Data Collection for a Production Dialogue System: A Clinc Perspective

Yiping Kang **Yunqi Zhang** **Jonathan K. Kummerfeld** **Parker Hill**
Johann Hauswald **Michael A. Laurenzano** **Lingjia Tang** **Jason Mars**
Clinc, Inc.
Ann Arbor, MI, USA
{yiping,yunqi,jkk,parkerhh,johann,mike,lingjia,jason}@clinc.com

Abstract

Industrial dialogue systems such as Apple Siri and Google Assistant require large scale diverse training data to enable their sophisticated conversation capabilities. Crowdsourcing is a scalable and inexpensive data collection method, but collecting high quality data efficiently requires thoughtful orchestration of crowdsourcing jobs. Prior study of data collection process has focused on tasks with limited scope and performed intrinsic data analysis, which may not be indicative of impact on trained model performance. In this paper, we present a study of crowdsourcing methods for a user intent classification task in one of our deployed dialogue systems. Our task requires classification over 47 possible user intents and contains many intent pairs with subtle differences. We consider different crowdsourcing job types and job prompts, quantitatively analyzing the quality of collected data and downstream model performance on a test set of real user queries from production logs. Our observations provide insight into how design decisions impact crowdsourced data quality, with clear recommendations for future data collection for dialogue systems.

1 Introduction

Large, high quality corpora are crucial in the development of effective machine learning models in many areas, including Natural Language Processing (NLP). The performance of the machine learning models, especially deep learning models, depend heavily on the quantity and quality of the training data. Developing dialogue systems such as Apple Siri, Google Assistant and Amazon Alexa poses a significant challenge for data collection as we need to do rapid prototyping and bootstrapping to train new virtual assistant capabilities. The use of crowdsourcing has enabled the creation of large corpora at relatively low cost (Snow et al., 2008) and is critical in collecting the quantities of data required to train models with high accuracy. However, designing effective methodologies for data collection with the crowd is largely an open research question (Sabou et al., 2014).

From the perspective of Clinc, a young AI company creating innovative conversational AI experiences, there exists a major challenge when collecting data to build a dialogue system. We have observed that the complexity of building production-grade dialogue system is often substantially greater than those studied in the research community. For example, one of our production deployed dialogue systems requires intent classification among 47 different intents to meet product specifications, whereas most academic datasets for text classification only have a small number (i.e., 2–14) of classes (Zhang et al., 2015). The few datasets that have a large number of classes, such as RCV-1 (Lewis et al., 2004), distribute intents across many distinct topics. We address the significantly more challenging problem of handling many intents within a single domain, specifically personal finance and wealth management, requiring the classifier to carefully distinguish between nuanced intent topics. Therefore, a large amount of high-quality training data tailored to our targeted problem is critical for creating the best user experience in our production dialogue system.

Crowdsourcing offers a promising solution by massively parallelizing data collection efforts across a large pool of workers at relatively low cost. Because of the involvement of crowd workers, collecting high-quality data efficiently requires careful orchestration of crowdsourcing jobs, including their instructions and prompts. In order to collect the large-scale tailored dataset we need via crowdsourcing, there are several research questions we need to answer:

- How can we evaluate the effectiveness of crowdsourcing methods and the quality of the datasets collected via these methods?

- During the data collection process, how can we identify the point when additional data would have diminishing returns on the performance of the downstream trained models?
- Which crowdsourcing method yields the highest-quality training data for intent classification in a production dialogue system?

There is limited work on effective techniques to evaluate a crowdsourcing method and the data collected using that method. Prior work has focused on intrinsic analysis of the data, lacking quantitative investigation of the data's impact on downstream model performance (Jiang et al., 2017). In this paper, we propose two novel model-indepedent metrics to evaluate dataset quality. Specifically, we introduce (1) `coverage`, quantifying how well a training set covers the expression space of a certain task, and (2) `diversity`, quantifying the heterogeneity of sentences in the training set. We verify the effectiveness of both metrics by correlating them with the model accuracy of two well-known algorithms, SVM (Cortes and Vapnik, 1995) and FastText (Joulin et al., 2017; Bojanowski et al., 2017). We show that while `diversity` gives a sense of the variation in the data, `coverage` closely correlates with the model accuracy and serves as an effective metric for evaluating training data quality.

