
Creation date:
2020-10-15 15:09:25

CSV status
Data: uploaded
Rows: 176
Gold: uploaded
Rows: 24

HITs status:
20 HITs created.

Status:
HITs created. Ready to set the layout.

Set layout for the project

Field:	Value:
css	Description
text#	sentence

How to convert answers:

If	ans#_y.yes	=	true	then	Annotation	=	1
If	ans#_y.yes	=	false	then	Annotation	=	0

Reject if gold is wrong and extend HIT to 1 assignments (min 1). Reason of reject:
Sorry, you did not answer correctly to the gold question(s).

Accept if gold is right

Restrict the worker if he/she spends less than 30 seconds on a HIT.

[✓ Restrict] Block the worker if he/she misclassifies the gold 3 times on 10 consecutive HITs.

Reject pending assignments from blocked users

Figure 2: Selection box for managing the behavior of the tool depending on the workers' answers.

blocked, instead, they will not see any more any HIT submitted by the requester. Both properties (restriction and block) are reversible.

To limit spammers (see Section 1), a worker can be blocked/restricted also when HITs are being submitted by a worker too fast, showing for example that the worker is not even reading the instances before annotation.

3.4 User management

When running EasyTurk, the user is asked to provide an administration password. With this credentials, the administrator can create new users, each of which having its own username and password. Each user is then linked to its AMT API keys, allowing a single instance of EasyTurk to serve different users having different AMT accounts. A flag can be set to switch a user to work on the Sandbox version of AMT.

sociated with a qualification: when a requester wants to restrict a worker, the tool assigns the qualification to the worker, and consequently the task is hidden in the AMT worker console for them.

3.5 The web interface

The web interface of EasyTurk is written using VueJS.[4] The structure of the website is build with Tailwind CSS[5], the design is inspired by Material Design.[6]

Through the interface, requesters can group HITs into projects, and follow all the steps from the project definition to the visualisation of the results.

Project definition. The general information about the project (description, reward, time alotted for the workers, layout, qualifications needed, and so on) are given and a project is created.

Data insertion. In this phase, a file with the data is uploaded to the system (plus an additional file, if needed, for the gold standard, see Section 3.2).

HITs generation. The HITs are generated by grouping the data (depending on how many items the requester wants for each HIT) and optionally mixing it with the gold standard (Figure 1).

[4] https://vuejs.org/
[5] https://tailwindcss.com/
[6] https://material.io/

Condition management. The requester sets the tool behaviour in specific cases, for instance when a worker misses the gold standard (Figure 2).

HITs submission. The HITs are submitted to AMT in bunches of predetermined size (set by the requester).

HITs monitoring. The dot matrix interface gives an overview on how the task is going (see Figure 3). In this phase, the requester can control all the aspects of the annotations: the approval rate, the speed, the workers, and so on.

Retrieval of Results. The resulting annotations (even when the gold is missed or the HIT is rejected) can be visualised and downloaded in JSON format.

In developing EasyTurk, we wanted to stress the importance of having a readable overview of how the annotation is going, from the HITs submission to the retrieval of the results. We found the dot matrix chart[7] to be an effective solution to achieve this goal (see Figure 3). Each dot represents a HIT and is painted with a different color depending on how many assignments have been rejected or whether the gold instances have been missed. Different colorization strategies have been chosen to highlight the different status of the HITs: unassigned, pending, completed. Using this interface, a considerable presence of red dots may point out that the gold standard was ambiguous, allowing the requester to tune it better in the future.

3.6 The API

An API supporting the web interface and written in php is included in the EasyTurk package. It can be used also as a standalone program to integrate the features of the tool into third-part packages. Since the web interface relies on this API to work properly, it is mandatory to install it to take advantage of the web interface.

3.7 The server

The last part of EasyTurk is a server script, written in php. It performs all the tasks needed to update the information based on the AMT APIs (for example, the status of a HIT or the triggering of the actions described in Section 3.3).

[7]https://datascientist.reviews/dot-matrix-chart/

EasyTurk can also be configured to work with Amazon Simple Notification Service[8] (SNS), so that most of the information about the HITs can be updated almost in real time.

4 Release

EasyTurk is completely free, available on GitHub,[9] and released as open-source under the Apache 2.0 license.[10] The web interface is developed in VueJS and needs NodeJS[11] to be compiled and launched.

