Word Sense Inventories by Non-experts

Anna Rumshisky[†], Nick Botchan[*], Sophie Kushkuley[*], James Pustejovsky[*]

[*]Brandeis University, [†]MIT CSAIL
Waltham, MA, Cambridge, MA
arum@csail.mit.edu, nbotchan@brandeis.edu, sophiek@brandeis.edu, jamesp@cs.brandeis.edu

Abstract

In this paper, we explore different strategies for implementing a crowdsourcing methodology for a single-step construction of an empirically-derived sense inventory and the corresponding sense-annotated corpus. We report on the crowdsourcing experiments using implementation strategies with different HIT costs, worker qualification testing, and locale restrictions. We describe multiple adjustments required to ensure successful HIT completion, given significant changes within the crowdsourcing community over the past three years.

Keywords: word sense disambiguation, crowdsourcing, sense inventory development

1. Introduction

Word sense disambiguation (WSD) is an essential component of many natural language processing (NLP) applications. However, defining a discrete inventory of word senses can be a difficult task. Historically, this task has required extensive labor by human experts, and the end result nonetheless usually contained many sense distinctions that were ad hoc and very often not well supported by corpus data. The procedure followed by the lexicographers in creating dictionaries warranted this situation and, as all resource creation, was slow and costly. Furthermore, the resulting resources are not always sufficient for addressing the dynamic nature of sense definition or quantitatively capturing how components from several meanings of a word can be activated simultaneously (Hanks, 2000; Rumshisky, 2008; Pustejovsky, 1995). A number of efforts have been undertaken to address this situation, including the efforts to re-organize and merge existing sense inventories or create them from scratch using corpus-based methods (Hovy et al., 2006; Hanks and Pustejovsky, 2005; Navigli, 2006; Palmer et al., 2007).

In this paper, we describe a method of creating robust word sense inventories from scratch that is relatively cheap and fast using Amazon's Mechanical Turk (MTurk) (http://aws.amazon.com/mturk/). In recent years, much attention in the NLP community has been given to the subject of using crowdsourcing services such as MTurk to create low cost, high-quality data resources (C. and Dredze, 2010; Snow et al., 2008; Ipeirotis et al., 2010; Akkaya et al., 2010). In this paper, we attempt to extend that research to the specific sub-problem discussed above of creating a robust lexical resource, and will focus on the methodological and experimental considerations that we have had to take into account to build a working data collection system on top of MTurk.

2. Overview of Methodology

Mechanical Turk, a division of the Amazon Web Services, provides an application programming interface and user interface that allows a requester to create a set of Human Intelligence Tasks (HITs) to be filled out by workers (henceforth, MTurkers) for a reward. This allows a large audience to access HITs and complete work at a minimal cost.

We used boto (http:boto.s3.amazonaws.com), the Python interface to AWS, to build an application capable of creating and publishing HITs, collecting results and creating sense clusters. The resulting word sense inventories contain precise consistency ratings and distance measurements between different clusters of word senses. We extend the technique we proposed previously (Rumshisky et al., 2009; Rumshisky, 2011) and which we have developed to explore the quality control mechanisms in preparation for a large-scale lexical development effort.

Our approach to creating new sense inventories relies on the linguistic intuition of non-expert native English speakers, who provide the core judgments used to define word sense clusters. Multiple MTurkers are asked to compare pairs of example sentences that use the same word, and make a judgment on whether they think the word in consideration is being used in the same sense, in a different sense, or if the distinction between the two is unclear.

Target Sentence	A United Nations conference on wiping out organized crime **opens** Tuesday in Sicily.
Comparison Sentence 1	The enclave was **opened** to the outside world more than 300 years before Hong Kong.
Comparison Sentence 2	It was perfectly correct for the front airbag not to **open**, because the front of the car was not damaged at all.
Comparison Sentence 3	Chancellor Kohl, meanwhile, has invited Mr. Krenz to **open** discussions with Bonn on a wide range of subjects.

Table 1: Example HIT using OntoNotes data for the word *open*

The resulting data lends itself to an undirected graph representation in which nodes are example sentences, edges are judgments between pairs of sentences, and interconnected clusters of nodes correspond to different senses for a word (see Figure 1).