We then describe in detail two crowdsourcing methods we use to collect intent classification data for our deployed dialogue system. The key ideas of these two methods are (1) describing the intent as a scenario or (2) providing an example sentence to be paraphrased. We experiment multiple variants of these methods by varying the number and type of prompts and collect training data using each variant. We perform metric and accuracy evaluation of these datasets and show that using a mixture of different prompts and sampling paraphrasing exmples from real user queries yield training data with higher `coverage` and `diversity` and lead to better performing models. These observations have impacted the way that we collect data and are improving the quality of our production system.

2 Many-intent Classification

We focus on a specific aspect of dialogue systems: intent classification. This task takes a user utterance as input and classifies it into one of the predefined categories. Unlike general dialogue annotation schemes such as DAMSL (Core and Allen, 1997), intent classification is generally domain-specific. Our system requires classification over 47 customer service related intents in the domain of personal finance and wealth management. These intents cover a large set of topics while some of the intents are very closely related and it requires the classifier to identify the nuanced differences between utterances. For example, user's queries to see a list of their banking transactions can often be very similar to their queries to see a summary of historical spending, e.g., *"When did I spend money at Starbucks recently?"* vs. *"How much money did I spend at Starbucks recently?"*.

Test Methodology Our test data contains a combination of real user queries from a deployed system and additional cases manually constructed by developers. This combination allows us to effectively measure performance for current users, while also testing a broad range of ways to phrase queries. Our test set contains 3,988 sentences labelled with intents.

3 Training Data Quality Metrics

When we look to improve a model's performance, there are generally two approaches that we can take: improve the model and inference algorithm and/or improve the training data. There is currently no reliable way to help us identify whether the training data or the model structure is the current bottleneck. One solution is to train actual models using the training set and measure their accuracy with a pre-defined test set. However, if only a single algorithm is used, over time this evaluation may lead to a bias, as the training data is tuned to suit that specific algorithm. Using a suite of different algorithms avoids this issue, but can be very time consuming. We need an algorithm-independent way to evaluate the quality of training data and its effectiveness at solving the target task. In this section, we introduce two metrics to achieve this, `diversity` and `coverage`.

Diversity We use `diversity` to evaluate the heterogeneity of the training data. The idea behind `diversity` is that the more diverse the training data is, the less likely a downstream model will overfit to certain words or phrases and the better it will generalize to the testing set.

We first define a pairwise sentence distance measure. For a given pair of sentences, a and b, we calculate the reverse of the mean Jaccard Index between the sentences' n-grams sets to represent the semantic distances between the two sentences:

$$D(a,b) = 1 - \sum_{n=1}^{N} \frac{|n\text{-grams}_a \cap n\text{-grams}_b|}{|n\text{-grams}_a \cup n\text{-grams}_b|} \quad (1)$$

where N is the maximum n-gram length. We use $N = 3$ in our experiments.

Our pairwise score is similar to the PINC score (Chen and Dolan, 2011), except that we use the n-grams from the union of both sentences instead of just one sentence in the denominator of Equation 1. This is because the PINC score is used in paraphrasing tasks to measures how much a paraphrased sentence differ from the original sentence and specifically rewards n-grams that are unique to the paraphrsed sentence. Our metric measures the semantic distance between two sentences and treat the unique n-grams in both sentences as equal contribution to the distance.

We define the `diversity` of a training set as the average distance between all sentence pairs that share the same intent. For a training set X, its `diversity` ($DIV(X)$) is:

$$DIV(X) = \frac{1}{|I|} \sum_{i=1}^{I} \frac{1}{|X_i|^2} \left[\sum_{a}^{X_i} \sum_{b}^{X_i} D(a,b) \right] \quad (2)$$

where I is the set of intents and X_i is the set of sentences with intent i in the training set X.

Coverage We now introduce `coverage`, a new metric designed to model how well a training dataset covers the complete space of ways an intent can be expressed. We use our test set as an approximate representation of the expression space for our classification task. As described in § 2, our test set is constructed primarily with real user queries collected from the log of a deployed system and annotated by engineers.