Both the API and the server are written in php[12] and need a machine with at least version 7 of the interpreter and MySQL server[13] installed. The server can be run as a service and does not need other particular dependencies to work. The API, instead, must be configured to work in a web server (such as Apache[14] or Nginx[15]).

5 Related Work

Since 2005, when AMT was released, an increasing number of researchers has used this platform for research purposes. In particular, the NLP community has taken advantage of AMT to bring linguistic resources to a new scale, also with the support of Amazon. For example, in 2010 Amazon sponsored a workshop during the NAACL conference, where researchers were given 100 dollars of credit on the platform to run an annotation task and answer some meta-research questions, such as how non-expert workers can perform complex annotations, or how can one ensure high quality annotations from crowd-sourced contributors.

Some past works have dealt with the above-mentioned issues related to crowd-worker quality. In (Hovy et al., 2013), the authors present a software that, after a round of annotations using AMT, tries to understand which workers perform better and, consequently, which are the best annotations to consider and which to discard when there is redundancy, in an unsupervised fashion. In (Wais et al., 2010), the efficiency of AMT is analysed over 100,000 local business listings for an online directory. A mechanism for filtering low-quality workers in order to build a reliable workforce that

[8]https://aws.amazon.com/it/sns/
[9]https://github.com/dhfbk/easyturk
[10]https://www.apache.org/licenses
[11]https://nodejs.org/it/
[12]https://www.php.net/
[13]https://www.mysql.com/
[14]https://httpd.apache.org/
[15]https://www.nginx.com/

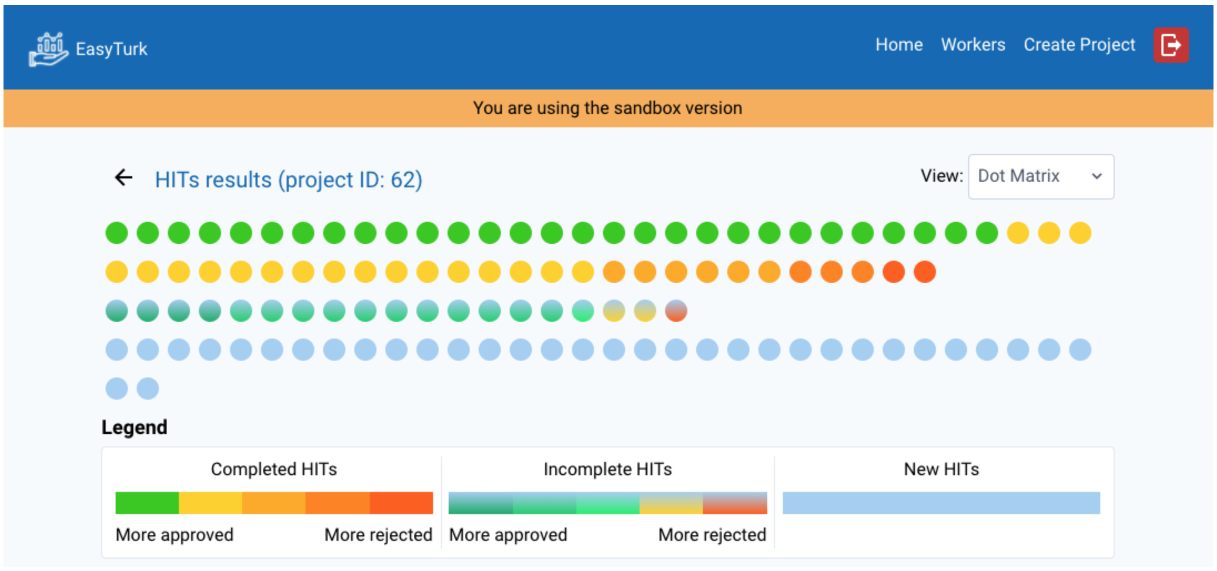


Figure 3: The dot matrix showing the HITs.

has high accuracy is described, to understand better the problem of quality control in crowdsourcing systems.