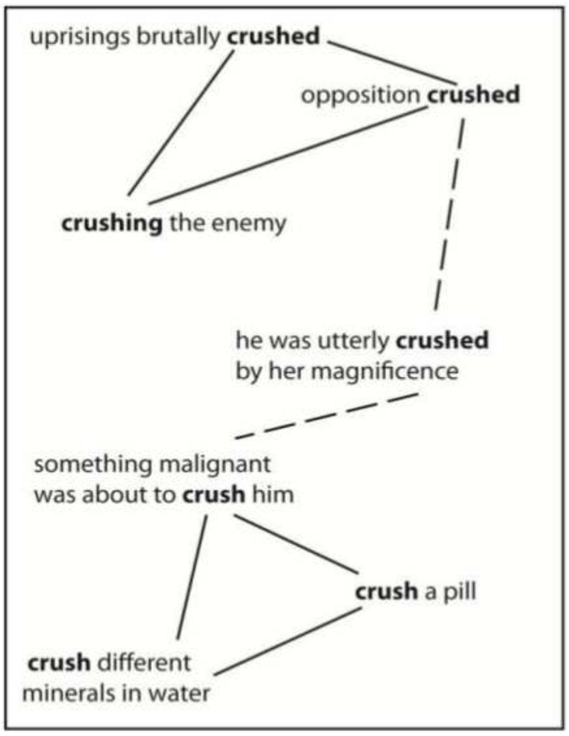
uprisings brutally crushed
opposition crushed
crushing the enemy
he was utterly crushed by her magnificence
something malignant was about to crush him
crush a pill
crush different minerals in water

Figure 1: Similarity judgment graph.

Ideally, comparisons between every pair of example sentences, i.e. full-pairwise comparisons, for a word would be collected, populating the complete graph. However, this leads to a number of comparisons that is quadratic in order, and therefore prohibitively expensive. To circumvent this issue, on top of collecting full-pairwise comparisons, we have implemented an alternative prototype-based methodology that populates a partial graph while explicitly creating sense clusters. Both methodologies employ control mechanisms to ensure high quality annotations.

In the prototype based framework, the annotation task is designed as a sequence of rounds, with each round resulting in a cluster corresponding to one sense of a word. To begin, a prototype sentence is selected at random from the list of unclustered sentences. HITs are then created in which every remaining unclustered sentence is compared to the prototype sentence.

When all of the corresponding judgments are returned, all of the sentences that were judged as similar to the prototype get clustered together, and the round is complete. This process is repeated until all sentences are placed in a cluster. The full-pairwise methodology works similarly, however, there is no need to select prototype sentences as every sentence gets compared to every other sentence, and clusters are created using clustering algorithms after all comparisons have been collected.

3. Crowdsourcing Considerations

Much of our methodological experimentation has revolved around determining precisely what it takes to attract high-quality workers on a service like MTurk to complete complex linguistic tasks such as our own. Previous work on this subject point out useful guidelines and best practices (C. and Dredze, 2010; Snow et al., 2008), but due to the flexibility and dynamic nature of MTurk as a data building resource, we found that much of the design and data collection process for our WSD task has required experimentation beyond these best practices. Section 4 of this paper discusses in more detail the experiments we ran with different system configurations.

3.1. Determining Worker Quality

Unfortunately, crowdsourcing services like MTurk are often subject to malicious workers who attempt to take advantage of the system, and as a result submit answers of a low quality. This problem stems from the fact that result verification is often very difficult. In general, it is not enough to analyse the accuracy of a worker against gold standard data. For example, if 80% of the data in a categorization task falls into one category, than a worker who places everything into that category automatically has 80% accuracy, despite the fact that they are not providing any meaningful input to the results.

Our WSD task in particular exhibits the sort of skewed distribution described above. In general, the words we are interested in have a high degree of polysemy, and therefore it is more likely that two different examples of a word in context will come from different senses.

While the MTurk API recently introduced some features to make worker analysis easier, the burden is placed upon task designers to determine the best way of filtering results and identifying good and bad workers. We use an open-source tool developed by Panos Ipeirotis that takes as input a list of worker labels and a corresponding list of gold standard data (if available), and outputs a scalar score representing worker quality (http://code.google.com/p/get-another-label/). One advantage of this software is that it uses but does not require gold standard data, and can estimate worker quality without any gold standard data by comparing worker results to worker-quality weighted majority votes.

3.2. HIT Design

The HIT design that was used to successfully obtain good quality annotation in Rumshisky et al. Rumshisky et al. (2009) gave the workers $.02 for 10 similarity judgments, and the HITs required to sort 350 concordance lines were completed in under two hours. However, the experiments run with the same parameters today do not lead to either fast completion or quality annotation. Below, we describe the issues that arise in running such HITs in the crowdsourcing community as it exists today.