To measure `coverage` of a training set given a test set, we first identify, for each test sentence, the most similar training sentence with the same intent, according to the pairwise sentence distance measure $D(a,b)$ defined in Equation 1. We then derive `coverage` by averaging the shortest distances for all sentences in the test set. For a given test set, we would want the training set to have as high `coverage` as possible. Specifically, for a training set X and a test set Y:

$$CVG(X,Y) = \frac{1}{|I|} \sum_{i=1}^{I} \frac{1}{|Y_i|} \sum_{b}^{Y_i} \max_{a}^{X_i} (1 - D(a,b)) \quad (3)$$

where I is the set of intents and X_i and Y_i are the sets of utterances labeled with intent i in the training (X) and test (Y) sets, respectively.

Correlating Metrics with Model Accuracy In order to evaluate the effectiveness of `diversity` and `coverage` at representing the training data quality, we collect training data via different methods and of varying sizes, train actual models, measure their accuracy and investigate the correlation between the metrics and the accuracy. We consider two well-known algorithms that have publicly available implementations: a linear SVM and FastText, a neural network-based algorithm.

SVM Support Vector Machines (Cortes and Vapnik, 1995) are a widely used and effective approach for classification tasks. We use a linear model trained with the SVM objective as a simple baseline approach.

FastText We also consider a recently developed neural network approach (Joulin et al., 2017; Bojanowski et al., 2017). This model has three steps: (1) look up vectors for each n-gram in the sentence, (2) average the vectors, and (3) apply a linear classifier. The core advantage of this approach is parameter sharing, as the vector look-up step places the tokens in a dense vector space. This model consistently outperforms linear models on a range of tasks.

For all experiments we apply a consistent set of pre-processing steps designed to reduce sparseness in the data: we lowercase the text, remove punctuation, replace each digit with a common symbol, expand contractions, and lemmatize (using NLTK for the last two).

4 Crowdsourcing Data Collection Methods

We consider two aspects of a crowdsourcing setup: the *template* style, and the *prompt*. The template defines the structure of the task, including its instructions and interface. Prompts are intent-specific descriptions or examples that define the scope of each task and guide workers to supply answers related to the target intent. We define a set of prompts for each intent and populate a

React to a Scenario

Suppose you have a device that has a Siri-like app for your bank account that acts as a customer service agent and can handle questions about **your bank account's balance.**

Given the original scenario described below that is related to your bank account, **supply 5 creative ways of asking the intelligent device to assist your situation**.

"You want to ask about the balance of your bank account."

Figure 1: An example of scenario-driven task instructions. The template sets up a real-world situation and asks workers to provide a response as if they are in that situation. The prompt shown here is for collecting data for the intent 'balance'.

template with each prompt to create a complete crowdsourcing job. We study two types of templates: scenario-driven (§ 4.1) and paraphrasing (§ 4.2), and two methods of generating prompts: manual generation (§ 4.1 and 4.2) and test set sampling (§ 5.3). A data collection method is the combination of a template and a prompt generation method. In this section, we describe each method and its variants.

4.1 Scenario-driven

The instructions for a scenario-driven job describe a real-world situation and ask the worker to provide a response as if they are in the situation. Figure 1 shows an example job for the intent of "asking about your bank account balance". Table 1 shows additional example prompts for generic and specific scenarios. Scenario-driven jobs simulate real world situations and encourage workers to create natural questions and requests resembling real user queries.

We consider two variations on the scenario-driven setup. Generic scenarios describe the situation in which the target intent applies, without specific constraints. For example, a generic scenario for the intent 'balance' is *"You want to know about your account balance"*. Specific scenarios refine the description by adding details. These are intended to encourage workers to write responses with more entities and constraints. These jobs also add specific information that the worker needs to include in their response. For example, a specific scenario for the intent 'balance' is *"You'd like to know the balance of one of your accounts. (Please specify the account you want to inquire about in your responses)"*.

For each intent, we use one generic scenario and three specific scenarios. To evaluate the different scenario types, we collected data with either generic scenarios only, specific scenarios only, or a combination of both. The mixed setting contains equal contributions from all four scenarios (one generic and three specific). In our experiments, we keep the number of training samples per intent balanced across intents regardless of the number of total training examples.

Paraphrase Sentence

Given the following sentence, **supply 5 creative ways of rephrasing the same sentence**.

Assume the original question is in regards to **your bank account's balance**.

"What is the balance of my bank account?"

Figure 2: An example of a paraphrasing task instructions.