Some attempts have also been done to improve the potential of AMT by writing new frameworks on top of the AMT API. CloudResearch, formerly TurkPrime, (Litman et al., 2017) was born for this purpose and at the time of launch was free to use for researchers. Now it is part of a bigger company and is not free any more. LingoTurk (Pusse et al., 2016) is an open-source, freely available crowdsourcing client/server system aimed primarily at psycholinguistic experimentation, where custom and specialized user interfaces are required but not supported by popular crowdsourcing task management platforms. OpenMTurk (Feeney et al., 2018) is a free and open-source administration tool for managing research studies using AMT. TurKit (Little et al., 2010) is a toolkit for prototyping and exploring truly algorithmic human computation, while maintaining a straightforward imperative programming style. Turktools (Erlewine and Kotek, 2016) is a set of free, open-source tools that allow linguists to post studies online and simplify the interaction with AMT. TurkGate[16] provides better control and verification of workers' access to an external site and allows the grouping of HITs, so that workers may only access one survey within a group. AMTI,[17] developed at the Allen Institute for AI, is a command-line interface for AMT that emphasizes the ability to quickly iterate on and run reproducible crowdsourcing experiments.

Finally, AMT is integrated to add human annotations in more complex tools. Qurk (Marcus et al., 2011), for example, is a query system for managing annotation workflows.

6 Conclusion and Future Work

In this paper, we presented EasyTurk, a free program that improves the potential of Amazon Mechanical Turk by adding some features which are not present out-of-the-box. In particular, the requester has now the ability to insert multiple instances of the task in a single HIT, and optionally mix them with a gold standard, that can be used to track the accuracy of the workers. Finally, when some events are triggered (for example a worker answering too quickly to a HIT or missing the gold standard), EasyTurk can be programmed to take an action such as reject the assignment, or block/restrict the worker.

The tool is free and open source, and can be downloaded from GitHub and installed locally.

In the future, we are planning to implement new features. For example, the system can intercept spammers using also a particular pattern of answers (for example a set of HIT where the same answer is always selected). We also would like to include in EasyTurk a collection of templates for basic annotations (for example, yes/no, a set of possible answers, a free text, and so on), so that requesters do not need any more to create their template on the AMT website.

[16] `https://github.com/gideongoldin/TurkGate`
[17] `https://github.com/allenai/amti`

References

Cem Akkaya, Alexander Conrad, Janyce Wiebe, and Rada Mihalcea. 2010. Amazon mechanical turk for subjectivity word sense disambiguation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 195–203, Los Angeles. Association for Computational Linguistics.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, page 286–295, USA. Association for Computational Linguistics.

Michael Yoshitaka Erlewine and Hadas Kotek. 2016. A streamlined approach to online linguistic surveys. *Natural Language & Linguistic Theory*, 34(2):481–495.

Justin Feeney, Gordon Pennycook, and Matthew Boxtel. 2018. OpenMTurk: An Open-Source Administration Tool for Designing Robust MTurk Studies. *SSRN Electronic Journal*.

M. Gheisari, G. Wang, and M. Z. A. Bhuiyan. 2017. A survey on deep learning in big data. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 2, pages 173–180.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Nolan Lawson, Kevin Eustice, Mike Perkowitz, and Meliha Yetisgen-Yildiz. 2010. Annotating large email datasets for named entity recognition with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 71–79, Los Angeles. Association for Computational Linguistics.

Leib Litman, Jonathan Robinson, and Tzvi Abberbock. 2017. TurkPrime.com: A versatile crowdsourcing data acquisition platform for the behavioral sciences. *Behavior Research Methods*, 49(2):433–442.

Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, page 57–66, New York, NY, USA. Association for Computing Machinery.

Adam Marcus, Eugene Wu, Samuel Madden, and Robert Miller. 2011. Crowdsourced databases: Query processing with people. pages 211–214.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, Los Angeles, CA. Association for Computational Linguistics.

Florian Pusse, Asad Sayeed, and Vera Demberg. 2016. LingoTurk: managing crowdsourced tasks for psycholinguistics. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 57–61, San Diego, California. Association for Computational Linguistics.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using Amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147, Los Angeles. Association for Computational Linguistics.

Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, ACL '12, page 5, USA. Association for Computational Linguistics.

Paul Wais, Shivaram Lingamneni, Duncan Cook, Jason Fennell, Benjamin Goldenberg, Daniel Lubarov, David Marin, and Hari Simons. 2010. Towards building a high-quality workforce with mechanical turk. In *In Proc. NIPS Workshop on Computational Social Science and the Wisdom of Crowds*.

Meliha Yetisgen, Imre Solti, Fei Xia, and Scott Halgrim. 2010. Preliminary experience with amazon's mechanical turk for annotating medical named entities.