3.2.1. Qualifications and Restricting the Workforce

The MTurk API provides several built in ways to restrict the workforce that is allowed to work on HITs that you create. The most relevant built in restrictions to linguistic and language resource development are: location (the country where workers must be based in order to complete HITs), the number of HITs previously completed, and the percent of previous HITs worked on that got approved.

Best practices suggest using some combination of these restrictions to reduce the risk of malicious workers introducing noise into your data. We found that in particular the location restriction can have a dramatic impact on the quality of results. We observed a significant increase in inter-annotator agreement and data quality when we restricted locale to the US only. This increase in quality came at a cost, however, namely that it took significantly longer for our HITs to complete, and that we had to pay more in general to attract worker attention in locale restricted experiments.

3.2.2. Payment Options and Maintaining a High Position in the HIT Search Space

Workers sort potential HITs to work on by the size of the reward and by the number of HITs available for them to complete. There are a broad range of language tasks available to workers on MTurk, from translation and transcription tasks that pay upwards of $20 per HIT and only have a few HITs available, to classification tasks such as our own, that pay only a few cents per HIT but have thousands of HITs posted.

Amongst all this variation, we found that since our task has a relatively small reward for a relatively small amount of work per HIT, it is important for us to take steps to inflate the number of HITs we post in order to stay high in the HIT search space and get our HITs completed. More difficult to control is the psychological component of how workers will perceive the amount of work you are asking them to do per HIT. Before workers accept a HIT, they have an opportunity to review a randomly selected sample of the work and determine if it is a task they will enjoy or think to be worth their effort. In order to make a good first impression on workers, it is important that HIT design is clean, and that instructions for your task are clear and concise.

3.3. Other Considerations

Due to the fact that requestors have the ability to reject work with impunity, which can result in the abuse of legitimate workers, several resources have been developed that MTurkers use to communicate with each other (turkopticon.com, turkernation.com). These resources make it important for requestors to stay in good standing with the community, and we have found that paying workers as promptly as possible and being able to respond quickly to worker questions and feedback is important for sustaining worker participation.

4. Experiments

In seeking out the optimal system configuration we conducted a series experiments, varying restrictions and quality control mechanisms. Our early efforts focused on fast and cheap prototype-style data collection which failed to yield satisfactory results.

Because our task is an English language task, the first built in MTurk qualification requirement we decided to explore was the locale restriction, which requires workers to be based in a specific country. Unfortunately, the built in locale restriction mechanism for MTurk only allows HIT creators to restrict worker locale to a single country. We ran prototype experiments with the locale restricted to the US, India, and some with no locale restrictions. Our results indicate that restricting to the US produced the highest quality data set for our task. Other work on this subject indicates that it is possible to more precisely restrict the locale of workers allowed to complete your HITs by looking at the IP addresses of workers or asking for workers to report their country of origin (Rand, 2011).

In the US local restricted prototype experiments, which were run with the compensation of $.01 for five judgments, the HITs failed to complete. With only a small amount of HITs visible (initially 30), our HITs were also very low in the MTurk search space. For our next set of experiments we posted a series of full-pairwise HITs, which greatly boosted the number of HITs visible to workers since all the HITs for a full-pairwise graph can be generated at the start of the experiment as opposed to the prototype framework which requires that HITs be generated dynamically after each round. The additional benefit of collecting full-pairwise data is the complete data set allows us to run prototype and clustering simulations, generating results using subsets of edges.

For the full-pairwise HITs we introduced a few of the quality control mechanisms described in section 3 above, namely using worker quality testing software and introducing two built in MTurk qualification requirements to our HITs. These requirements mandated that only workers who had greater than an 85% previous HIT approval rate and who had completed at least 200 HITs were eligible to complete our HITs, an initial filter against bots and chronic spammers.

Lastly, the final experiments we ran tested the effect of increasing the reward per HIT while also increasing the number of comparisons made per HIT, and offering to pay out bonuses to the best workers.

The configuration that we found to work the best was paying 3 cents per 10 judgments in single HIT, allowing a maximum worker error rate of 27%, restricting locale to the US only, requiring workers to have at least an 85% previous HIT approval rate and having completed at least 200 previous HITs, and offering bonuses to the top workers. This configuration yielded a kappa score of 0.69 and an F-score of 0.72. One interesting thing to point out is that this optimal configuration seems to be rather different from our findings in 2009, when similar HITs with more work could be completed at a fraction of the price (Rumshisky, 2009)–evidence that the MTurk marketplace is rapidly changing.