4.2 Paraphrasing

Paraphrasing jobs provide an example sentence and ask workers to write several creative paraphrases of it. Figure 2 shows an example of job instructions for paraphrasing the sentence *"What is the balance of my bank account?"* To make sure we can directly compare the results of paraphrasing and scenario-driven jobs, we convert each scenario used in § 4.1 into a user question or command, which is provided as the sentence to paraphrase. As a result, there are two types of paraphrasing prompts: generic prompts and specific prompts. Table 1 shows example pairs of scenarios and paraphrasing prompts. Like in the scenario-driven case, we construct training sets with three different mixes of prompts, generic only, specific only and a combination of both.

5 Evaluation

In this section, we first verify that `diversity` and `coverage` provide insight regarding training data quality. We compare trends in these metrics with trends in model accuracy as the amount of training data is increased. We then evaluate the performance of the scenario-driven and paraphrase methods and their variants by comparing the quality of training data collected via these methods. Finally, we explore sampling paraphrasing examples from the test set and compare against manually generation by engineers.

Type	Scenario	Paraphrasing
Generic	You want to learn about your spending history.	"Show me my spending history."
Specific	You want to learn about your spending history during a specific period of time.	"Show me my spending history in the last month. (Use different time periods in your answers)."
Generic	You want to ask about your income.	"What's my income?"
Specific	You want to ask about your income from a specific employer.	"How much money did I make from Company A? (Use different employers in your answers.)"

Table 1: Examples of generic and specific scenario description and paraphrasing prompts.

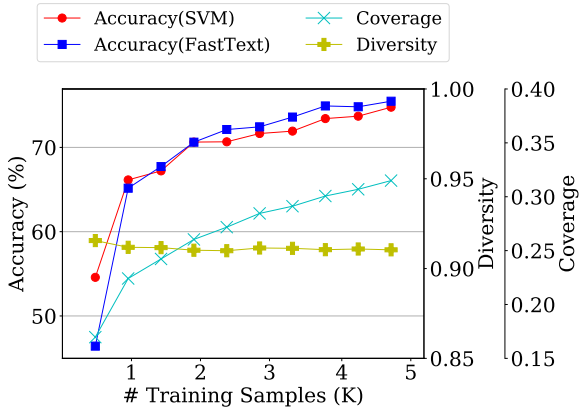

Figure 3: Accuracy, `coverage` and `diversity` for scenario-driven jobs as the training data size increases. This data is collected using a mixture of generic and specific scenarios.

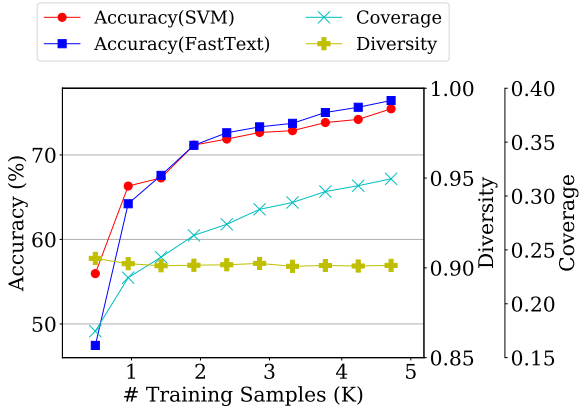

Figure 4: Accuracy, `coverage` and `diversity` for paraphrasing jobs as the training data size increases. This data is collected using a combination of generic and specific paraphrase examples.

5.1 Correlating Diversity and Coverage with Model Accuracy

Figure 3 and 4 show `diversity`, `coverage`, and accuracy of the SVM and FastText models as we vary the number of training examples for scenario-driven and paraphrase-based jobs, respectively. In this experiment, we use a combination of both generic and specific scenarios and paraphrasing examples.

We observe that for both scenario and paraphrase jobs, the `diversity` starts high (> 0.90) with a few hundred training samples and stay stable as training data size increases. This means that the new training examples generally have a low percentage of n-grams overlap and a long distance ($D(a, b)$) with the existing examples, therefore maintaining the overall high `diversity`. This indicates that the newly introduced examples are generally creative contributions from the crowd and not repeats or straightforward rephrase of the existing samples with the same words.