Table 2 shows a summary of all the experiments run under different system configurations.

5. Evaluation

There is a wide variety of evaluation metrics available for determining the quality of clusters compared against a gold standard. Among them we use set matching measurements, pairwise evaluation measures, and entropy/mutual information measurements. For simplicity, we only report f-scores here.

Word	Source	# of Examples	Pairwise / Prototype	Cost per HIT ($)	Comps per HIT	Locale Restriction	Max Error Rate	# HITs	Completed	Total Cost ($)	Kappa	Actual Agreement	F-score
shut	CPA	150	Prototype	0.01	5	None	None	68	Yes	3.40	0.17	0.58	0.27
open	Ontonotes	150	Prototype	0.01	5	None	None	95	Yes	4.75	0.12	0.50	0.41
swell	CPA	150	Prototype	0.01	4	None	None	89	Yes	4.45	0.08	0.51	0.36
lose	Ontonotes	150	Prototype	0.02	5	US	None	30	29 HITs	2.96	0.17	0.76	–
lose[†]	Ontonotes	150	Prototype	0.02	5	None	None	30	0 HITs	0.00	–	–	–
lose	Ontonotes	150	Prototype	0.02	5	India	None	222	222	22.20	0.04	0.50	–
shine	CPA	84	Pairwise	0.01	6	None	33%	616	Yes	30.80	0.07	0.67	0.54
shine[*]	CPA	84	Prototype	0.01	6	None	33%	580	–	29.00	0.07	0.65	0.54
open	Ontonotes	100	Pairwise	0.01	5	US	33%	1030	941 HITs	49.84	0.20	0.71	0.55
open[*]	Ontonotes	100	Prototype	0.01	5	US	33%	657	–	32.85	0.22	0.68	0.48
rain	CPA	79	Pairwise	0.03	10	US	27%	344	Yes	51.60	0.68	0.84	0.72
rain[*]	CPA	79	Prototype	0.03	10	US	27%	172	–	26.25	0.63	0.71	0.79

Table 2: Summary of experiments run under different system configurations.
[*] Indicates that experiment was simulated based on full-pairwise results.
[†] This experiment was run with a custom made qualification requirement that nobody completed

5.1. F-scores

The F-score (Zhao et al., 2005; Agirre et al., 2007) is a set-matching measure. Precision, recall, and their harmonic mean (van Rijsbergens F-measure) are computed for each cluster/sense class pair. Each cluster is then matched to the class with which it achieves the highest F-measure. The F-score is computed as a weighted average of the F-measure values obtained for each cluster.

For full-pairwise results, we performed a series of experiments using Markov Clustering (MCL) to produce final clusters from the data. The MCL algorithm takes as input a graph encoded as a series of nodes and weighted edges between them, and runs flow simulations to prune the graph and generate hard clusters (Van Dongen, 2000).

The key parameter in MCL is the inflation value, which affects cluster granularity. We ran MCL (http://micans.org/mcl/) across a range of inflation values for several of the pairwise judgment graphs. Another parameter is the initial edge weights between pairs of sentences. We initialized edges between all sentences to a value of 1 and then added 1.5 to an edge for each worker who judged two sentences as similar, subtracted 1 from edge weights when workers judged two sentences as different, and added 0.25 for unclear judgments. This setting yielded better results than some of the other configurations (such as initializing all edge weights to 0, weighting same or different judgments equally, and varying the weight of unclear judgments).

Table 2 shows the f-scores of the runs that generated the results closest to the gold standard. Inflation values achieving best f-scores varied between different targets, which is expected, given that the granularity of senses also varies. However, as one can see from the overall kappa and agreement values, as well as the f-scores obtained in prototype simulations for the same targets, the quality of the annotation in some experiments was relatively poor, and the noise in the annotation may also partially account for the discrepancy in the best inflation value. For *shine-v*, the best f-score of .54 was achieved with inflation value 5.1. For *open-v*, the best f-score of .55 was achieved with an inflation value of 4.03. For *rain-v*, the best f-score of .72 was achieved with an inflation value of 2.35.