`coverage` starts low with a few hundred training examples and steadily increases as the training set grows. This indicates that the new training examples contain sentences that are semantically closer to the test set sentences than exisiting training examples, increasing the training set's scope to better cover the expression space represented by the test set. The accuracy of both SVM and FastText models follow a very similar trend to that of `coverage`, gradually increasing as more training samples are collected. The correlation between model accuracy and `coverage` shows that `coverage` is a more effective metric than `diversity` in evaluating the quality of a training set without training models.

We also observe diminishing returns in `coverage` as more data is collected. This trend roughly correlates with the diminishing return in accuracy of the SVM and FastText models. The trend in `coverage` provides insight into improvements in training data quality, which can inform the choice of when to stop collecting more data and start focusing on improving algorithms. This is further demonstrated by the way FastText consistently outperforms the SVM model when their accuracies and `coverage` of the training data saturate, indicating that the algorithm is the bottleneck for improving accuracy instead of the amount of training data.

Key Insights (1) `diversity` stays relatively constant with a high value as more training samples are collected, indicating that new distinct training examples are being introduced. (2) `coverage` continuously improves as data scales, showing that the training data is becoming more

Template	Type	Accuracy SVM	FastText	CVG	DIV
Scenario	Generic	68.49	69.70	0.30	0.90
	Specific	65.86	68.10	0.29	0.89
	Both	**74.77**	**75.48**	**0.32**	**0.91**
Paraphrase	Generic	68.60	70.50	0.30	0.88
	Specific	67.80	67.77	0.29	0.87
	Both	**75.46**	**76.44**	**0.32**	**0.90**

Table 2: Accuracy, `coverage` and `diversity` for the six template + prompt conditions considered, all with ~4.7K training samples.

effective at covering the expression space defined by the test set. The trend of `coverage` closely correlates with the trend in model accuracy, indicating that `coverage` is an effective metric at evaluating training data quality without requiring model training.

5.2 Comparing Scenario and Paraphrase Based Collection and Their Variants

Table 2 summarizes the model accuracy, `coverage` and `diversity` of both scenario-driven and paraphrase-based jobs. We studied three variants for each job type, where we use different mixtures of prompt type (generic prompts only, specific prompts only and combined prompts). All configurations are evaluated using training data of the same size (~4.7K) and on the same test set.

For both scenario and paraphrase jobs, using a mixture of both generic and specific prompts yields training data with higher `coverage` and models with higher accuracy than using only generic or specific prompts.

Table 2 compares scenario and paraphrasing jobs. As described in § 4.2, the paraphrasing examples were based on the scenario descriptions so we are only measuring the impact of different job types. The two approaches lead to similar results across all metrics. This shows that despite the instructions being distinctly different, scenario-driven and paraphrasing jobs generally yield training data of similar quality.

Key Insights (1) A mixture of generic and specific scenarios and paraphrasing examples yields the best training data given a fixed number of training examples, in terms of both `coverage` of the training set and the accuracy of the downstream models. (2) Scenario-driven and paraphrasing based crowdsourcing jobs yield similar quality training data despite having different job instructions and templates.

	Accuracy SVM	FastText	CVG	DIV
Manual generation	75.46	76.44	0.32	0.90
Test set sampling	**83.05**	**84.69**	**0.40**	**0.92**

Table 3: Comparison of manually generating prompts and sampling from test set, evaluated on half of the test data (kept blind in sampling).

# of paraphrasing prompts	Accuracy SVM	FastText	CVG	DIV
1	71.46	71.69	0.31	0.88
2	78.34	79.33	0.36	0.91
3	81.47	82.67	0.39	0.91
4	83.05	84.69	0.40	0.92
5	**84.61**	**85.96**	**0.41**	**0.92**

Table 4: Accuracy, `coverage` and `diversity` of paraphrasing jobs using 1-5 prompts sampled from the test set, with constant training set size (~4.7K).

5.3 Sampling Prompts from the Test Set

We now investigate a different way to generate the prompts used for the crowdsourcing jobs. In the context of scenario-driven and paraphrasing jobs, prompts are the scenario descriptions and the example sentences provided to workers to rephrase, respectively. In § 4.1 and 4.2, engineers manually generated the prompts based on the definition of each intent. While manual generation guarantees high quality prompts, it requires engineering effort and could potentially be biased by the engineer's perspective. One way to reduce such effort and bias is to automatically source prompts based on real user queries.

We divide the test set into two equal halves. For each intent, we randomly sample 5 utterances from the first half of the test set and use them as prompts to construct paraphrasing jobs. The second half of the test set is kept entirely blind and used for evaluation.

Manual Generation vs. Test Set Sampling Table 3 shows the accuracy, `coverage` and `diversity` of a training set collected with 4 manually generated paraphrasing examples vs. with 4 paraphrasing examples sampled from the first half of the test set. The accuracy for both methods is evaluated on the second half of the test set (kept blind from prompt sampling). The results show that sampling from the test set leads to a training set that has 8% higher `coverage`, 2% higher `diversity` and yields models with 8% higher accuracy, compared to manual generation.

Varying the Number of Prompts Table 4 shows the accuracy, `coverage` and `diversity` of training data collected

using a varying number (1-5) of unique paraphrasing examples sampled from the test set. We observe that test set accuracy improves as we use more unique prompts but eventually there are diminishing returns. Increasing the number of prompts from 1 to 2 increases the accuracy by 6.9% and 7.6% for SVM and FastText, respectively, while increasing the number of prompts from 4 to 5 improves their accuracy by only 1.6% and 1.3%.

6 Related work

This study complements a line of work on understanding how to effectively collect data with non-expert workers. The closest work is Jiang et al. (2017)'s study of a range of interface design choices that impact the quality and diversity of crowdsourced paraphrases. However, their work focused on intrinsic evaluation of the paraphrases only, whereas we explore the impact on performance in a downstream task. The variations we consider are also complementary to the aspects covered by their study, providing additional guidance for future data collection efforts.

In terms of the variations we consider, the closest work is Rogstadius et al. (2011), who also considered how task framing can impact behavior. Their study made a more drastic change than ours though, attempting to shift workers' intrinsic motivation by changing the perspective to be about assisting a non-profit organization. While this shift did have a significant impact on worker behavior, it is often not applicable.

More generally, starting with the work of Snow et al. (2008) there have been several investigations of crowdsourcing design for natural language processing tasks. Factors that have been considered include quality control mechanisms (Rashtchian et al., 2010), payment rates and task descriptions (Grady and Lease, 2010), task naming (Vliegendhart et al., 2011), and worker qualification requirements (Kazai et al., 2013). Other studies have focused on exploring variations for specific tasks, such as named entity recognition (Feyisetan et al., 2017). Recent work has started to combine and summarize these observations together into consistent guidelines (Sabou et al., 2014), though the range of tasks and design factors makes the scope of such guidelines large. Our work adds to this literature, introducing new metrics and evaluation methods to guide crowdsourcing practice.

7 Conclusion

Training data is the key to building a successful production dialogue system, and efficiently collecting large scale robust training data via crowdsourcing is particularly challenging. In this work, we introduce and characterize two training data quality evaluation metrics. We verify their effectiveness by training models of well-known algorithms and correlating the metrics with model accuracy. We show that an algorithm-independent `coverage` metric is effective at providing insights into the training data and can guide the data collection process. We also studied and compared a range of crowdsourcing approaches for collecting training data for a many-intent classification task in one of our deployed dialogue systems. Our observations provide several key insights that serve as recommendations for future dialogue system data collection efforts, specifically that using a mixture of generic and specific prompts and sampling prompts from the real user queries yields better quality training data.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Portland, Oregon, USA.

Mark G. Core and James F. Allen. 1997. Coding dialogs with the damsl annotation scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Cambridge, MA.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Oluwaseyi Feyisetan, Elena Simperl, Markus Luczak-Roesch, Ramine Tinati, and Nigel Shadbolt. 2017. An extended study of content and crowdsourcing-related performance factors in named entity annotation. *Semantic Web*.

Catherine Grady and Matthew Lease. 2010. Crowdsourcing document relevance assessment with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 172–179, Los Angeles.

Youxuan Jiang, Jonathan K. Kummerfeld, and Walter S. Lasecki. 2017. Understanding task design trade-offs in crowdsourced paraphrase collection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 103–109, Vancouver, Canada.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain.

Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. 2013. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information Retrieval*, 16(2):138–178.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

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.

Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *International AAAI Conference on Web and Social Media*.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. 2014. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii.

Raynor Vliegendhart, Martha Larson, Christoph Kofler, Carsten Eickhoff, and Johan Pouwelse. 2011. Investigating factors influencing crowdsourcing tasks with high imaginative load. In *Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining*, pages 27–30.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.