5.2. Worker Error Rates

As mentioned in section 3.1, worker error rates are estimated using software that attempts to separate out true error rate from biases that some workers exhibit. The main procedure to estimate worker cost runs as follows: for each worker, prior probabilities of assigning a label i to an object are computed. Next, for each assignment, the best possible estimate for the true label of the assignment is computed. Given the true label, the expected cost of a worker assigning that label is calculated. Finally, knowing how often the worker assigns a label and the expected cost, the average misclassification cost of each worker is calculated, a scalar value for which perfect workers have costs of zero and random workers or spammers have high expected costs (Ipeirotis et al., 2010).

6. Source Data

Our methodology does not presume the existence of a gold standard sense-annotated corpus. HITs can be created for any word as long as there are a sufficient number of example sentences that can be used to populate a comprehensive graph. However, in the present experiments, our HITs are populated with gold standard sense-annotated sentences from the OntoNotes and CPA (CPA, 2009). Typically, these sources have too many example sentences across the various senses of a single word to facilitate fast and inexpensive MTurk annotation. To keep trials inexpensive and consistent, for every word, a subset of is sampled from the gold standard source, to be run through MTurk. In the initial prototype-style experiments, 150 sentences were used. In pairwise experiments, 80 sentences were sampled. The target sentences are chosen to preserve the original distribution of senses present in the gold standard corpus, with a minimum threshold below which example sentences for a sense are not removed.

7. Conclusion

The set of experiments we conducted suggest that there has been a significant change in the crowdsourcing marketplace since its early days just a few short years ago. However, preliminary results indicate that it is possible to create a high-quality word sense inventories using the sense-crowdsourcing methodology outlined above without significant modification to the HIT structure and cost. In a large-scale lexical development effort, however, even a small adjustment to the per-HIT pay would be costly.

There is also a great deal of flexibility in HIT structure for the MTurk marketplace. Determining the exact HIT layout that is best for this task is more of an art than a science, requiring a balance between user-friendliness, cost considerations, and the cognitive limitations of MTurkers.

8. References

E. Agirre, B. Magnini, O. Lopez de Lacalle, A. Otegi, G. Rigau, and P. Vossen. 2007. Semeval-2007 task 01: Evaluating wsd on cross-language information retrieval. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 1–6, Prague, Czech Republic, June. Association for Computational Linguistics.

C. Akkaya, A. Conrad, J. Wiebe, and R. 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*, CSLDAMT '10, pages 195–203, Stroudsburg, PA, USA. Association for Computational Linguistics.

Callison-Burch. C. and M. Dredze, editors. 2010. *CSLDAMT '10: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, Stroudsburg, PA, USA. Association for Computational Linguistics.

CPA. 2009. Corpus Pattern Analysis.

P. Hanks and J. Pustejovsky. 2005. A pattern dictionary for natural language processing. *Revue Française de Linguistique Appliquée*.

P. Hanks. 2000. Do word meanings exist? *Computers and the Humanities*, 34(1):205–215.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA, June. Association for Computational Linguistics.

P.G. Ipeirotis, F. Provost, and J. Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM.

R. Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 105–112, Sydney, Australia, July. Association for Computational Linguistics.

M. Palmer, H. Dang, and C. Fellbaum. 2007. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Journal of Natural Language Engineering*.

J. Pustejovsky. 1995. *Generative Lexicon*. Cambridge (Mass.): MIT Press.

D.G. Rand. 2011. The promise of mechanical turk: How online labor markets can help theorists run behavioral experiments. *Journal of theoretical biology*.

A. Rumshisky, J. Moszkowicz, and M. Verhagen. 2009. The holy grail of sense definition: Creating a sensedisambiguated corpus from scratch. In *Proceedings of 5th International Conference on Generative Approaches to the Lexicon (GL2009)*. Citeseer.

A. Rumshisky. 2008. Resolving polysemy in verbs: Contextualized distributional approach to argument semantics. *Distributional Models of the Lexicon in Linguistics and Cognitive Science, special issue of Italian Journal of Linguistics / Rivista di Linguistica*. forthcoming.

A. Rumshisky. 2011. Crowdsourcing word sense definition. *Proceedings of 5th Linguistic Annotation Workshop, ACL HLT 2011*, page 74.

R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

S. Van Dongen. 2000. A cluster algorithm for graphs. *Report-Information systems*, (10):1–40.

Ying Zhao, George Karypis, and Usama M. Fayyad. 2005. Hierarchical clustering algorithms for document datasets. 10:141–168.